SZAKDOLGOZAT

Ráthonyi Tamás

Debrecen

2008

Debreceni Egyetem Informatikai Kar

Ontológiák és szolgáltatásorientált architektúrák

Témavezető:

Dr. Juhász István

egyetemi adjunktus

Készítette:

Ráthonyi Tamás

Programtervező Informatikus B.s.c

Debrecen

2008

Tartalomjegyzék

1. fejezet: Ontológiák

Bevezetés

Az ontológia fogalom már az ókorban, Arisztotelész idejében is létezett. Ekkor a filozófia egyik ága volt (a lételmélet), majd szemléletmódja és módszerei fokozatosan megjelentek a többi tudományterületen. Jelenleg a mesterséges intelligencián belül az ontológia az egyik olyan részterület, melyre a legtöbb figyelmet fordítják, és előszeretettel alkalmazzák a szemantikus webnél is.

Az ismeret vagy a tudás megosztása kontextusban az ontológia a fogalomalkotás, fogalom feltérképezés specifikációját, vagyis a konkrét körülírását jelenti. Az ontológia azon fogalmak és viszonyok leírásának fajtája (akárcsak egy program formális specifikációja), amelyek egy ágens (intelligens program) vagy egy sor ágens vonatkozásában fenn állnak vagy léteznek. Ez a meghatározása azzal az ontológia szóhasználattal vág egybe, amely szerint az ontológia egy sor fogalom-meghatározás, de annál általánosabb.

Ontológiákat az MI területén abból a célból készítenek, hogy lehetővé váljon elosztott tudás létrehozása és ismételt felhasználása. Ebben az értelemben az ontológia az ontológiai elkötelezettségek, vagy értelmezési előírások készítésére használt specifikáció.

Ebben a fejezetben röviden ismertetem a szemantikus webet, az ontológia fogalmát és az OWL-t (Web Ontology Language).

A szemantikus web felé

Napjainkban a weben az információk nagyon sokféle alakban vannak jelen: a klasszikus szöveges információk mellett akár fényképek, két- és háromdimenziós grafikák, videók, zenék formájában is. A keresett információk sokszor csak különböző weblapokon találhatók meg. Gyakran emlegetett példa az utazásszervezés. A számunkra szükséges adatok egy része valószínűleg a légitársaság lapján, míg mások pedig például a szálloda honlapján

lesznek elérhetőek. Ebben az esetben még problémát okozhat az is, ha a különböző cégek különböző terminológiát használnak ugyanarra a fogalomra.

Mindezek ellenére a web csodálatos módon működik, az emberek könnyen tudnak inkoherens vagy nem teljes információkat használni, megértik a fénykép tartalmát, áthidalnak terminológiai különbségeket, és sokszor félkész információkat teljessé tudnak alakítani. Ez azonban a számítógépekre nem igaz. Mint ahogy ezt mindannyian tudjuk, a számítógépek alapvetően „buták". Az ágensek alkalmazhatóságához a rendelkezésre álló adatokról, az adatok közötti összefüggésekről, azok esetleges ekvivalenciájáról, nyelvéről sokkal precízebb információkat kell biztosítani. Csak ezek alapján lehet elvárni, hogy az ágensek képesek legyenek kihasználni azt a fantasztikus mennyiségű adatot, amely ma a világhálón rendelkezése áll. Márpedig ágens már most rengeteg van, és az ezekre épülő alkalmazások száma, és főleg az irántuk támasztott felhasználói igények állandóan növekednek.

Egyre többször fordul elő olyan eset, hogy az alkalmazáson belül koherens módon kombinálni kell weben lévő adatokat, azaz a weben lévő adatok között kapcsolatokat kell létrehozni, definiálni. Itt az „adat" lehet egyszerűen a weben lévő állomány (például egy on-line címlista), elérhető adat (például egy adatbázisban lévő rekord elérése valamilyen alkalmazási felületen keresztül), vagy metaadat: vagyis adat az adatról (például egy könyvtári katalóguscédula elektronikus változata). A szemantikus web ezt a problémakört próbálja megoldani.

Ha röviden akarjuk jellemezni, akkor azt mondhatjuk, hogy a szemantikus web célja egy olyan infrastruktúra létrehozása, amely lehetővé teszi a weben lévő adatok integrálását, a közöttük levő kapcsolatok definiálását és jellemzését, illetve az adatok értelmezését.

Ahhoz, hogy ezt az infrastruktúrát létrehozzuk, néhány fontos építőelemre van szükség, melyek közül néhányat később ismertetek ebben a fejezetben:

1. Az adatokat egyértelműen meg kell „címezni" a weben, vagyis el kell őket nevezni (URI=URL+URN)
2. Szükség van egy precíz adatmodellre, amely formális keretet ad az adatok egymáshoz való kapcsolatának definiálására, és a kapcsolatok leírására (RDF[30])
3. Az adatok közötti kapcsolatokat illetve a kapcsolt adatok referenciáit el kell tudni érni, le kell tudni kérdezni (SPARQL[1])

4. A kapcsolatok leírására szolgáló terminológiát tudni kell definiálni (RDFS[2], OWL[40], SKOS[3])
5. A kapcsolatokon illetve azok leírásán logikai következtetéseket kell tudni levonni (OWL[40], RIF[39])

Az RDF illetve újabban az RDF+SPARQL kombináció már érdekes és fontos alkalmazási lehetőségeket rejt magában. A teljes szemantikus web azonban ennél többet igényel.

Két fő terület van, amely további technológiákat kíván, nevezetesen:

1. Milyen terminológiát használhatunk egy alkalmazáson belül? Egyáltalán, hogyan lehet osztályozni a rendelkezésre álló erőforrásokat?
2. Hogyan jellemezhetjük az erőforrások közötti logikai kapcsolatokat? Mikor mondhatjuk, hogy két erőforrás (például két különböző nevű tulajdonság) ugyanazon fogalomra utal? Lehet-e a tulajdonságok logikai viselkedését jellemezni (például hogy szimmetrikus vagy tranzitív, hogy valójában egy függvény, stb.)?

Világos, hogy ezek a kérdések felvetődnek minden komolyabb szemantikus web alkalmazás esetén; de az is világos, hogy ezen kérdések megválaszolására egy bonyolultabb rendszer kell, amely ki kell egészítse az RDF egyszerű adatmodelljét. Az is viszonylag hamar kiderül, hogy az összes kérdés megválaszolása rendkívül bonyolulttá válhat, ezért érdemes technológiák egy családját kifejleszteni, amelyek közül a felhasználó igénye szerint választhat. Így fejlődött ki a W3C keretében az RDFS, OWL, SKOS, és indult el 2005 legvégén az RIF fejlesztése. Ezek közül az RDFS és az OWL technológiák a legkiforrottabbak.

Az alapvető megközelítés fogalomjegyzékek, ontológiák használata, illetve a tradicionális ontológiai módszerek adaptálása RDF környezetekre. De mit is takar az ontológia fogalom?

Régen és most

Ontológia a filozófiában és tudományban

Az ontológia eredetileg a filozófia egyik ágazata, a lételmélet**,** amely a létező dolgok szisztematikus számbavételével foglalkozik. Az ontológia jellemzően kvalitatív, minőségi jellegű, szemben a többi tudománnyal, amelyek kvantitatívak, mennyiségekkel foglalkoznak. Ezek mérhető dolgokkal foglalkoznak, például azzal, hogy egy bizonyos osztály tulajdonságainak mérhető viselkedése hogyan kapcsolódik egy másik osztály tulajdonságainak viselkedéséhez. Továbbá, definíció szerint, csak az érdeklődési területükbe eső kategóriákkal (az adott terület alapvető formáival és viszonyaival) foglalkoznak. A filozófia ontológusa nem a mérhető világgal, hanem a mértékek ontológiájával, a kategóriákon túli kapcsolatokkal foglalkozik – beleértve a különböző tudományterületek eltérő tartományainak kategóriái közötti kapcsolatokat, valamint a hétköznapi gondolkodásban előforduló fogalmakat, objektumokat.

Tehát egy tudományterület ontológiája nem más, mint az adott területre jellemző kategóriákat (fogalmakat, objektumokat, kifejezéseket), illetve a köztük fennálló kapcsolatokat írja le jelentésükkel együtt. Minden ontológia megad egy olyan kommunikációs szövegkörnyezetet, amelyben az adott terület fogalmai vitathatók, egyértelműen elemezhetők. Ebben van az ontológiák alkalmazásának erőssége.

Ontológia az informatikában

A mesterséges intelligenciában jelenleg elfogadott meghatározás szerint egy adott tárgyterület vonatkozásában az ontológia a fogalomalkotás explicit specifikációja: egy tárgyterület fogalmainak és az azok között fennálló kapcsolatoknak formális specifikációja, amelyhez általában természetes nyelvű leírás is társul. Egy adott tárgyterület ontológiája egy olyan reprezentációs szójegyzék, amely a tárgyterület leírandó fogalmairól és objektumairól, azok tulajdonságairól és kapcsolatairól szól. Tartalmazza azok olvasható formában leírt megnevezését, a nevek jelentését és jellemzését.

Gyakorlatilag ez azt jelenti, hogy egy ontológia egy formális fogalomgyűjtemény definícióinak halmaza, amely osztályok, relációk, függvények stb. definícióiból áll. A tudásmegosztás szempontjából fontos, hogy e definíciók az olvasótól és az alkalmazás kontextusától szemantikailag függetlenek legyenek. Azonos tárgyterületen dolgozó közösség

az ontológia közvetítésével azonos módon legyen képes értelmezni és használni a közösen használt fogalmakat, objektumokat, tulajdonságaikat és relációikat. Ezt úgy is mondják, hogy minden ilyen közösség tagjai az ontológia által specifikált elméletre nézve konzisztens – esetleg nem teljes – szótárhasználatra vonatkozó ontológiai elkötelezettséget vállalva dolgoznak.

Egy ontológiai egyezség olyan szerződés, amely az ontológia által specifikált elméletre nézve konzisztens (de nem teljes) szótárhasználatra vonatkozik – például kérdésfeltevésnél, állítások megfogalmazásánál. Egy ágenst úgy kell megépíteni, hogy az adott tárgykörben „ontológiai elkötelezettséget" vállaljon, míg ontológiák tervezésénél az a cél, hogy segítse az ágensek egymásközti tudáscseréjét.

Az MI esetében egy ontológia a következőket tartalmazza:

- a kommunikációs szövegkörnyezet úgynevezett entitásainak nevéhez kapcsolt, e nevek jelentését megadó, ember által olvasható definíciókat (egy entitás lehet osztály, reláció, függvény stb.),
- az interpretációk körét korlátozó formális axiómákat,
- az előbbi definíciókból és axiómákból képezett jól-formált szövegeket, „formulákat".

Formálisan: egy ontológia egy logikai elmélet leírása. Megjelenési formái: leíró szótár, típushierarchia (leírásokkal bővített taxonómia). Fokozatai: informális (például szójegyzék), félig-formális, formális (ontológia-nyelvű leírás).

Ontológia és taxonómia

A taxonómia a logikának az a része, amely az ismeretek rendszerezésével foglalkozik. Egy taxonómia tulajdonképpen rendszertan, pontosabban valamely leíró tudománynak az ismereti anyagát a saját rendszerező elvei alapján rendszerbe foglaló része. Bármely tudományos vizsgálódás azzal szokott kezdődni, hogy valamilyen rendbe szervezik, osztályozzák a megfigyelés alatt álló fogalmakat és objektumokat, kiépítve az adott terület fogalomhierarchiáját, taxonómiáját. Adott tárgyterületen belül egy meghatározott osztályozási szempont szerint kidolgozott taxonómiát leggyakrabban hierarchikus fával ábrázolnak.

Egy szakterületi ontológia - első közelítésben - kategóriákból álló olyan táblázat, amely táblázat minden egyes sorának entitástípusa egy hierarchikus gráf (irányított

ciklusmentes gráf, fa) valamely csomópontjához kapcsolódik. Azt már most leszögezhetjük, hogy egy ontológia jellemzően bővebb egy taxonómiánál, mivel egyben a fogalmak, objektumok leírását is tartalmazza.

Mármost, nem minden taxonómia lehet alapja egy ontológiának; annak jól-formáltnak kell lennie.

Egy taxonómia jól-formáltságának három követelménye szerint:

1. Egy taxonómiát – matematikai értelemben – irányított ciklusmentes gráffal, fával lehet leírni. Egy ilyen fa csomópontjai reprezentálják a kategóriákat, a legfelső, legáltalánosabb kategóriáktól kezdve az egyre kevésbé általános felé haladva, szigorú alá- és fölérendeltségben. Utóbbi azt jelenti, hogy ha két kategória példányai átfedik egymást, akkor az egyik a másik alkategóriája kell, hogy legyen. A ciklusmentesség (más néven trapézmentesség) elve azon alapul, hogy a taxonómia által adott osztályozás „kétszeres beszámítást" nem tartalmazhat. A természettudományok, például az állattan, a növénytan vagy a kémia kielégítik (legalábbis ideálisan) a ciklusmentesség követelményét, azonban a valós életben nagyon sok az ellenpélda. Olykor hasznos fa-szerkezettől eltérő taxonómiát alkalmazni, ahol egy adott kategóriát szimultán számos független ágra vághatunk oly módon, hogy minden ágból öröklődik az információ. Az ilyen „kereszt-osztályozások" két cél keverésével jönnek létre. Az egyik cél szigorúan taxonómiai: a fa minden levele – egymást páronként kizárva – együttesen kimerítik a tekintett tárgyterületet. A másik cél annak biztosítása, hogy egy kategória példányairól szóló ismereteket egy másik fa egy adott csomópontjához rendeljük hozzá.
2. Egy taxonómia építésének alapelve az, hogy a legalacsonyabb kategóriákat reprezentáló, vagyis az alkategóriával nem rendelkező (levél-)csomópontok száma minimális legyen. Ez a szabály azt garantálja, hogy a fa legalsó szintje kimerítő módon megadja a maximális számú „elemi" kategóriát. Ez a szabály egyben azt is biztosítja, hogy minden közbülső csomópont csak a minimális számú csomópont kombinációja lehet.
3. Egy taxonómia egyesíthető legyen abban az értelemben, hogy kell lennie egy legfelső vagy maximális csomópontnak, amely a „maximális kategóriát" reprezentálja. Ez a maximális kategória magába foglalja (általánosítja) a fa összes

alsóbb csomópontja által reprezentált kategóriát. Ez az elv azt jelenti, hogy két maximális csomóponttal rendelkező taxonómiának rendelkeznie kell egy extra, még magasabb szintű, e két (maximális) kategória egyesítését reprezentáló csomóponttal. Ha nincs ilyen még maximálisabb csomópont, akkor két elkülönülő, egymással versengő taxonómiánk van. A taxonómia legfelső szintű, maximális kategóriáját megcímkézhetjük egyetlen termmel, mint entitással; ezt általában „objektum", „elem", „létező" névvel látják el.

A gyakorlatban egy ontológiát jól-formált faként ábrázolni nem mindig lehet. Ennél erősebb állítás is igaz: a fenti három szabály teljesítése egyszerre általában nem realizálható. Ráadásul, egy ontológia nem egy fa, hanem fák családja, amelynek minden tagja az adott szakterületet specifikus szempontból veszi szemügyre különböző szemcsézettségű (pl. mikro-, közép- vagy makro-) szinten. Az egyetlen fával történő ábrázolás ellen szól az is, ha elvárjuk, hogy az alsóbb szintek fogalmai a felsőbb szintek fogalmaiból származnak. Ha minden ontológiát egyetlen fával ábrázolnának, mi lenne a legfelsőbb szintű kategória? Egy ilyen fa esetlegesen lehetne rendezett – és ismétlések és kihagyások lennének benne.

Az információrendszerek ontológiájának problémái

A filozófus-ontológusokat elvileg egy cél vezérli: a valóságról megszerezni az összes igazságot, miközben arra keressük a választ, hogy „mi az, ami létezik". Az információrendszereknél - ezzel ellentétben - egy ontológia egy olyan szoftverről vagy formális nyelven leírt műalkotásról szól, amelyet valamely számítógépes környezet specifikus használatára terveztek.

Egy ontológia a gyakorlati életben olykor nem más, mint amit az ügyfél megrendel: specifikus igények kielégítése specifikus környezetben, specifikus források felhasználásával.
Új információrendszerek ontológiájának készítése során a legfontosabb feladat megbirkózni az adatbázisok Bábel tornyának problémájával. Történeti, kulturális és nyelvi okokból az adat- és ismeretbázisú rendszerek építőinek különböző csoportjai saját egyéni jellemző szokásaik szerint dolgozták (és dolgozzák) ki rendszerük információ-reprezentációját: ugyanazt a fogalmat másképpen nevezik, illetve ugyanazt a szót más fogalom megnevezésére használják. Az ilyen jellegű információ mennyisége egyre nő; ennek megosztása és közös használata átfordítás nélkül lehetetlen.

Közös ontológiák kidolgozása

Korán felismerték annak szükségességét, hogy valamiféle szisztematikus módszert kellene találni a terminológiai és a fogalmi inkompatibilitás feloldására. Először konkrét esetenként próbálták ezt megoldani, majd rájöttek, hogy egy közös hivatkozási taxonómia ebben nagy segítséget nyújtana.

Később ezt ontológiának kezdték nevezni; ebben az esetben ez egy olyan szótár, amelyben a fogalmak kanonikus, hitelesnek elismert szintaxissal vannak leírva, és közösen elfogadott definícióval rendelkeznek. Ez egy lexikális és taxonómiai keretet jelent a különböző információrendszerközösségek számára az ismeretek reprezentációjára. Már csak egy lépés ehhez képest az, hogy ez olyan formális elmélet legyen, amely megfelelő axiómákkal van megtámogatva, ahol az axiómák implicit definíciókkal vagy értelmezésre vonatkozó korlátozásokkal lehetnek adottak.
A közös ontológia megalapozását adó ezen Nagy Enciklopédiának két része lenne:

- T-Box: terminológiai komponens (fogalmak, objektumok leírása, fogalomhierarchia) az adatbázis közösség számára,
- A-Box: állításokat (assertions) tartalmazó komponens, amelynek állításai a terminológiai komponens entitásairól szóló ismeretek, „róluk szóló tudásunk”.

Egy ontológia egy mérnöki műalkotás, amely egy bizonyos realitás leírását adó specifikus szójegyzék, továbbá a szójegyzékben szereplő szavakra vonatkozó explicit feltételezések halmaza. A legegyszerűbb esetben egy ontológia a tartalmazási relációt feltüntető fogalomhierarchia. Bonyolultabb esetben ehhez megfelelő axiómák is tartoznak, amelyek a fogalmak közötti relációkat, valamint a lehetséges interpretációs megszorításokat fejezik ki. A Guarino - információrendszer-ontológia legfőbb alakja, aki a FOIS (Formal Ontology and Information Systems) konferencia-sorozat elindítója - által javasolt ontológia-építés módszere egyrészt az adatbázis-kezelő rendszereknél alkalmazott módszerekből, másrészt a logikában és az analitikus filozófiában használt (például axiomatikus) módszerekből származik. A már meglévő taxonómiákból, adatbázis szótárakból indult ki, amelyeket különböző korlátozásokkal egészített ki (például terminológiai konzisztencia és hierarchikus jól-formáltság), de származtatott elemeket a nyelvi gyűjteményekből.

Azonban az ontológia-kiterjesztés akadályokba ütközik; az a valós probléma, hogy az adatbázisok integrálását megoldjuk, sajnos óriási méretű. Hasonló ez a világtörténelem közös ontológiájának kidolgozásához, ahol egy semleges és közös keretben kellett volna a már megtörtént összes történelmi tényt, jogi és politikai rendszert, törvényt, hiedelmet, erőforrást stb. leírni – ráadásul eltérő eredetű források felhasználásával.

A kiterjesztés előbbi problémáját még tetézi az elfogadás szintjének problémája. Egy ily módon kidolgozott, igen nagyszámú kifejezés hitelesnek elismert definícióját tartalmazó ontológia bármennyire semleges, és a különböző adatkezelő közösségek bármennyire is megegyeztek korábban benne, a gyakorlatban nagyarányú ellenérdekeltség, ütközés van a semlegességi korlátozás, valamint a széleskörűség és az erőteljesség követelményei között.
Egy lehetséges megoldás itt az lehet, hogy az ontológiaépítést két részfeladatra bontjuk:
1. Általános szintű ontológiára (amely több szakterületen alkalmazható), valamint
2. Szakterület-specifikus vagy regionális ontológiára, pl. orvosi, földrajzi ontológiára.

A zártvilág feltételezés problémája

Az ontológiával kapcsolatban beszélnünk kell még a jól ismert zártvilág feltételezés problémájáról is. Ez (adatbázisoknál, Prolog programoknál stb.) azon alapul, hogy feltételezzük: programunk a tárgyterület objektumairól szóló összes pozitív információt tartalmazza – amit nem tartalmaz, az tehát hamis információ. A zártvilág feltételezés nem csak azt jelenti azonban, hogy csak azok az entitások léteznek, amelyeket reprezentáltunk, hanem hogy ezek az objektumok csak olyan tulajdonságokkal rendelkeznek, amelyeket reprezentáltunk a rendszerben. Egy személyekről szóló adatbázis minden személyről csak véges számú tulajdonságot reprezentál (név, nem, születési dátum stb.), több tulajdonságról nem is beszélhetünk a rendszeren belül.

A zártvilág feltételezés alapján megfogalmazott modellek mind egyszerűbbek a valós világtól. Ha a mindig változó valós világ ontológiáját szeretnénk kidolgozni a maga „hús-vér” mivoltában, el kell vetnünk a zártvilág feltételezést (a szoftverfejlesztés sokkal nehezebb lesz). Ezek a problémák nyilvánvalóan fontosak az orvosi informatikában; pl. ha egy beteg rekordjában nincs reprezentálva a cukorbetegség, ez nem hatalmazza fel a rendszert arra, hogy kijelentse, hogy a beteg nem cukorbeteg!

Web Ontology Language (OWL)

Az OWL (Web Ontology Language) egy szemantikai jelölőnyelv, mely ontológiák weben történő publikálására és közös használatára szolgál. Az OWL nyelvet az RDF szókészletének kiterjesztéseként, a DAML+OIL[42] nyelvből kiindulva fejlesztették ki.

OWL Full / DL / Lite

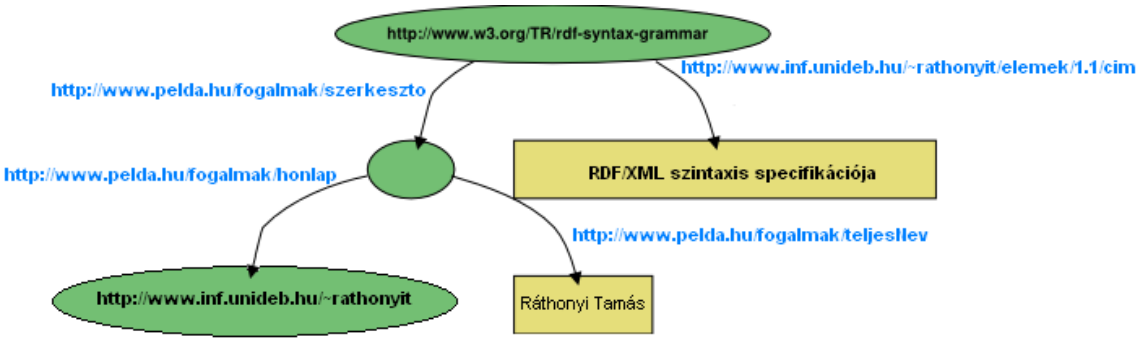
Az OWL nyelvnek két specifikus részhalmaza van. Ezek közül az OWL Lite-ot úgy tervezték meg, hogy egyrészt könnyen implementálható legyen, másrészt a nyelv olyan funkcionális részhalmazát valósítsa meg, mely segíti a felhasználót az elindulásban. Az OWL DL-t – ahol a "DL" a Description Logic (leíró logika) rövidítése – arra tervezték, hogy támogassa a meglévő leíró logikai üzleti szegmenst, és olyan nyelvi részhalmazt biztosítson, mely a következtető rendszerek szempontjából kedvező kiszámíthatósági tulajdonságokkal rendelkezik. A teljes OWL nyelv (OWL Full) feloldja az OWL DL néhány kötöttségét, amennyiben olyan opciókat biztosít, amelyek sok adatbázis-kezelő és tudásábrázoló rendszer számára előnyösek, de amelyek áthágják a leíró logikai következtető rendszerek határait. Az RDF dokumentumok leggyakrabban ezen íródnak, kivéve, ha nem kifejezetten OWL DL-re vagy Lite-ra tervezték őket.

Az OWL Full és az OWL DL az OWL nyelvi konstrukcióinak ugyanazt a halmazát támogatja. A kettő különbsége abban áll, hogy milyen korlátozások mellett használhatják ezeket a nyelvi konstrukciókat, valamint abban, hogy kihasználhatják-e az RDF alapvető tulajdonságait. Az OWL Full például lehetővé teszi az OWL és az RDF séma vegyes használatát, és ugyanúgy, mint az RDF séma, nem erőlteti az osztályok, tulajdonságok, egyedek és adatértékek szigorú szétválasztását. Az OWL DL azonban korlátozza a vegyes használatot, és megköveteli, hogy az osztályok, tulajdonságok, egyedek és adatértékek diszjunkt halmazokat alkossanak. Az OWL DL alnyelv létét az indokolja, hogy az eszközfejlesztők már korábban olyan ontológiák támogatására szolgáló nagy teljesítményű következtető rendszereket fejlesztettek ki, amelyek korlátozásai éppen az OWL DL korlátozásait igénylik.

Az OWL Lite az OWL DL alnyelve, mely az OWL nyelvi konstrukcióinak csupán egy részhalmazát támogatja. Az OWL Lite-ot elsősorban olyan eszközfejlesztőknek szánták, akik támogatni kívánják az OWL-t, de első lépésben a nyelvi konstrukcióknak csak egy viszonylag egyszerű, de alapvető részhalmazát szeretnék megcélozni. Az OWL Lite ugyanazokhoz a szemantikai korlátozásokhoz tartja magát, mint az OWL DL, s ez lehetővé teszi, hogy a következtetőgépek garantálni tudjanak bizonyos kívánatos tulajdonságokat.

Az OWL szintaxisa

Egy OWL ontológia tulajdonképpen egy RDF gráf, mely RDF tripletekből áll. Ugyanúgy, mint bármilyen RDF gráf, az OWL ontológia gráfja is sokféle szintaktikai formában leírható. Az OWL ontológia jelentését azonban kizárólag az RDF gráf határozza meg. Így tehát használhatunk bármi más RDF/XML szintaxisformát is, ha ez ugyanazokat az alapvető RDF tripleteket produkálja.



1. ábra RDF gráf

OWL és RDF szemantika

Az OWL az RDF szókészlet kiterjesztése, ezért minden RDF gráf egy OWL Full ontológiát alkot. Ebből következően az OWL által előállított RDF gráf jelentése ugyanaz, mint az RDF által előállított gráfé. Az OWL Full ontológiák ezért tetszőleges RDF tartalmat foglalhatnak magukban, amelyek kezelése is konzisztens az RDF kezelésével. Az OWL azonban további jelentést tulajdoníthat egyes RDF tripleteknek.

Mivel az OWL DL és OWL Lite kiterjeszti az RDF szókészletet, de korlátozásokat is bevezet e szókészlet használatára ezért az RDF dokumentumok általában OWL Full-ban is érvényes dokumentumok (kivéve, ha kifejezetten OWL DL vagy Lite dokumentumként kódolták őket).

Adategyesítés és a privát jogok

Az OWL-nek az a képessége, hogy ontológiai információt lehet vele kifejezni olyan egyedekről, amelyek több dokumentumból származnak, szigorúan szabályozott módon támogatja az adatok összekapcsolását. A nyelv beépített szemantikája támogatja az összekapcsolható információk logikai feldolgozását is, s ez nem várt eredményekhez vezethet.

Ha egy olyan tulajdonságról, mint például a "TAJszáma", azt jelentjük ki, hogy fordított funkcionális tulajdonság, akkor a két különböző egyedet (például ugyanazon TAJ-szám két különböző néven szereplő birtokosát) azonosnak lehet tekinteni azon az alapon, hogy ezen tulajdonságuk értéke azonos. Amikor az egyedekről ilyen módon meg lehet állapítani, hogy azonosak, akkor a velük kapcsolatos, különböző forrásokból származó információk egyesíthetők. Ez az adategyesítési funkció tehát arra is használható, hogy olyan tényeket is megállapíthassunk, amelyek egyik forrásban sincsenek közvetlenül ábrázolva.

A szemantikus webnek az a képessége, hogy több forrásból tudunk vele információkat egyesíteni, egy szükséges és rendkívül gyümölcsöző tulajdonság, mely sok alkalmazásnál jól kihasználható. Azonban az a lehetőség, hogy különböző forrásokból adatokat lehet integrálni, az OWL következtető képességeivel kombinálva magában hordja a visszaélés lehetőségét is.

Az OWL felhasználóinak ébereknek kell tehát lenniük, ha meg akarják őrizni a privát szférájuk integritását.

Osztályok

Az osztályok olyan absztrakciós mechanizmusok, amelyek segítségével csoportosíthatjuk az azonos tulajdonságú erőforrásokat. Ugyanúgy, mint az RDF-nél, az OWL-nél is egy osztályhoz egyedek halmazát asszociáljuk, s ezt az osztály kiterjedésének nevezzük. Az osztály kiterjedését alkotó egyedeket más szavakkal az osztály példányainak, eseteinek, vagy előfordulásainak is hívjuk. Az osztály jelentése intenzionális (azaz absztrakt fogalmat takar), amely kapcsolatos ugyan, de nem azonos az osztály kiterjedésével (ami extenzionális, tehát konkrét jelentésű). Így tehát két osztálynak lehet ugyanaz a kiterjedése, mégis különböző osztályokat alkothatnak.

Az OWL Lite-ban és az OWL DL-ben egy egyed nem lehet egyidejűleg osztály is: az egyedek és osztályok itt diszjunkt fogalmi kört alkotnak. Ezzel szemben az OWL Full az RDF sémáéhoz hasonló szabadságot biztosít: egy osztály felléphet egy másik (meta)osztály egyedeként is.

Az OWL osztályokat "osztályleírások" segítségével definiáljuk, amelyek "osztályaxiómákká" kombinálhatók.

Osztályleírások

Az osztályleírások alatt az osztályaxiómák alapvető építőköveit értjük. Egy osztályleírás az osztályt a nevének, vagy a kiterjedésének a megadásával írja le. Az előbbi esetben nevesített, az utóbbi esetben névtelen osztályról beszélünk.

Az OWL hatféle osztályleírást különböztet meg:

1. Osztályazonosító (egy URI hivatkozás)
2. Felsorolás (azon egyedeknek a teljes felsorolása, amelyek az osztályt alkotják)
3. Tulajdonságkorlátozás

4. Metszet (két vagy több osztály kiterjedésének a halmazmetszete)
5. Unió (két vagy több osztályleírás halmazuniója)
6. Komplemens (egy osztályleírással megadott osztály egyedeinek komplementer halmaza)

Az első típus különleges abban az értelemben, hogy az osztályt pusztán az osztály nevével írja le (amit szintaktikailag egy URI hivatkozás ábrázol). A többi öt osztályleírás típus névtelen osztályt ír le, mégpedig oly módon, hogy korlátozásokat fogalmaz meg az osztály kiterjedésére (azaz a lehetséges egyedeire) vonatkozóan.

A 2. típus olyan osztályt ír le, amelyik pontosan és kizárólag a felsorolt egyedekből áll. A 3. típus olyan osztályt határoz meg, amelynek az egyedei kielégítenek egy bizonyos tulajdonságkorlátozást. A 4., az 5. és a 6. típus olyan osztályt specifikál, mely más osztályleírások Boole-algebrai kombinációjának felel meg. (A metszet, az unió és a komplemens rendre az ÉS, VAGY, illetve a NEM operátornak felel meg.) Az utolsó négy osztályleírás egymásba skatulyázható, és így elvileg tetszőlegesen bonyolult osztályleírást alkothatnak. A gyakorlatban azonban az egymásba skatulyázható szintek számát általában korlátozzák.

Az owl:Class az rdfs:Class alosztályaként van definiálva. Annak az oka, hogy egy külön "OWL Class" konstrukciót használunk, az OWL DL és az OWL Lite korlátaiban keresendő. E korlátok miatt nem minden RDFS osztály legális OWL DL osztály is. Az OWL Full-ban ezek a korlátozások nem érvényesek, ezért ott az owl:Class és az rdfs:Class egyenértékű.

A másik öt osztályleírási forma RDF tripletek (gráf-elemek) halmazából áll, ahol egy üres csomópont az éppen definiált osztályt ábrázolja. Ennek az üres csomópontnak van egy rdf:type tulajdonsága, amelynek értéke az owl:Class.

Ha egy RDF azonosítóval látjuk el a felsorolás, a metszet, az unió vagy a komplemens típusú osztályleírást, akkor ezt már nem osztályleírásnak, hanem egy komplett osztályt definiáló osztályaxiómának tekintjük

Két OWL osztályazonosítót előre definiál a nyelv; nevezetesen az owl:Thing és az owl:Nothing azonosítót. Az owl:Thing osztály kiterjedése az összes egyedek halmaza, míg az

owl:Nothing osztály kiterjedése az üres halmaz. Ebből az következik, hogy minden osztály az owl:Thing alosztálya, valamint az, hogy az owl:Nothing minden más osztálynak alosztálya.

Felsorolás

A "Felsorolás" típusú osztályleírást az owl:oneOf tulajdonság segítségével adjuk meg. Ennek a beépített OWL tulajdonságnak az értékei azon egyedek listája, amelyek az osztályt alkotják. Ez lehetővé teszi, hogy egy osztályt definiálhassunk az egyedeinek a teljes felsorolásával. Az owl:oneOf segítségével leírt osztály kiterjedése pontosan a felsorolt egyedeket tartalmazza; sem többet, sem kevesebbet. Az egyedek listáját tipikusan az rdf:parseType="Collection" RDF konstrukcióval ábrázoljuk, mely kényelmes rövidítési lehetőségeket biztosít a listaelemek leírásához. Ez az elem az OWL Lite-ban nem használható.

Tulajdonságkorlátozás

A tulajdonságkorlátozás az osztályleírás speciális formája, mely egy névtelen osztályt ír le, nevezetesen: mindazon egyedek osztályát, amelyek megfelelnek az adott korlátozásnak. Az OWL a tulajdonságkorlátozások két típusát különbözteti meg: az értékkorlátozást és a kardinalitáskorlátozást.

Az értékkorlátozás valamely tulajdonság értéktartományát határozza meg az éppen leírt osztályra érvényes hatállyal. Például ennek segítségével hivatkozhatnánk azokra az egyedekre, amelyeknél a "szomszédosMegyéje" tulajdonság értéke valamilyen "DunántúliMegye", majd pedig használhatnánk ezt a hivatkozást egy osztályaxiómán belül (akár még magán a "DunántúliMegye" osztályaxiómáján belül is).

A kardinalitáskorlátozás segítségével azt határozhatjuk meg az adott osztályleírás kontextusán belül (azaz lokális érvénnyel), hogy egy tulajdonság hány különböző értéket vehet fel. Például: megadhatnánk azt, hogy a Futballcsapat nevű osztály "játékosa" tulajdonságának 11 értéke lehet (pl. a 11 játékos neve). Egy kosárlabdacsapat esetén ugyanez a tulajdonság csak 5 értéket vehetne fel.

Tulajdonságkorlátozásokat alkalmazhatunk adattípus-tulajdonságokra (amelyek értéke literál típusú) valamint objektumtulajdonságokra (amelyek értéke egyed típusú).

Metszet, unió és komplemens

Az ebben a szekcióban tárgyalt három osztályleírás-típus azokat a fejlettebb osztálykonstruktorokat reprezentálja, amelyeket a leíró logikában használnak. Ezeket úgy is tekinthetjük, mint osztályokra alkalmazott ÉS, VAGY és NEM műveleteket. Ez a három művelet a halmazoperátorok szokásos nevét viseli: metszet, unió és komplemens. Ezek a nyelvi konstrukciók abban hasonlítanak egymásra, hogy egymásba ágyazott osztályleírásokat tartalmaznak: vagy egyet (mint a komplemens esetén), vagy pedig többet (mint az unió és a metszet esetén).

Osztályaxiomák

Az osztályleírások alkotják azokat az építőköveket, amelyekből osztályaxiómák segítségével osztályokat definiálhatunk. Az osztályaxiómák legegyszerűbb formája az első típusú osztályleírás, mely csupán az osztály létét és azonosító nevét deklarálja az owl:Class konstruktor segítségével.

```
<owl:Class rdf:ID="Human"/>
```

Ez egy teljesen szabályos OWL mondat, de nem sokat árul el a Human osztályról. Az osztályaxiómák általában további komponenseket is tartalmaznak, amelyek megadják egy osztály szükséges és/vagy elégséges jellemzőit. Az OWL három olyan nyelvi konstrukciót tartalmaz, amelyekkel az osztályleírások osztályaxiómákká kombinálhatók:

- Az rdfs:subClassOf segítségével az jelenthető ki, hogy egy osztályleírás kiterjedése egy másik osztályleírás kiterjedésének a részhalmaza.
- Az owl:equivalentClass útján az mondható ki, hogy egy osztályleírás kiterjedése azonos egy másik osztályleírás kiterjedésével.
- Az owl:disjointWith használatával az deklarálható, hogy egy osztályleírás kiterjedésében nincs egyetlen olyan egyed sem, mely egy másik megadott osztályleírás kiterjedésében is szerepel (azaz nincsenek közös egyedeik).

Szintaktikailag ezek a nyelvi konstrukciók olyan tulajdonságok, amelyeknek az érvényességi köre és az értéktartománya is osztályleírás. Az OWL megenged olyan

osztályaxiómákat is, amelyekben a felsorolás típusú és a halmazoperátoros típusú osztályleírásoknak nevet adunk.

Tulajdonságok

Az OWL a tulajdonságok két kategóriáját különbözteti meg, amelyeket az ontológiafejlesztő használhat:

- Objektumtulajdonságok, amelyek egyedeket kapcsolnak egyedekhez.
- Adattípus-tulajdonságok, amelyek egyedeket kapcsolnak adatértékekhez.

Egy objektumtulajdonságot az owl:ObjectProperty nevű beépített OWL osztály tagjaként, egy adattípus-tulajdonságot pedig az owl:DatatypeProperty tagjaként definiálhatunk. Mind az owl:ObjectProperty, mind az owl:DatatypeProperty az rdf:Property nevű RDF osztály tagja.

Az OWL Full-ban az objektumtulajdonságok és az adattípus-tulajdonságok nem alkotnak diszjunkt halmazokat. Mivel itt az adatértékek egyedként kezelhetők, az adattípus-tulajdonság lényegében az objektumtulajdonság alosztályaként funkcionál. Az OWL Full-ban az owl:ObjectProperty gyakorlatilag egyenértékű az rdf:Property-vel.

Egy tulajdonságaxióma egy tulajdonság jellemzőit definiálja. A tulajdonságaxióma a legegyszerűbb formájában csupán a tulajdonság létét definiálja. Például:

```
<owl:ObjectProperty rdf:ID="hasParent"/>
```

A tulajdonságaxiómák azonban többnyire egyéb jellemzőket is megadnak a tulajdonságok leírásánál. Az OWL az alábbi konstruktorokat nyújtja a tulajdonságaxiómák definiálásához:

- RDF séma konstrukció: rdfs:subPropertyOf, rdfs:domain és rdfs:range
- Más tulajdonságokhoz való viszony: owl:equivalentProperty és owl:inverseOf
- Globális kardinalitáskorlátozás: owl:FunctionalProperty és owl:InverseFunctionalProperty
- Logikai tulajdonságjellemzők: owl:SymmetricProperty és owl:TransitiveProperty

2. fejezet: Szolgáltatásorientált architektúrák

Bevezetés

Napjainkban mind az üzleti, mind az informatikai világ részéről egyre nagyobb érdeklődés övezi a szolgáltatásorientált technológiát, annak ellenére, hogy nem új fogalom. Bár számos előnye és jelentősége ellenére az így megvalósított rendszerek száma nem a szakértők által megjósolt mértékben növekszik, mégis sokan úgy tartják a jövő egyik meghatározó technológiai tényezőjével állunk szemben.

Egyes nézetek szerint a SOA egy lehetőség, mellyel az üzleti élet és az információ technológia sokkal hatékonyabb módon összehangolható. Míg mások szerint a SOA maga a híd, mely egy olyan szimbolikus és szinergetikus kapcsolatot alakít ki a kettő között, mely minden eddig megálmodottnál rugalmasabb és robosztusabb lehet.

Ebben a fejezetben bemutatom a SOA kialakulásának körülményeit, ismertetem a kialakult nézőpontokat, eloszlatom a kialakult legendákat, téves hiedelmeket és értelmezéseket, majd megpróbálom pontosan definiálni mint fogalmat és a webszolgáltatásokat mint SOA implementációs technológiát megismertetni, végezetül egy rövid áttekintést adok a webszolgáltatásokról mint SOA implementációs technológiáról.

2. ábra „Ahol az információ technológia és az üzleti igény találkozik” – A SOA célja az IT és vállalatok közötti szakadék áthidalása

Történeti háttér

Üzleti élet

A 90-es évek végére a web technológiáknak köszönhetően az üzleti világ eljutott arra a szintre, hogy adatbázisok, email üzenetek, elektronikus boltok, felhasználói fórumok és multimédia tartalmak váltak néhány kattintással szinte mindenki számára elérhetővé.

Először egyre több nagy-, majd később kisvállalkozás jelent meg az Interneten és ezzel párhuzamosan automatizálta üzleti folyamatait. Ha a vállalatok versenyképesek akartak maradni, akkor egyre újabb és újabb szolgáltatásokat, funkciókat kellett biztosítaniuk az ügyfelek és partnerek számára, ami azzal a következménnyel járt, hogy új szoftver- illetve hardverelemeket kellett bekapcsolniuk már meglévő, működő rendszereikbe. Ennek eredményeképpen a vállalati informatikai infrastruktúrájuk (hardver és szoftver) szinte kezelhetetlen méretűvé kezdett válni, így egyre nehezebb és költségesebb lett az üzemeltetés, karbantartás, továbbfejlesztés, ezért az újonnan fellépő piaci igények gyors és színvonalas kielégítése sok esetben lehetetlenné vált. Ezen körülmények hosszú távon piacvesztéshez, vagy bukáshoz is vezethettek.

Informatikai világ

Informatikai oldalról tekintve a cégek újabbnál újabb technológiákkal, módszerekkel próbáltak egyre rugalmasabb megoldásokat kínálni a kiélezett piaci versenyben élre törni kívánó vállalatok igényeinek kielégítésére.

A rendszerfejlesztési modellek fejlődése során mindig központi kérdés volt a programok komplexitásának kezelése. A fejlesztők felismerték, hogy az egyre összetettebb és bonyolultabb igényeket kielégítő alkalmazások készítéséhez szükség van az absztrakciós szint növelésére. Első lépések voltak a strukturált programozás és a moduláris programtervezés,

majd ezek elterjedése után megjelent a kód újrafelhasználhatóságot támogató objektumorientált paradigma. Ez lehetőséget teremtett az implementációk elrejtésére és jól definiált interfészen keresztül az objektum állapotának és viselkedésének manipulálására. Ennek a továbbfejlesztése a komponens alapú szoftverfejlesztés, mely nemcsak a kód, hanem a funkcionalitás újrafelhasználhatóságát is lehetővé teszi. Ezek technológiák azonban nem voltak alkalmasak a heterogén környezet, az elosztott rendszerek működése, a hardver eszközök diverzitása és az Internet okozta akadályok leküzdésére.

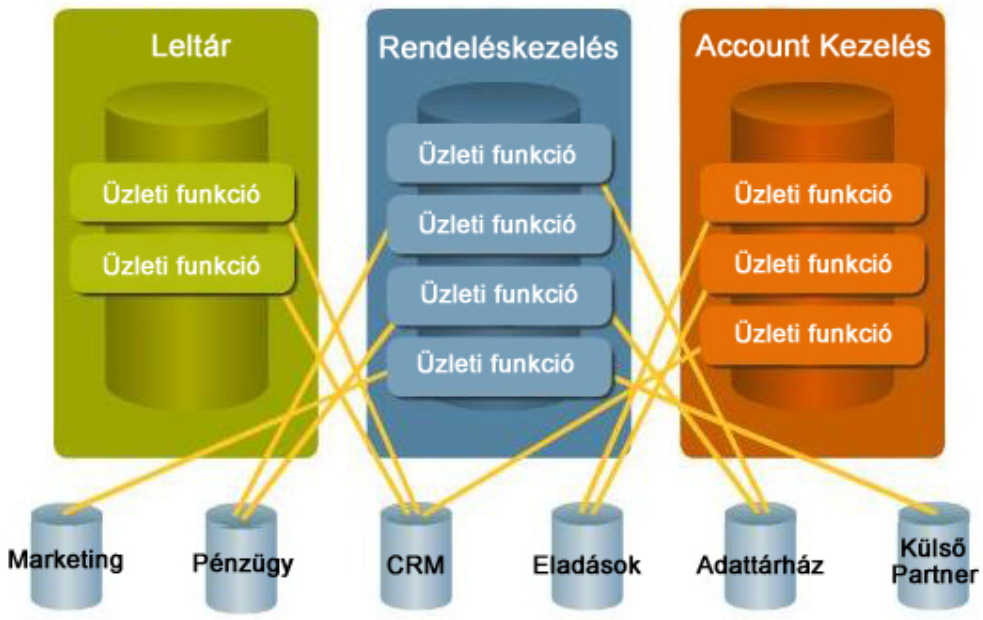
Úgy tűnik, a várt áttörést az üzleti élet fentebb említett problémáinak megoldására a webszolgáltatások (Web Services[4]) megjelenésével újra előtérbe kerülő szolgáltatásorientált technológiák és a segítségükkel létrehozható szolgáltatásorientált rendszerek jelentik. A szolgáltatásorientált technológiák általánosan használható, magas szintű kommunikációs és kapcsolati protokollt definiálnak, melyek segítségével különböző rendszerkomponensek, erőforrások természetüktől és funkciójuktól függetlenül összekapcsolhatók.

Két okra vezethető vissza, hogy alkalmazása még nem terjedt el olyan széles körben, mint várták. Az egyik, hogy a vállalatok – kudarcaikból okulva – csak végszükség esetén mernek meglévő rendszereikhez hozzányúlni, a teljes technológiai átalakítás költségét pedig a kecsegtető nyereségek ellenére is túl magasnak tartják. A másik ok a jelenleg rendelkezésre álló technológiák kiforratlanságában, hiányosságaiban rejlik.

Mindezek ellenére a következő alfejezetben található két ábra alapján látható, hogy egy szolgáltatásorientált architektúrával tervezett vállalati modell mennyivel áttekinthetőbb és mennyivel több problémára kínál megoldást.

Napjaink architektúrái, avagy az alkalmazás-központú architektúra

A mai vállalatok IT architektúrájára gyakran alkalmazások egy kollekciójaként tekinthetünk. A szoftverrendszerek tervezése, fejlesztése, üzemeltetése és karbantartása ezen alkalmazások körül forog.



3. ábra Vállalati modell „SOA nélkül”

Ez a megközelítés ihlette a vállalati architektúrákon belül az úgynevezett „elkülönített tárolók” készítését, melyek végül drága és rugalmatlan IT rendszereket eredményeztek. Mint a harmadik ábrán látható, mindegyik alkalmazás egyetlen speciális probléma megoldására lett készítve (mint például accountkezelés, rendeléskezelés), saját adattárolókkal rendelkezik és ténylegesen csak egyetlen szűk felhasználói réteg használja. Ezekből nyilvánvalóvá válik, hogy az üzleti élet folyamatainak csak egy részhalmazát implementálják, a teljes adatmennyiségnek csak egy részét használják és állítják elő, mindezt anélkül, hogy törődnének a vállalat más üzleti folyamataival. A tárolókat általában két nagy csoportba szokták osztani: „Islands of data” és „Islands of automation”.

Islands of data

Mindegyik Island Of Data (IOD) rendelkezik saját kialakított tudással és definícióval a vállalat objektumairól. Nagyon gyakori, hogy két IOD tartalmazza ugyanazt a fogalmat, de mégis más jelentést társít hozzá. Például, míg az egyik alkalmazás számára az „ár” a nettó árat jelenti, addig egy másik számára jelenthet olyan árat, melybe már bele van foglalva az adó. Még ha egy objektumnak ugyanaz is a jelentése, mint például a „postai címnek”, akkor is előfordulhat, hogy egyik esetben egy sorként van kezelve, míg másik esetben lebontva utca,

város, irányítószám, ország részekre. Mindkét eset szemantikus ellentmondást eredményezhet az alkalmazások között.

Előfordulhat olyan eset, hogy két különböző tároló által tartalmazott információk között átfedés alakul ki. Például az egészségügyi irányítással foglalkozó alkalmazások kezelik a biztosítottak demográfiai adatait, míg ezzel párhuzamosan az ügyfélkapcsolatokat nyilvántartó rendszer tárolja mind a biztosítottak címét, mind a demográfiai adatait. Ez a kettősség hosszútávon integritási és konzisztencia problémákhoz vezethet.

Egyik sem tudna a vállalat adatállományáról teljes képet nyújtani, ha szükség lenne rá. Például egy lakásépítési hitellel foglalkozó alkalmazás nem tud információt szolgáltatni arról, hogy a kölcsönt felvevőnek milyen más jellegű adósságai vannak a cégnél. Látható, ahhoz, hogy egy egységes képet kapjunk a cégek által használt adatokról, szükség van a különböző adatforrások adatainak integrálására.

Islands of automation

Minden egyes Island Of Automation (IOA) a vállalaton belül tevékenységek egy limitált halmazára koncentrál. Például az egészségügyi ellátáskérést kezelő program csak az ellátáskéréssel foglalkozik, anélkül, hogy tudná ezen tevékenység szerepét vagy helyét az egész üzleti folyamatban. Ez arra kényszeríti a felhasználókat, hogy munkájuk és hatékonyságuk növeléséhez, úgynevezett „alkalmazásugrásokat” (application hop) hajtsanak végre.

Mint az IOD-nál az adatok, úgy az IOA-nál a folyamatok között is könnyen előfordulhat duplikáció és átfedés. Ebben az esetben gondoskodni kell az alkalmazások szinkronizációjáról, az üzleti folyamatok és szabályok konzisztenciájának megőrzéséről.

Összegzés

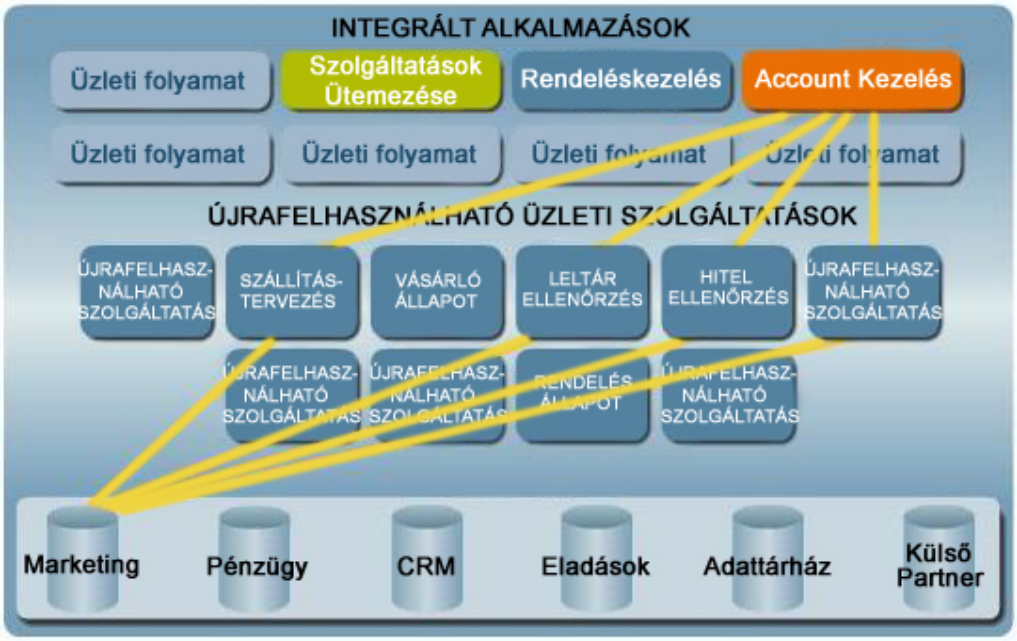
Az egész alkalmazás szintjén az Islands of data és Islands of automation hatásai szinte láthatatlanok. De a vállalkozás szintjén észrevehető problémákat okozhatnak:

- nincs szabványosított módja az adatok hozzáféréséhez, ami a hatékonyság leromlását és plusz költséget eredményezhet
- előfordulhat, hogy a különböző „tartályokban” redundánsan ugyanazt az üzleti logikát megvalósító szolgáltatások vagy adatok vannak tárolva, ami karbantartási és konzisztencia problémákat eredményezhet

- nem valósul meg az egyes folyamatok integrációja, az egyes alkalmazások a vállalati funkciók csak egy korlátozott részhalmazát biztosítják, így az üzleti és információ technológiai célok nem biztos, hogy mindig szinkronba hozhatóak
- a karbantartás és a továbbfejlesztés nehéz

Sok vállalati projektben ezen problémák megoldására próbálták bevezetni az Enterprise Application Integrationt (EAI) és az Enterprise Information Integrationt (EII). Ezek a tevékenységek világunkban az IT cégek költségvetésének jelentős részét felemésztik.

A fő ok, amiért ezen integrációs kezdeményezések drágák, összetettettek és munkaigényesek, hogy megpróbálják az alkalmazásokban mind az adatokat, mind a folyamatokat egy egységként kezelni, de ezeket soha nem úgy tervezték, hogy együtt legyenek kezelve.



4. ábra Előző vállalati modell „SOA-val”

Mi is az a SOA?

Mint fentebb említettem, ezek az alkalmazás-központú architektúrák nem hozták meg a várt rugalmasságot és sikert. A szakértőknek így új válasz után kellett kutatniuk, mely végül így hangzott: szolgáltatásorientált rendszerek. Ahhoz, hogy jól működő szolgáltatásorientált rendszereket tudjunk létrehozni, szükséges bizonyos elvek követése, szabályok betartása, melyeket a szolgáltatásorientált architektúra (rövidítve: SOA) definiál. Ezen architektúra az alkalmazások tervezéséhez rugalmas, gazdaságos funkcionalitás újrafelhasználhatóságot kínál lazán csatolt üzletközpontú szolgáltatások formájában.

Az elkövetkezendő néhány alfejezetben azt a kérdéskört járom körül, hogy vajon mit is takar pontosan ez a sokak által már-már mitikus, megváltói magasságokba emelt fogalom.

A megválaszolhatatlan kérdés

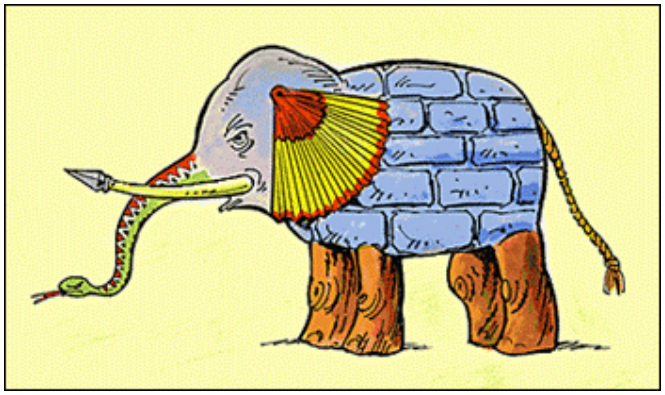
Maga a fogalom nem számít újnak, már az OMG[9] CORBA[8] technológiájának megjelenése óta ismeretes, de kialakulásakor nem számított technológiai áttörésnek. A benne rejlő lehetőségeket a webszolgáltatások megjelenésekor ismerték fel a szakemberek. Bár a WS volt a katalizátor az újra előtérbe kerüléséhez, mégis a definiálása előtt gyakran elhangzanak a következő szavak: „Nincs szükség webszolgáltatásokra a SOA kialakításához". Ez a megállapítás teljesen helytálló, mert nem ez az egyetlen lehetőség, hogy SOA alapokon jól működő rendszert hozzunk létre. (Egy másik lehetőség lehet akár a Jini technológia[10], melyet a Sun Microsystems fejlesztett ki 1999-ben) Ámbár ezt a mondatot tipikusan követi a következő megállapítás: „..de a webszolgáltatások használata a SOA kialakításához egy egészen kiváló ötlet…".[7] Ezekből látható, hogy ezt a három betűs mozaikszót nem olyan könnyű egzakt, teljesen körülhatárolt módon definiálni, mint első ránézésre tűnik.

Martin Fowler[34] a híres szerző és nemzetközileg elismert architektúra szakértő Blikijében (blog és wiki kombinációja) a következőképpen vélekedik: „Egyszer megkérdezték tőlem, hogy mit gondolok a SOA-ról. Ez az a kérdés, amire szinte lehetetlen válaszolni, mert nagyon sok dolgot jelenthet a különböző emberek számára.”[35]

Ezt a sokszínűséget gyakran szokták szemléltetni egy szerintem nagyon érdekes és izgalmas példával, amikor a SOA-t John Godfrey Saxe költeményében szereplő elefánthoz hasonlítják.

A történet alapja, hogy hat vak indiai férfi találkozik egy elefánttal. Mindegyikük a saját tapasztalataira hagyatkozva megpróbálja leírni, hogy mivel is állhatnak szemben:

- a férfi, aki megérinti az ormányt, úgy gondolja, kígyó
- a férfi, aki megérinti az agyarat, úgy hiszi, lándzsa
- a fület megérintő férfi úgy véli, legyező
- az elefánt oldalát megérintő férfi úgy gondolja fal
- a férfi, aki a farkát érinti, kötélnek véli
- a lábakat érintő férfi fatörzsnek hiszi.



5. ábra Saxe elefántja

Látható, hogy egyenként mindegyikük véleménye helytálló, de valójában az összképet tekintve tévednek. A párhuzamot a költemény és a való élet, esetünkben a szolgáltatásorientált architektúrák világa között az szolgáltatja, hogy ha megkérdeznénk a

rendszerfejlesztési életciklus különböző szakaszaiban a kifejlesztendő rendszerrel kapcsolatba kerülő személyeket, hogy mit gondolnak mi is az a SOA, akkor valószínűleg nagyon sokféle és néha talán egymásnak ellentmondó véleményt kapnánk válaszul. A 6. ábrán látható táblázatban röviden összefoglaltam a különböző informatikai és üzleti nézőpontokat képviselő személyek „SOA definícióit. Ezek az állítások teljes mértékben igazak abból a szemszögből, amelyben felállították őket, de a teljes képet magáról a fogalomról sem egyedül, sem együtt nem tudják kialakítani.

Ezen nézőpont szerint	a SOA nem más, mint:
Üzleti igazgató és üzleti elemző	szolgáltatások egy halmaza, melyek az IT lehetőségeit alkotják; olyan üzleti megoldások készíthetők a segítségével, melyek az ügyfelek és partnerek számára megfelelőek
Vállalati Architekt	architektúrális elvek halmaza, melyek támogatják a modularitást, a bezárást, laza csatolást, fogalmak elkülönítését, újrahasznosítást a szoftvertervezés során
Projekt Manager	fejlesztési megközelítés, mely masszív párhuzamos fejlesztést támogat
Tesztelő és minőségbiztosítási tervező	az egész rendszer tesztelésének és ellenőrzésének egyszerűsítését segítő módszer
Szoftverfejlesztő	egy programozási modell tele szabványokkal, eszközökkel és technológiákkal

6. ábra A SOA különböző értelmezései

Legendák és tények

Ehhez a „fogalomzavarhoz” még számtalan téves hipotézis és legenda társul. Úgy gondolom, hogy ezeket mindenképpen fontos tisztázni.

Legenda: SOA nem más, mint Web Services

Tény: Az egyik leginkább elterjedt tévhit, hogy bármi, amit Web Service technológiával valósítanak meg, az már SOA. Ezen elmélet alapja a HTTP-n keresztül elérhető metódusok félreértelmezhetőre sikeredett névválasztása, azaz a Web Szolgáltatás kifejezés. Több híres, a témában megjelent könyv is azt sugallja, hogy a SOA nem más WS, bár külön fejezetet fordítanak arra, hogy nem az egyetlen technológia, mellyel megvalósítható. Valójában egy webszolgáltatás köteg nem sokban különbözik az RPC-től (Remote Procedure Call).

Legenda: SOA és EAI egy és ugyanaz

Tény: Másik széles körben elterjedt definíció: „speciális EAI megvalósítás, melyben sikerült a költséget csökkenteni". Az EAI egy próbálkozás az üzleti integráció problémáinak megoldására az alkalmazások összekapcsolásával és adatok átvitelével közöttük egy félig szabványosított úton. Egyes nézetek szerint több téren is alulmúlta a várakozásokat:

- Adatcentrikus és nem folyamat
- Nem tud megfelelően lépést tartani az üzleti folyamatok változásaival
- Az így megvalósított rendszerek nagyon komplexek, speciális szakképzettséget igényelnek és nagyon drága a karbantartásuk.

Szokták használni a SOI kifejezést is (Service Oriented Integration), az EAI webszolgáltatásokkal történő megvalósítására, azonban ez nem más, mint egy már létező gondolkodásmód új technológiai köntösbe bújtatása.

Legenda: SOA könnyűvé teszi az újrafelhasználást

Tény: Az állítás, hogy a SOA egyszerűbbé teszi az újrafelhasználást, szintén nem helytálló, mégis nagyon sok szakirodalomban megtalálható. Mint az objektumorientált világban megtapasztalhattuk, az újrafelhasználás annál nehezebb, minél nagyobb, összetettebb komponenst szeretnénk újrafelhasználni. Minél komplexebb a komponens, annál több elemmel kerül kapcsolatba a környezetében és annál nehezebb az újrafelhasználása. Ugyanez igaz a SOA-ra is. A szolgáltatások különböző üzleti lehetőségeket ölelnek fel. Ha jól vannak megtervezve, akkor tökéletesen használhatók a funkciójuk kielégítésére és az üzleti folyamatokba is könnyen integrálhatóak. Mindezek ellenére csak nagyon ritka esetben lehetséges az újrafelhasználásuk. A SOA sokkal inkább az üzleti változások

dinamikusságáról, a gyors változtathatóságról szól, mint a szolgáltatások újbóli felhasználásáról egy más környezetben.

Legenda: mindent vagy semmit

Tény: Az újrafelhasználás fogalmát megközelíthetjük abból a nézőpontból is, hogy egy működő vállalat szeretné informatikai infrastruktúráját SOA irányába elmozdítani. Felvetődik a kérdés, hogy a fejlesztést egyben vagy lépésről lépésre kell végrehajtani. Az utóbbit szokták alkalmazni, mint evolúciós fejlesztési folyamatot.

Tegyük fel, hogy egy vállalat a teljes szoftverrendszerét le szeretné cserélni. Nyilván amíg a fejlesztés tart, nem állhat meg az üzleti élet, ezért biztosítani kell, hogy a folyamatok továbbra is megfelelően működjenek. A SOA egyik jellemzője, hogy támogatja az úgynevezett ősrendszerek (legacy systems) fokozatos üzemen kívül helyezését az új interfészeken keresztül, ezzel támogatva a vállalat zavartalan működését.

Legenda: egyszerűbb integráció

Tény: A SOA nagy hangsúlyt fektet az interfészekre, melyeken keresztül a szolgáltatások könnyen tudnak egymással kommunikálni. Ezt gyakran valamilyen szabványosított technológia segítségével teszik, például XML, WS, HTTP, melyek segítik a szolgáltatások összekapcsolását. De a szolgáltatások teljes integrációjához, hogy használhatóak legyenek az üzleti folyamatok megvalósításához, modelleznünk kell a közöttük lévő kapcsolatokat. Ez a legidőigényesebb fejlesztési lépés, és a SOA nem rendelkezik semmilyen eszközzel a könnyebbé tételére.

Legendáktól valóságig…

Martin Fowler szerint a SOA egy olyan kifejezés, mely mögött nagyon sok olyan jó ötlet található, melyeket ki kellene emelni, és saját névvel függetlenül kellene létezniük. Mások szerint egyszerűen csak félreértelmezik a jelentést.

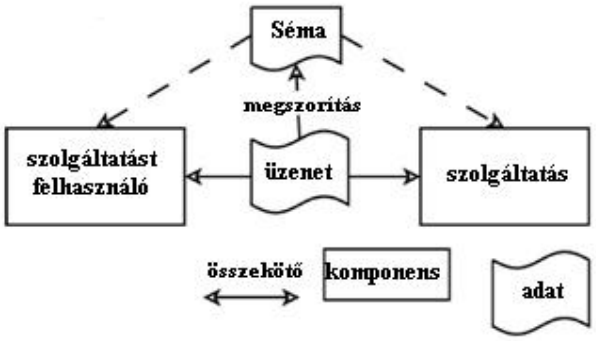
Mindezek ellenére napjainkban egyre több szolgáltatásorientált architektúra mintájára létrejövő rendszer van. Ennek oka valószínűleg, hogy egyre többen kezdik megérteni a fogalmat és felismerni a benne rejlő lehetőségeket. Két elterjedt definíciós nézőpont van: az egyik az üzleti világé (gyakran emlegetik, mint SO a SOA-ból), és az információ technológia világáé (az A a SOA-ban). Ezek közül a másodikat mutatom be a következő alfejezetben.

A válasz a kérdésre

Az előző néhány alfejezetben ismertettem a különböző kialakult nézőpontokat, megpróbáltam eloszlatni a téves hiedelmeket és definíciókat. Úgy gondolom elérkezett az idő, hogy megpróbáljam megadni a választ a kérdésre, azaz definiálni a SOA-t.

Ha nagyon egyszerűen szeretnénk megfogalmazni, akkor azt mondhatnánk ez nem más, mint egymással kommunikáló jól definiált, egységbezárt szolgáltatások kollekciója, mely szolgáltatások információcserét, feldolgozást biztosítanak, vagy más szolgáltatások koordinálását hajthatják végre.[38] Ez egy rövid, tömör, általános definíció, de szerintem ettől létezik egy jobb megközelítés is.

Mielőtt bemutatnám a másik definíciót, szeretném ismertetni az architektúrális minta fogalmát, bár erre sem létezik szabványosított változat. Ez nem más, mint megszorítások egy halmaza az architektúra elemeire vonatkozóan és kapcsolatok ezen elemek és más olyan architektúrák elemei között, melyek hasonlóak a mintához. Általában az architektúra elemek a feldolgozó komponensek, az összekötők és az adat. Egy feldolgozó komponens átalakíthatja az adat elemeket. Az összekötők egy absztrakt kommunikációs, koordinációs és együttműködési mechanizmust biztosítanak a komponensek között. A feldolgozó komponens nézőpontból az összekötő nem tesz mást, mint adatot továbbít módosítás nélkül. Habár az összekötő tartalmazhat egy összetett alrendszert, mely számos közbülső transzformációt hajthat végre az adaton.



7. ábra SOA, mint architektúrális minta

Ezt felhasználva tehát a SOA egy architektúrális minta, mely támogatja a lazán kapcsolt, üzletközpontú, hálózati szolgáltatások rugalmas együttműködését platform független módon.

Konkretizálva: az architektúra szolgáltatásokból és a szolgáltatásokat felhasználó kliensekből áll. A szolgáltatás egy olyan funkcionalitás halmaz, mely a külvilág számára egy szolgáltatás interfészen segítségével van leírva, azon keresztül érhető el. Bár egy szolgáltatás reprezentálhat tetszőleges üzleti szolgáltatást, a továbbiakban csak szoftverszolgáltatásokra szorítkozunk, feltételezve, hogy szoftverszolgáltatások jelenítik majd meg az üzleti szolgáltatásokat is. Minden szolgáltatás interfész egy szerződést definiál, mely egyértelműen megadja, hogy a szolgáltatás mit nyújt a felhasználó (kliens) számára. Az interfész megadása egy megfelelő interfészleíró segítségével történik.

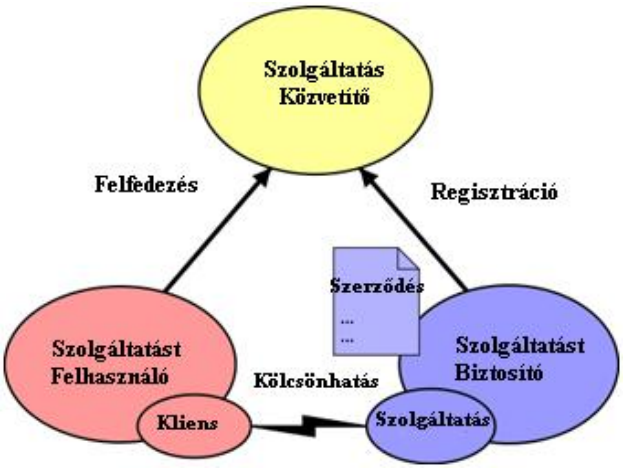
Több architektúra szakértő „Asimov: a robotika három törvénye” mintájára beszél a „SOA három törvényéről”:

1. törvény: A szolgáltatáshatárok explicitek. (A szolgáltatások egyértelműek, jól definiált és érthető, hogy mi az, ami egy adott szolgáltatás körében benne foglaltatik és mi az, ami nem. Együttműködő szolgáltatások esetében pontosan definiálható, hogy mi tartozik az egyik és mi a másik szolgáltatás hatáskörébe.)

2. törvény: A szolgáltatások autonómok, önálló entitásként működnek. (Minden szolgáltatás a külvilágtól függetlenül létezik. Működésük során nem bíznak meg a külvilágban, adataikat, belső erőforrásaikat önállóan kezelik.)

3. törvény: A szolgáltatások együttműködését a közöttük lévő szerződés határozza meg. (Az együttműködés kapcsán a szolgáltatások kizárólag a közöttük lévő „szerződés” alapján kommunikálnak, nincsenek „nem szerződött” előfeltételezéseik egymás kapcsán.)

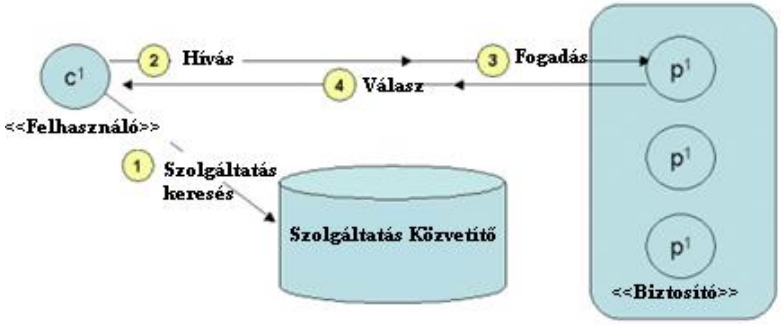
A szolgáltatásorientált architektúra alapja, hogy a kliens és szolgáltatás kapcsolata lazán csatolt. A kliens nem rendelkezik előzetes információval a szolgáltatás helyéről, valamint implementációjának részleteiről. A működés fontos eleme emiatt a felfedezés és regisztráció. Amint a 8. ábrán látható, a kliensek egy közvetítő segítségével fedezik fel a szolgáltatásokat. A közvetítő a nála regisztrált szolgáltatásokról tárol információt.



8. ábra

A szolgáltatás leíró lehetővé teszi, hogy a kliens a felfedezett szolgáltatást használni tudja, vagyis kezdeményezni tudja a kívánt funkció végrehajtását. A szolgáltatások fekete dobozként jelennek meg a rendszerben, azaz a kliens a belső implementációs részleteket nem ismeri, és a szolgáltatások állapotmentesek; a szolgáltatás a hívások sorrendjétől és végrehajtójától függetlenül mindig ugyanúgy működik.

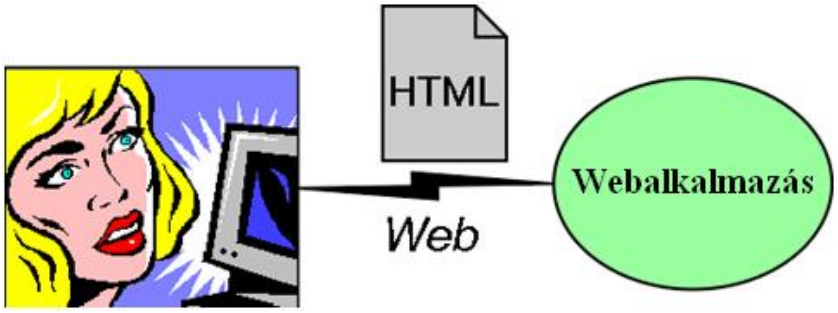
A szolgáltatásorientált architektúra talán legfontosabb jellemzője, hogy a szolgáltatások felhasználói nem személyek, hanem programok. A szolgáltatásorientált jövőképet egymással együttműködő programok alkotják. A kliensek futási időben képesek számukra hasznos szolgáltatásokat felfedezni, majd azokat megfelelő módon használni. A szolgáltatások komponensekként is funkcionálnak, több szolgáltatásból úgynevezett kompozit szolgáltatások hozhatók létre.



9. ábra

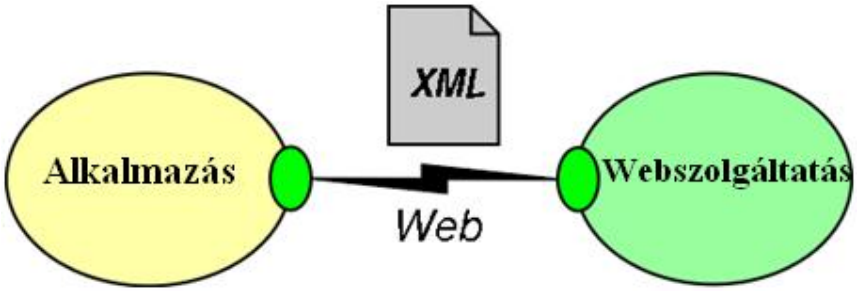
Webszolgáltatások

A webszolgáltatások (Web Services) az egyik (de nem az egyetlen) lehetséges módja SOA kialakításának a weben. A szakirodalom a WS alatt általában a WSDL/SOAP alapú szolgáltatásokat érti, de van egy általánosított nézőpont, mely az összes többi típusra is érvényesíthető.



10. ábra Hagyományos web interakció

A W3C definíciója szerint: Egy webszolgáltatás egy olyan URI által azonosított szoftverrendszer, melynek publikus interfészei és kapcsolódásai XML segítségével vannak definiálva és leírva. Más szoftverrendszerek által is elérhető a definíciója. Ezek a rendszerek kapcsolatba léphetnek a webszolgáltatással előre definiált módon XML-alapú üzeneteket szállítva Internet protokollokon keresztül.

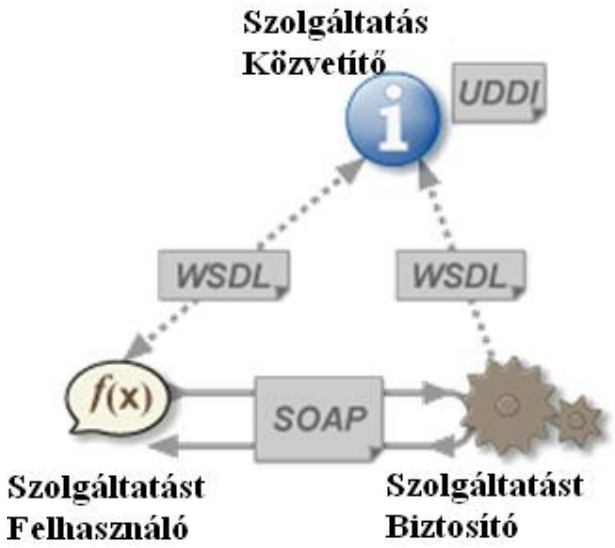


11. ábra Web Service interakció

A HTTP az Internet alapvető protokollja, míg az XML lehetővé teszi a távoli komponensek közötti adatcserét függetlenül a komponensek konkrét implementációjától. Erre a rétegre épülnek további technológiák. Amint a 12. ábra illusztrálja, minden szolgáltatás egy WSDL (Web Service Description Language[11]) nyelvű interfész leírás segítségével hirdeti képességeit a hálózaton. Ez egy XML-alapú leírás, mely megadja a szolgáltatás felhasználásához szükséges információkat a kliens számára. A közvetítő egység az **UDDI** (Universal Description, Discovery and Integration[12]) regiszter. Az itt tárolt információk között kereshetnek a kliensek. A szolgáltatásinformációk csoportosítva jelennek meg. A „White Pages" tartalmazza a szolgáltatás üzemeltető elérhetőségi adatait. A „Yellow Pages" rész tevékenység szerint tartja nyilván a szolgáltatásokat, míg a „Green Pages" a szolgáltatás felhasználásához szükséges technikai információkat tárolja. A keresés eredményeként kiválasztott célszolgáltatást a kliens a szintén XML-alapú **SOAP** (Simple Object Access Protocol[13]) protokoll segítségével érheti el és használhatja.

A webszolgáltatás technológia dokumentum-alapú kommunikációra épül. A felek XML-ben leírt dokumentumokat küldenek egymásnak üzenetek formájában. A működés alapfeltétele, hogy az XML üzeneteket minden egyes résztvevőnek fel kell tudni dolgozni, majd ezután értelmezni kell. Az üzeneteknek a megoldandó problémát és nem a probléma megoldását kell tartalmazniuk. Tehát a szolgáltatás biztosítja a szolgáltatást felhasználó számára a probléma megoldásához szükséges „szakértelmet". Az üzenetváltás célszerűen aszinkron módú, bár a gyakorlatban a szinkron XML-RPC terjedt el a legjobban.

A WS technológia állapotmentes, továbbá nem tranziens szolgáltatásokat feltételez. Ez azt jelenti, hogy a szolgáltatás nem tárolja az állapotot hívások között, azokra mindig azonosan reagál. A nem tranziens működés pedig nem teszi lehetővé szolgáltatások létrehozását, megszűntetését.



12. ábra

WSDL/SOAP Web Services

Ez napjaink legnépszerűbb SOA megvalósítási módja a weben. A WSDL egy leíró keretrendszert biztosít a webszolgáltatások számára és középpontjában elsősorban a szolgáltatáshívás áll. A SOAP olyan szabványosított üzenetstrukturálási módszert kínál, mellyel az üzenet akár több, különböző protokollon keresztül is továbbítható. Leggyakrabban a HTTP protokollal használják.

Kezdetben a SOAP csak az RPC kommunikációs stílust támogatta, a dokumentum-alapú változatát csak később mutatták be. A dokumentum-alapú SOAP üzenetek képesek bármilyen XML-alapú tartalom hordozására, és ami a legfontosabb, hogy nincsenek rákényszerítve az RPC kommunikációra. A SOA szemszögéből nézve az RPC csak újabb mesterséges függőségeket okoz a szolgáltatást használó és a szolgáltatás között, ezzel csökkentve a komponensek közötti laza csatolás lehetőségét. Ezen kívül inkább előíró, mint leíró, abban az értelemben, hogy az RPC-üzenet előír egy szolgáltatást, hogy hogyan kell a problémát megoldani, ahelyett, hogy mit kell megoldani. Az üzenetek szemantikája gyakran munkamenetfüggővé válik, így meggátolja a szolgáltatások integrációját. A SOAP 1.2-ben az RPC opcionálissá vált, ezzel lehetővé téve kihagyását a WSDL/SOAP alapú webszolgáltatásokból, habár a kutatói közösségekben még mindig több figyelmet kap.

Dokumentum alapú WSDL/SOAP webszolgáltatások már teljesen kompatibilisek a SOA-val. Általában XML sémanyelvet használják XML dokumentumok struktúrájának leírására, bár ez nem teszi lehetővé az üzenetelemek közötti szemantika modellezését. Ezen szemantika megfelelő kezelését a WSDL kiterjesztő elemek teszik lehetővé.

WSDL/SOAP Web Services jelentős számú fogalmi és architektúrális elemet mutatnak be. Ezeknek jelentős része a SOA-ban megelőzendő RPC-hez tartozik. Mindezek mellett sok esetben egy dokumentum-alapú hívás, melyet egy jól kiépített kommunikációs interfészen (HTTP protokoll és a kérés-válasz üzenetalapú minta) keresztül lehet végrehajtani, elegendő egy operációs SOA létrehozásához. Így a SOA egyszerre lehet egyszerűbb és megfelelőbb sok gyakorlati alkalmazás számára.

Representational State Transfer (REST) Web Services

Alapja a REST architektúrális minta, mely az Erőforrás és URI fogalmak köré épül. A kliensek közötti interfész a HTTP-re korlátozódik, mely biztosítja mind a transzport réteget, mind az erőforrásokon végrehajtható műveleteket. A REST WS-ek általában az XML-t alkalmazzák az üzenetek kifejezésére és az XML sémát, mint szótár definíciós mechanizmust. Mivel az infrastruktúrája kevésbé bonyolult, ezért egyszerű az ontológia-alapú sémadefiníciós mechanizmusok alkalmazása. Másik fontos újítása, hogy a web szükségleteinek kielégítésére jött létre, így a web sok olyan architektúrális tulajdonságát megkapta, melyek már sikeresnek bizonyultak.

WSDL/SOAP jelentős infrastruktúra igényével szemben a REST számára csekély mértékű kiegészítés kell a web által már biztosított infrastruktúra mellé, mely teljes mértékben megfelel a SOA megszorításainak, és könnyen alkalmazhatók vele az ontológia-alapú sémadefiníciós nyelvek.

Annak ellenére, hogy a REST webszolgáltatások nem szabványos webszolgáltatások és kevésbé széles körben ismertek, néhány esetben előnyben vannak részesítve a WSDL/SOAP megközelítéssel szemben. Például az Amazon, mind WSDL/SOAP, mind REST webszolgáltatásokat biztosít, mégis a felhasználás 85% a REST interfészen keresztül történik.

Semantic Web Services

A szemantikus webszolgáltatások modern megközelítései, mint az OWL-S és WSMF ontológiákat alkalmaznak a WSDL/SOAP alapú szolgáltatások szemantikailag gazdag leírásához. Az ilyen leírások metaadatot biztosítanak számunkra a szolgáltatásról, és alkalmazhatóak a webszolgáltatásokhoz kötődő olyan funkciók automatizálásához, mint például a felfedezés, hívás, összeállítás.

A SOA nézőpontjából az SWS definiál egy viszonylag összetett szemantikus kapcsolót, mely a szolgáltatást használó és a szolgáltatás közötti olyan kapcsolat kialakításáért felelős, mely biztosítja az üzenetek szemantikus kompatibilitását. Az összekötő elem komplex műveleteket hajthat végre úgy, hogy a szolgáltatás még mindig szemantika-mentes üzenetekkel dolgozik (legtöbb esetben SOAP-RPC).

Tehát a modern SWS megközelítések szerint nincs közvetlen szemantikailag gazdag üzenetcsere a SOA komponensek között. Például, ha adott egy webszolgáltatás RDF üzeneteken keresztül történő kommunikációhoz, akkor a modern SWS felfogás itt alkalmazhatatlan. Ha mindezek mellett a szolgáltatás nem WSDL/SOAP alapú, akkor még kevesebb az esély, hogy meglévő SWS megközelítést alkalmazzunk.

3. fejezet: Ontológiák és szolgáltatásorientált architektúrák kapcsolata

Bevezetés

Az ontológiák és SOA összekapcsolásából származó előnyök jelentőségét először a SWS területén ismerték fel. Az OWL és WSMF megközelítések egy olyan ontológia-alapú keretrendszert nyújtanak WSDL webszolgáltatások számára, mellyel automatizálhatók az olyan magas szintű feladatok, mint például a felderítés, szolgáltatáshívás, szolgáltatáskompozíció.

A kapcsolatot leggyakrabban Onto-SOA szemszögből szokták vizsgálni, mely az Ontology enabled Service-Oriented Architecture rövidítése, és egyesíti a webszolgáltatások által biztosított átjárhatóságot és az ontológiák nyújtotta gazdag szemantikus leírást. Az Onto-SOA független bármilyen ontológia nyelvtől, és csak részlegesen alapszik a webszolgáltatás technológián, ezzel biztosítva, hogy alkalmazható bármilyen olyan általános megközelítésben, melyben az ontológiákat és SOA-t szeretnék együttesen alkalmazni. Egyes nézetek szerint ez a kapcsolat kétirányú és mindkét irányból átjárható, tehát készíthető olyan ontológia, melybe valamilyen szolgáltatás van integrálva, ezzel lehetővé téve különböző kapcsolódási mechanizmusokat. Ezen elmélet igazolására fejlesztették ki a MoRe-t, ami egy RDF/S engedélyezett SOA REST webszolgáltatásokkal megvalósítva.

Az előző két fejezetben bemutattam az ontológiákat, az OWL-t, a szolgáltatásorientált architektúrákat és az ezekhez kapcsolódó fontosabb fogalmakat. Ebben a fejezetben pedig ismertetem, hogyan lehet ezeket összekapcsolni és milyen előnyökkel járhatnak a különböző megközelítések szerint.

Open SOA Ontology

Az első kérdések között vetődhet fel, hogy egyáltalán miért is van szükség arra, hogy egy szolgáltatásorientált architektúra számára ontológiát fejlesszünk ki? A válasz az, hogy

lehetőségünk legyen a SOA fogalmainak, terminológiájának és szemantikájának még pontosabb definiálására, melyeket mind az üzleti, mind az informatikai világban lehet alkalmazni. Ez elősegíti többek között a későbbi munkák szakterület-specifikus megalapozását, az üzleti élet szereplői és a fejlesztők közötti könnyebb kommunikációt és nem utolsó sorban közreműködhet egy olyan modell-vezérelt SOA implementáció elkészítésében, mely megkönnyíti a SOA adoptációját.

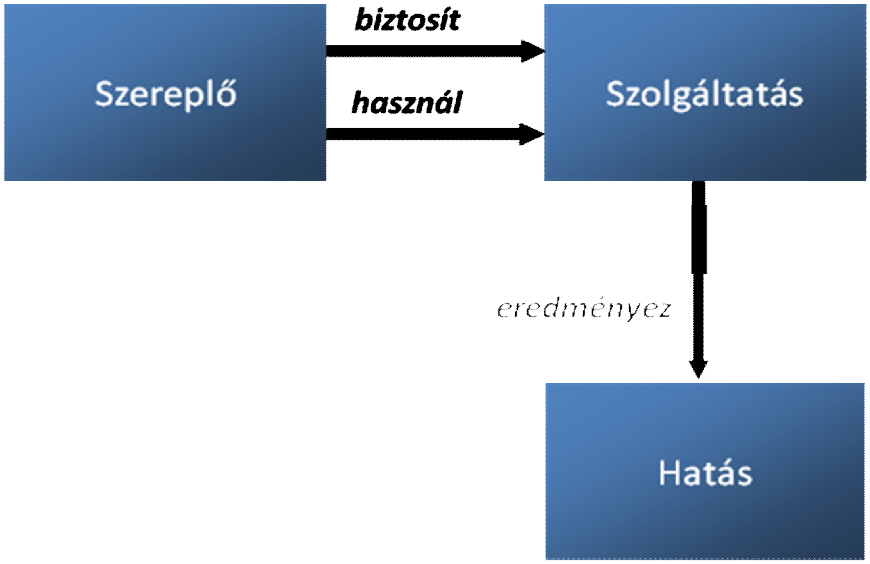
Egyik oldalról az ontológiák modellezhetik az üzleti fogalmakat, míg másik oldalról akár az architektúrális modellezésére is alkalmasak, a kettő kombinálásával pedig eljuthatunk az MDA-hoz (Model-Driven Architecture). Az MDA jelentősége abban rejlik, hogy ha az architektúra modell eléggé világos és részletes, akkor az interfész definíciók és talán egyes programblokk implementációk automatikusan generálhatók. Ennek előnye, hogy a webszolgáltatás modell eléggé világos és részletes így különböző SOA modellek fejleszthetőek ki, illetve bizonyos esetekben generálhatóak ontológia segítségével, mely ebben az esetben egy generikus keretrendszer szerepét tölti be.

Az Open SOA Ontology alapötlete az általánosítás-részletezés koncepcióban rejlik. Az egyes szakterületek fogalmait próbálja kategorizálni és is-a viszonyt felállítani közöttük.

13. ábra Egészségügy

Az ábrán balról jobbra haladva általános fogalmak felől egyre konkrétabb fogalmak felé tartunk. Ezzel együtt egyre több adattaggal is rendelkeznek. Például a Biztosító csak azonosítóval rendelkezik, míg a kórháznak van azonosítója, címe, tudjuk, hogy hány ággyal rendelkezik; a bőrgyógyászatról pedig tudunk mindent, amit egy kórházról tudhatunk, és

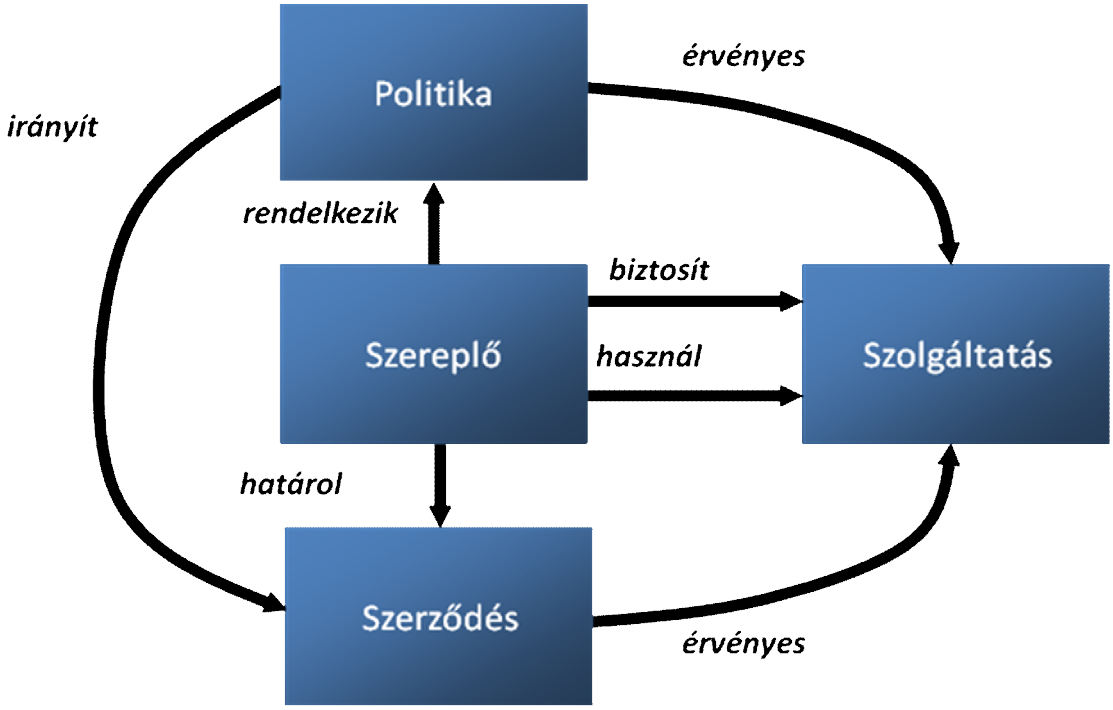
például plusz információ lehet például a részleg vezetőjének neve és azonosítója. Ez alapján arra a kérdésre, hogy mik lesznek a példányok, ebben a megközelítésben nem tudjuk és nem is szeretnénk tudni a választ, mert a különböző általánosítás-részletezés koncepciók különböző módokon definiálhatják a példányokat. Például eltérés lehet abban, hogy a Magán beteget alosztálynak vagy már példánynak tekintik-e. Szintén nem kell foglalkoznunk azzal, hogy mik az alosztályok, példányok, tulajdonságok illetve speciális információk az egyes szakterületeken. Ugyanakkor az alapvető SOA osztályok és tulajdonságok helyes definiálása fontos és különös figyelmet kell fordítani arra, hogy a szolgáltatások milyen módon cserélik ki egymás között az információkat. Az szolgáltatások által kicserélt információ nem más, mint információt leíró információt leíró információ, tehát metameta adat. Például, ha a kórházban található ágyak számáról van szó, akkor a szolgáltatásoknak azt kell tudniuk, hogy ez a kórházban található ágyak számát leíró információ és nem a kórházban található ágyak száma.



14. ábra Open SOA Ontology alaposztályai és kapcsolatok

A 14. ábrán láthatóak az alaposztályok és a közöttük lévő kapcsolatok. Az egyik legfontosabb és szinte nélkülözhetetlen osztály a Szereplő. A Szereplő lehet egy személy, egy cég vagy akár egy szoftver – valaki, aki csinál valami. A modellezésben ez inkább egy szerepkört reprezentál és nem konkrét individuumokat. Az is előfordulhat, hogy egy szolgáltatás kerül ebbe a szerepkörbe, de általánosan igaz az a megállapítás, hogy nem

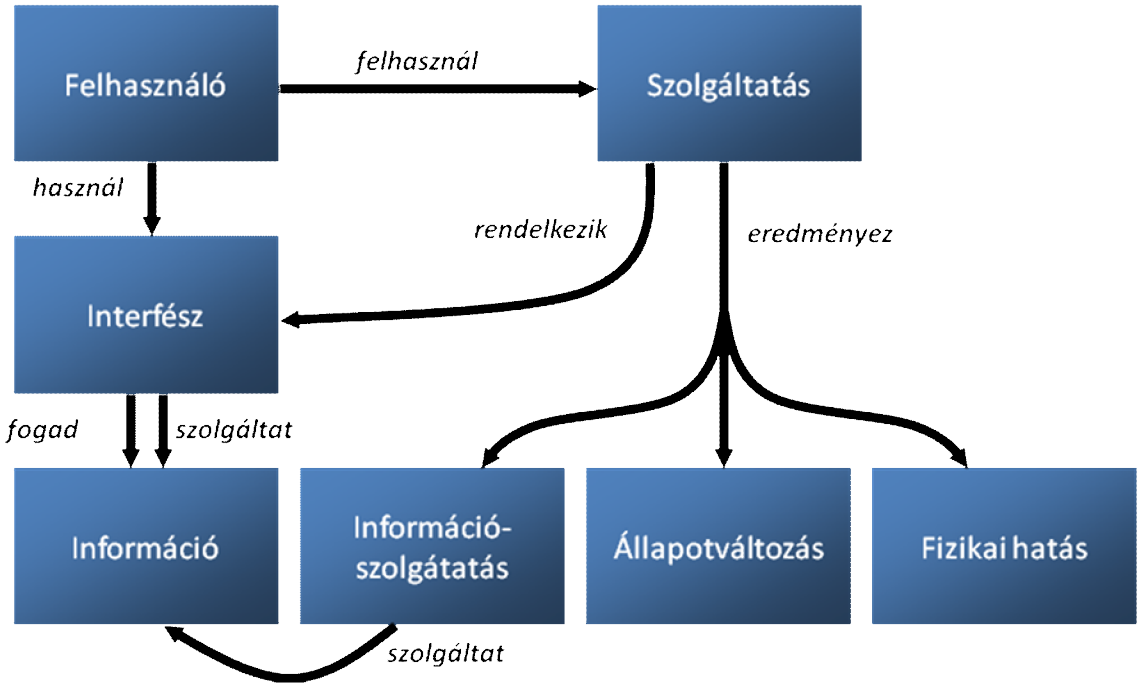
minden Szereplő Szolgáltatás és nem minden Szolgáltatás Szereplő. A Szolgáltatás egy viselkedési minta és nem egy konkrét példány, sőt az is előfordulhat, hogy különböző viselkedési minták ugyanazon Szolgáltatás különböző szolgáltatásai. A Hatást pedig három csoportba osztották: az első esetben az eredmény valamilyen információ, második esetben lehet egy bekövetkező állapotváltozás illetve bizonyos fizikai hatás.



15. ábra Szerződés és politika

Ahogy az előző fejezetben említettem, a Szerződés a SOA egyik alapfogalmai közé tartozik. Rendelkezhet meghatalmazottal és opcionális feltételekkel, melyeket egy Szereplő hozhat létre. Ellentétben a politikával, melyet nem szükséges, hogy akár egy Biztosító, akár egy Felhasználó birtokoljon, egy szerződésnek legalább kettő vagy több résztvevő között kell létrejönnie. Ebben a megközelítésben figyelmen kívül hagyták, hogy mindkettő rendelkezhet leírással.

Összegezve a fentebb leírtak teljes mértékben megállják a helyüket és validáltak üzleti, technikai és operációs nézőpontokból; de a modell-vezérelt megközelítéshez szükséges egy speciális úgynevezett fejlesztői nézőpontból tekinteni, mely nem feltétlenül jelenti azt, hogy az implementáció technológiai korlátok közé van szorítva, mert a szolgáltatást még biztosíthatja például egy személy vagy egy szervezet is például.



16. ábra Fejlesztői szemszög

Szolgáltatások ontológiája és taxonómiája

Szolgáltatás taxonómia

A szolgáltatásorientált architektúrákban új megoldásokat új alkalmazás-specifikus üzleti logika és funkcionalitás, valamint létező üzleti lehetőségek összekapcsolásával lehet létrehozni. Napjainkban ezek az üzleti lehetőségek főként cégen belül vagy fejlesztésekhez csomagokként megvásárolva léteznek. Ha a közeljövőt tekintjük, valószínűleg tovább folytatódik a Szoftver mint Szolgáltatás modell térnyerése, mint egy lehetőség a cégek számára, hogy „bérbe vegyenek” komponens alapú megoldásokat és üzleti lehetőségeket, ezáltal növelve a választható komponensek számát, melyek a cég összetett alkalmazásaiban lesznek.

Mint az első fejezetben említettem az ontológia legegyszerűbb megfogalmazásban egy adatmodell, mely egy területen belül lévő fogalmak halmazát és a köztük lévő kapcsolatokat

reprezentálja. A taxonómia pedig a dolgok egy osztályzása. A hierarchikus taxonómia adott objektumok egy halmazának fa struktúrájú kategorizálása. Ebben az alfejezetben a SOA-ban előforduló szolgáltatás osztályokat mutatom be és a köztük lévő kapcsolatokat, melyek mind a szoftverarchitektúra, mind az üzletközpontú aspektusokból jelentősek összetett SOA alkalmazások fejlesztése és karbantartása során.

Szoftverarchitektúra szemszögéből vizsgálva a kompozíciót, egy közös ontológia és taxonómia lehetővé teszi, hogy megállapítsuk a szolgáltatások azon közös jellemzőit, melyek eltűnhetnek az általános kategorizálás során. Azonban ezek a jellemzők hatással vannak az architektúrára és a SOA alapú megoldások tervezésére egészen az individuális szolgáltatás szinttől a teljes, összetett alkalmazásig. Kategorizálás segít a különböző komponensek szerepkörének megállapításában, így könnyebbé téve a közöttük lévő kölcsönös kapcsolatok definiálását, és könnyebbé teszi a szolgáltatások felfedezését is, mely az újrafelhasználhatóság további támogatását jelenti.

Az üzlet nézőpontjából nézve a kompozíciót, egy helyes közös ontológia és taxonómia segíthet az üzlettel összefüggő döntések meghozatalában, mint például, hogyan használjunk ki megfelelően egy lehetőséget, hogyan szervezzük meg a szolgáltatásokat.

Szolgáltatás kategóriák és típusok

Ha megvizsgáljuk a szolgáltatásokat, akkor látható, hogy két fő szolgáltatástípus van: az egyik az infrastrukturális természetű, melyek valamilyen egyszerű tevékenységet hajtanak végre és nem részei magának az alkalmazásnak, a másikba pedig azok tartoznak, melyek részei és építőkövei az alkalmazásnak.

A szoftverek egyszerű tevékenységek széles tárházát használják fel, az operációs rendszer által kínált alacsony szintű szolgáltatásoktól kezdve, mint például a memóriakezelés, I/O kezelés, egészen a magas szintű futtató környezet specifikus szolgáltatásokig. A SOA számára készített megoldások gyakran használnak olyan egyszerű tevékenységeket, mint a szolgáltatás-jogosultság keretrendszer vagy olyan szolgáltatások halmazát, melyeket az

elosztott infrastruktúra rendszer nyújt támogatásként. Ezeket Bus szolgáltatásoknak vagy infrastruktúraszolgáltatásoknak nevezik.

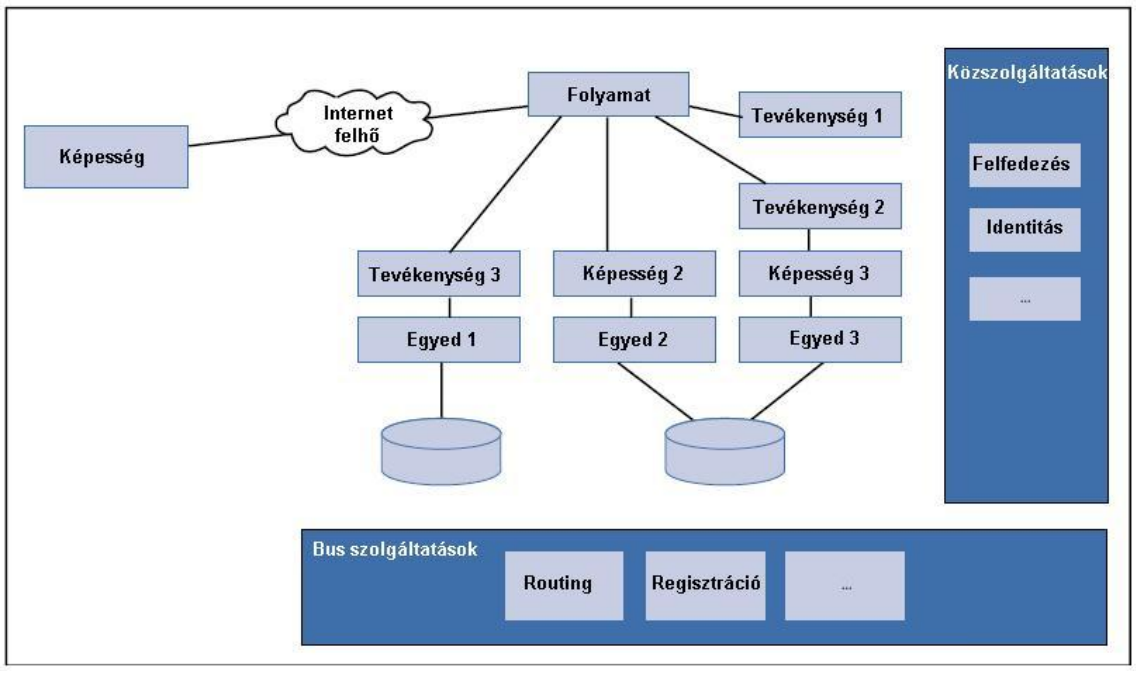
A Bus szolgáltatások két további alcsoportba oszthatóak. A Kommunikációs szolgáltatások, ahová az üzenettovábbító tevékenységek tartoznak: üzenet-útvonalmeghatározás, megjelenés-hitelesítés. A Közszolgáltatások pedig olyan lehetőségeket biztosítanak, melyek teljesen függetlenek az üzenetkezeléstől, mint például a szolgáltatás-keresés.

Az alkalmazásfejlesztés folyamatának hatékonysága tovább növelhető durva szemcsézettségű, magas szintű építőelemek felhasználásával. A RAD programozási környezetek, melyek a komponensorientált érában virágoztak fel, biztosítják a lehetőséget, hogy létező építőelemekből gyorsan és könnyen elkészítsük a funkcionális kompozíciót, majd ezt kombinálhatjuk alkalmazás-specifikus kódokkal teljesen új alkalmazások létrehozásához. Ilyen komponensekre találhatunk példákat a legáltalánosabb GUI konstrukcióktól, az adatbázis hozzáférés absztrakciókon keresztül, egészen a legspecifikusabb tevékenységekig, mint az eseménynaplózás és grafikonkészítés. A szolgáltatásorientált architektúrák kompozit alkalmazásai szintén használnak ilyen természetű alkotóelemeket a saját modelljükben. Ezeket az építőköveket Alkalmazásszolgáltatásoknak nevezik.

Alkalmazásszolgáltatások tovább oszthatóak Egyedszolgáltatásokra, melyek lehetővé teszik az üzleti példányok manipulálását. Képességszolgáltatásokra és Tevékenységszolgáltatásokra, melyek implementáljak az alkalmazás funkcionális építőköveit (gyakran nevezik ezeket komponenseknek vagy moduloknak). Folyamatszolgáltatásokra, melyek megszervezik és felügyelik az Egyed-, Képesség- és Tevékenységszolgáltatások működését úgy, hogy alkalmasak legyenek üzleti folyamatok modellezésére.

Az 17. ábrán szolgáltatás kategóriákba sorolt szolgáltatások egy kompozíciója látható egy üzleti folyamat implementálására. Látható egy Folyamatszolgáltatás, mely három Tevékenységszolgáltatásból és két Képességszolgáltatásból áll. Teljesen tipikus eljárásmód, hogy több más szolgáltatást fog össze, ezáltal modellezve egy összetett üzleti funkciót. A diagramon az egyik Tevékenységszolgáltatás (Tevékenység 2) használja a Képesség 3 elnevezésű Képességszolgáltatást, melynek egy lehetséges oka, hogy a Tevékenység 2 egy többlépéses tevékenységet hajt végre a Képesség 3-on keresztül vagy más a szolgáltatás

interfésze, mint a lentebb lévő szolgáltatásoknak. A Képesség 2 és 3 szolgáltatások a 2-es és 3-as számú Egyedszolgáltatások implementációjára támaszkodnak. Tovább haladva lefelé az ábrán érdekesség még, hogy az Egyed 2 és 3 szolgáltatások ugyanahhoz az adattárházhoz kapcsolódnak. Ez lehet azért, mert különböző egyedeket szeretnének elérni az adatbázisban vagy a Képesség 2 és 3 eltérő követelményei, esetleg adatmodellje miatt teljesen különböző módú hozzáférés szükséges. Azt is érdemes megjegyezni, hogy a Képesség 1 szolgáltatás a szervezet határain kívül van (melyet az Internet felhő jelent) és egy külső erőforrás szerepében jelenik meg, mely hozzászövődik a Folyamatszolgáltatás által implementált üzleti funkcióhoz.



17. ábra Szolgáltatás kategóriákban szolgáltatások egy lehetséges kompozíciója

Bus szolgáltatások

A Bus szolgáltatások nem igazán szolgáltatnak explicit üzleti értéket, sokkal inkább az üzleti folyamatok implementációjához szükséges SOA infrastruktúra részei. Tipikusan megvásárolt vagy központilag beépített komponensek, melyek összetett alkalmazásokat kínálnak és legtöbbször központosított irányításúak.

Kommunikációs szolgáltatások	
Fő cél	Üzenettovábbítás
Interfész	Nem dolgozza fel az alkalmazás üzeneteit; lehetnek irányító és megfigyelő interfészei
Állapotkezelés	Nincs alkalmazásszintű állapotkezelés
Tranzakciók	Nem alkalmazható (nem dolgozzák fel az alkalmazás üzeneteit)
Hibakezelés	Nincs alkalmazásszintű hibakezelés
Biztonság	Felhasználó/Alkalmazás/Mindkettő
Irányítás	Központosított
Készítés	Beépített vagy megvásárolt

Közszolgáltatások	
Fő cél	Generikus (nem alkalmazás-specifikus) infrastruktúra funkcionalitás
Interfész	Szolgáltatás interfész a szolgáltatás funkcionalitásának ismertetésére lehetnek irányító és megfigyelő interfészei
Állapotkezelés	Nincs alkalmazásszintű állapotkezelés; lehet saját állapota
Tranzakciók	Általában nem támogatott
Hibakezelés	Nincs vagy limitált alkalmazásszintű hibakezelés
Biztonság	Felhasználó/Alkalmazás/Mindkettő
Irányítás	Központosított
Készítés	Általában megvásárolt

Alkalmazásszolgáltatások

Ezek a szolgáltatások vesznek részt az üzleti folyamatok implementációjában, explicit üzleti értéket biztosítanak. Ide tartoznak szolgáltatások a teljesen általánosaktól, melyeket bármilyen összetett alkalmazásban lehet használni a vállalaton belül, az egészen speciálisakig, melyeket csak egyetlen összetett alkalmazás részeként lehet alkalmazni.

Egyedszolgáltatások	
Fő cél	Üzleti egyedek ismertetése és irányítása
Interfész	Egyedszint és tartomány-specifikus műveletek
Állapotkezelés	Alkalmazásállapot kezelés a szolgáltatás elsődleges célja
Tranzakciók	Atomi és/vagy foglalás minta
Hibakezelés	Korlátozott hibajavítás
Biztonság	Felhasználó/Alkalmazás/Mindkettő
Irányítás	Központosított
Készítés	Cégen belül/Megvásárlás/Bérlés

Képességszolgáltatások	
Fő cél	Általános üzleti lehetőségeket implementál
Interfész	Szolgáltatás interfész a fő funkcionalitás ismertetésére
Állapotkezelés	Tipikusan nem alkalmazás-specifikus állapotokat tárol
Tranzakciók	Nincs állapot, ezért nincs szükség tranzakciókra; implementálhat atomi és/vagy foglalás mintát egy Egyedszolgáltatáson
Hibakezelés	Korlátozott hibajavítás
Biztonság	Alkalmazásszintű
Irányítás	Központosított
Készítés	Cégen belül/Megvásárlás/Bérlés

Tevékenységszolgáltatások	
Fő cél	Speciális alkalmazás-specifikus üzleti lehetőségeket implementál
Interfész	Szolgáltatás interfész a fő funkcionalitás ismertetésére
Állapotkezelés	Nem tárol alkalmazás-specifikus állapotot
Tranzakciók	Támogathat atomi tranzakciókat és/vagy implementálhat foglalás mintát létező rendszer felett
Hibakezelés	Korlátozott hibajavítás
Biztonság	Alkalmazásszintű
Irányítás	Alkalmazásonként
Készítés	Cégen belül

Folyamatszolgáltatások	
Fő cél	Üzleti folyamat implementálása, más szolgáltatások csoportba szervezésével
Interfész	Szolgáltatás interfész a felhasználó által használt alkalmazásokban
Állapotkezelés	Kezeli a folyamatállapotokat
Tranzakciók	Nincs atomi tranzakció támogatás; implementálhat foglalás mintát
Hibakezelés	Üzleti logika részeként implementált
Biztonság	Felhasználó
Irányítás	Alkalmazásonként
Készítés	Cégen belül

Szolgáltatások ontológiájának és taxonómiájának összegzése

Ha az architekt tisztában van ezekkel a kategóriákkal, akkor könnyen osztályozhat már meglévő vagy új szolgáltatásokat. Mindezek mellett segít a különböző szolgáltatásokba foglalt funkcionalitás megfelelő definiálásában a kompozíció és újrafelhasználás szemszögéből. Ez az osztályozás használható akár új rendszerek tervezéséhez, akár már létező rendszerek újratervezéséhez, mindkét esetben észrevehető előnyökkel jár.

Az üzleti döntések szemszögéből ezen komponensek lehetőségeinek teljes megértése könnyebbé teszi a készítés vagy vásárlás vagy kölcsönzés problémahármas megoldását illetve lehetővé teszi olyan szolgáltatások készítését, melyeket mások számára elérhetővé lehet tenni.

Az ontológia engedélyezett szolgáltatásoktól a szolgáltatás engedélyezett ontológiákig

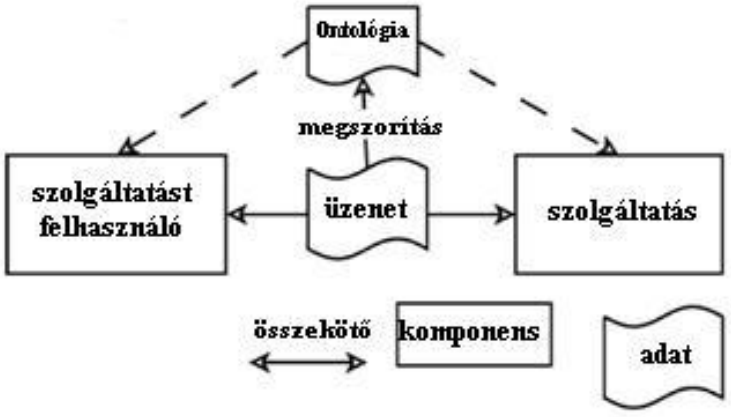
Egy harmadik közismert, bemutatásra váró közelítés maradt utoljára. Bár jelenleg túl sok a feltételezés, melyet igazolni kell, de elképzelhető, hogy egy szoftvertervezés során hasznos modellé válhat idővel.

Célja a szoftver architektúrák szemszögéből egy olyan Ontológia engedélyezett szolgáltatásorientált architektúra minta definiálása, mely általános, technológia- és ontológia-független keretrendszert nyújt az ontológiák és SOA integrálásához. Ehhez az elmélethez elkészítettek egy MoRe néven ismert RDF/S engedélyezett SOA-t, mely egy pragmatikus keretrendszer ontológia engedélyezett alkalmazások készítéséhez. A fejlesztés során arra a következtetésre jutottak, hogy nem csak egy ontológia képes megosztott szemantikát hozni SOA környezetbe, de a SOA biztosít egy engedélyező mechanizmussal rendelkező ontológiát, mely megengedi, hogy tetszőleges szolgáltatásokat csatolhassunk egy ontológiához.

Ontológia engedélyezett SOA (Onto=>SOA)

A kiindulási pont a 2. fejezetben bemutatott SOA definíció, mint architektúrális minta (7. ábra). Az Onto=>SOA nézőpont szemantikailag gazdag üzenetek közvetlen cseréjét feltételezi a feldolgozó komponensek között (szolgáltatások és szolgáltatás használók), és ezek mellett kombinálható bármely SOA-kompatibilis architektúrával (WSDL/SOAP vagy REST) és bármely ontológia nyelvvel. Az Onto=>SOA mintát a SOA architektúrális mintából az architektúra adatelemein bevezetett kiegészítő korlátozással lehet származtatni: ontológiát kell használni a séma-alapú üzenetek kifejezéséhez. Ez a megszorítás kifejezi a szemantikai

átjárhatóságot az üzenetek között, feltételezve, hogy mind a szolgáltatás és mind a szolgáltatás-használó tud dolgozni ontológia-alapú üzenetekkel.



18. ábra Ontológia engedélyezett szolgáltatásorientált architektúra minta

Az Onto=>SOA-ban egy szolgáltatásnak fontos, hogy egy bizonyos ontológiát támogasson és helyesen interpretálja az ontológia által létrehozott üzeneteket. Ha ezt feltesszük, akkor itt eltekinthetünk az üzenetek szintaktikájától és szemantikájától, és az ontológiában definiált ontológia nézőpontra fókuszálhatunk. Bár elő van írva az ontológia nyelv használata az üzenetsémákban, de nincs megszabva a használandó üzenetnyelv. Azonban ez újabb problémát vethet fel: ha különböző nyelveket alkalmazunk a dokumentumokban, akkor egy megfelelő összerendelési mechanizmus szükséges a séma és a dokumentum összekapcsolásához. Ehhez szükséges egy kiegészítő architektúra elem – szemantikus összekötő – mely felelős az ontológia terminológiája és az üzenet tartalma közötti kapcsolat kezeléséért.

Például alkalmazhatjuk az RDF sémanyelvet a séma kifejezésére, ebben az esetben az RDF lenne a legoptimálisabb jelölt az üzenet nyelvére. De ha XML lenne az alkalmazott technológia RDF helyett, akkor egy kiegészítő mechanizmust kellene alkalmazni, hogy az XML üzenetben tárolt struktúrákat RDF séma ontológiában definiált fogalmakhoz hozzá lehessen rendelni.

Ahhoz, hogy az újonnan származtatott Onto=>SOA-nak a SOA-val való kompatibilitását igazolni lehessen, validálni kell a bevezetett új megszorítást a séma kiterjeszthetőségre vonatkozó általános SOA követelmények tükrében. A legtöbb modern alkalmazásban az XML és az XML séma biztosít egy egységes szintaxist és szótár definíció mechanizmust az üzenetek számára. Azonban az XML sémanyelv csak a struktúrális aspektusokat kezeli, az implicit módon definiált szótárra hagyva a szemantikát. Megoldatlanul hagyja a szemantikai átjárhatóságot a feldolgozó komponensek között, de ugyanakkor nem korlátozza a rugalmasságot tartomány-specifikus szótárak definiálásában.

Egy ontológia biztosíthat tartományszótárt, az alapul szolgáló nyelv ontológiai primitívek függvényében precízen definiált szemantikákat. De az ontológia nyelvek, mint az RDFS vagy az OWL fix szemantika kifejezésére lettek tervezve, ezért nem tölthetik be egy kiterjeszthető követelmény szerepét, tekintettel a séma szemantikákra. Így a felhasználót az ontológia nyelv által nyújtott lehetőségek korlátozzák az alkalmazástartomány modellezése során. Ez azt jelenti, hogy minél inkább korlátozó egy ontológia nyelv, annál rugalmatlanabb az erre épülő Onto=>SOA és a minta annál inkább eltávolodik az eredeti SOA követelményektől.

Tehát ezen elmélet teljesüléséhez, és a SOA által szükséges rugalmassági szint eléréséhez az ontológia nyelv szemantikájának kiterjeszthetőnek kell lennie. Napjainkban ilyen nyelvek nem léteznek, és nehéz előre megjósolni, hogy megjelennek-e valaha is. Ezért, ha egy kiterjeszthető ontológia nyelv elérhetetlen, mint jelen esetben, akkor a legkevésbé korlátozó (pl. RDFS) megfelelő jelölt lehet.

Szolgáltatás engedélyezett ontológiák (Onto<=SOA)

Mint említettem a mai modern ontológia nyelvek nem teljesen megfelelőek SOA környezetben történő alkalmazásra a fix és nem kiterjeszthető szemantika miatt, bár néhányan úgy tartják, hogy egy ontológia nyelv kiterjeszthetősége kritikus tényező a sikerhez olyan összetett és változatos környezetben, mint a web. Egyes állítások szerint az Onto=>SOA megközelítés pont egy ilyen kiterjeszthetőségi mechanizmust biztosít számunkra. Egy ilyen Onto=>SOA származtatás megvalósításához 2 fő komponenst kell kifejleszteni: egy szolgáltatást és egy ontológiát. Az ontológia a szolgáltatás egy absztrakt specifikációjaként

működik, mely közelítés megengedi, hogy egy magasabb absztrakciós szintre lépjünk a szolgáltatások felé és az Onto-SOA-t közvetlenül úgy fejlesszük, mint egy Onto-SOA ontológiát. Továbbhaladva a gondolattal, így talán az is elérhető, hogy egy ontológia fejlesztési folyamatot úgy tekintsünk, mint az Onto-SOA fejlesztési folyamat egy részét.

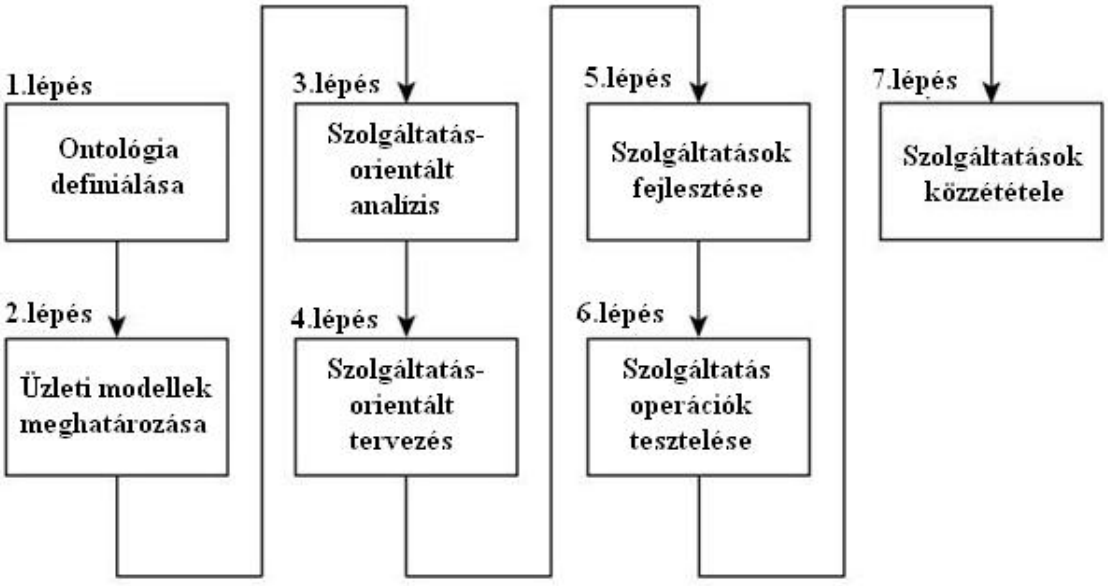
Ebben a pillanatban az Onto-SOA-n elkövetett nézőpontváltással felvetődik a kérdés, hogy lehetséges-e olyan mechanizmus létrehozása, mely egy ontológiát szolgáltatás engedélyezetté képes tenni (Onto<=SOA). A kérdésre válaszként egy újabb elmélet született, mely szerint egy szolgáltatást magába foglalhat egy ontológia nyelv, ezáltal lehetőséget nyújtva a fejlesztőknek tetszőleges szolgáltatások csatolására az ontológiai fogalmakhoz, ezen a módon definiálva úgynevezett operációs szemantikát hozzájuk. Az operációs szemantika nem más, mint az ontológiában definiált fogalmak szolgáltatás által történő leképezésének a módja. Ez lehetőséget ad, hogy bevezessünk új modellező primitíveket, melyek alkalmazásspecifikus követelmények, így elérhető a SOA által kívánt rugalmasság. Az operációs szemantikák más természetűek, mint a modern ontológia nyelvekben alkalmazott szabványos szemantikák, ezért nem helyettesítik őket, de nagyon jól kiegészítik.

Az elmélet összegzése szerint a két megközelítés összekapcsolásával érhető el az Onto⇔SOA, mely nem más, mint: az Onto=>SOA egy leszármazottja, mely kombinálja az RDF/S nyelveket és a REST webszolgáltatások elemeit, illetve az Onto<=SOA mechanizmus implementációja, mely lehetővé teszi a REST webszolgáltatások csatolását RDF/S ontológiákhoz. Ha a jövőben valóban teljesülni fognak az említett követelmények, akkor az ontológiák és szolgáltatások ilyen módú kombinálása az ontológia- és alkalmazásfejlesztők számára is előnyökkel járhat, egy ontológia engedélyezett szoftverrendszert eredményezve.

4. fejezet: Összefoglalás

Úgy tűnik napjainkban mind az ontológiák, mind a szolgáltatásorientált architektúrák térhódítása elindult. Sok szakember folytat kutatásokat e két témakörben, és próbálják a kombinálásukból származó előnyöket felmérni és dokumentálni. A SOA-ról több könyvet is író Thomas Earl szerint az ontológiák lehetnek az úgynevezett felülről lefelé tervező stratégia alapja a szolgáltatásorientált architektúrák fejlesztése során. Ha egy többlépéses felülről lefelé haladó folyamat első lépése ontológiákkal kezdődik, akkor a legmagasabb minőségi szinten lévő SOA lesz az eredmény.

Ez egy nyitott, kiterjeszthető, könnyen módosítható architektúra, mely támogatja a szolgáltatásorientáltságot és autonóm, QoS-képes, sokrétű, átjárható, felfedezhető és könnyen újrafelhasználható, webszolgáltatásként implementált szolgáltatásokból áll.

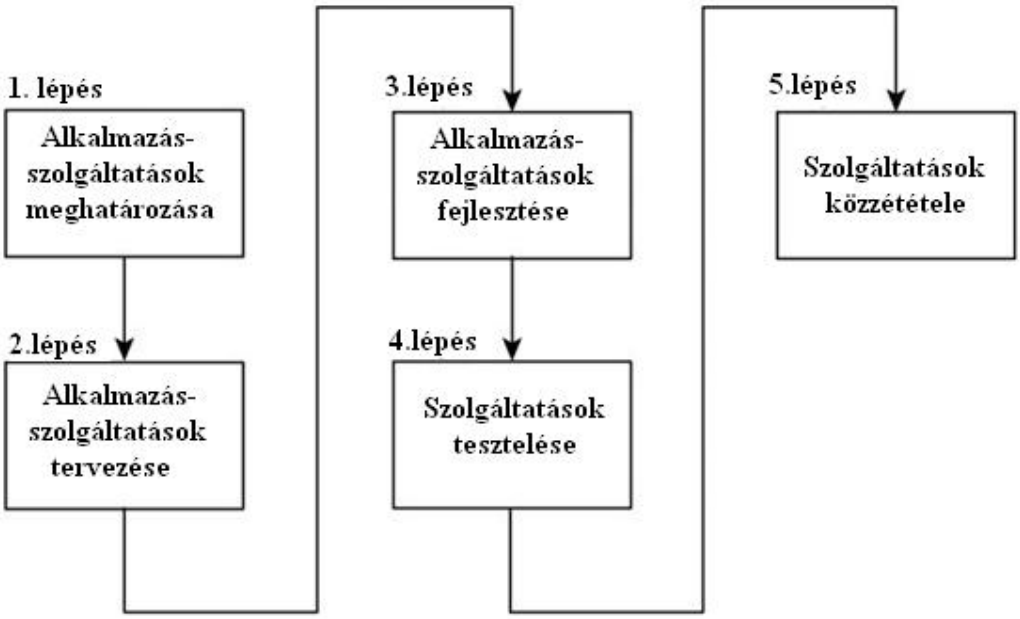


19. ábra Felülről-lefelé tervező stratégia

Ellenben - Earl szerint - a másik tradicionális szoftvertervezési megközelítés, azaz az alulról-felfelé tervező stratégia megvalósítása, mely SOA környezetben általában webszolgáltatásokkal történik, nem lehetséges ontológiákkal. Sokak szerint ez nem is egy validált technika komplexebb SOA eléréséhez, mert a kezdeti lépésekben tényleg szolgáltatások vannak, de olyanak, melyek az éppen aktuális problémát próbálják megoldani

szabványosított eljárás szerint. Később pedig valószínűleg kialakul egy már átlátható szabványosított réteg az átláthatatlan kompozíciójú alsóbb rétegben található szolgáltatások felett.

Ezért szoktak egy harmadik stratégiát is emlegetni, az úgynevezett agilis stratégiát, mely próbál elfogadható egyensúlyt találni szolgáltatásorientált szemléletmód alkalmazására az üzleti elemző környezetben anélkül, hogy a technológiai környezetbe először bevezetésre kerülne a webszolgáltatás technológia.



20. ábra Alulról-felfelé tervező stratégia

Ezt sokan el is fogadnák, ha nem lenne az a tény, hogy lehetséges informális ontológiák létrehozása nem SOA környezetben alulról-felfelé haladó folyamatok során. Ha ez tényleg lehetséges, akkor megéri kutatásokat folytatni az informális ontológiák alkalmazásáról alulról-felfelé haladó SOA fejlesztésbe. Úgy gondolom, ha ez az elmélet bebizonyosodik, akkor jelentős áttörések lesznek a SOA fejlesztésekben, és lehetséges lesz ezzel a módszerrel is elérni a legmagasabb minőségi szinten lévő SOA-t. Egyes szűk réteg még tovább vitte a gondolatmenetet: mivel az agilis stratégia ötvözi mind az alulról-felfelé,

mind a felülről lefelé tervező szemléletmód előnyeit, megérné megvizsgálni, hogy esetleg van-e lehetőség itt is informális ontológiák alkalmazására.

Egyetlen következtetésként azt vonnám le: hogy bár még mindig sok a megválaszolatlan kérdés, jelenleg mégis úgy tűnik, hogy a közeljövő egyik nagy áttörése lehet az ontológiák alkalmazása a szolgáltatásorientált architektúrák fejlesztésében mind az ületi, mind az információ technológia világában.

Irodalomjegyzék

[1] SPARQL Query Language for RDF, E. Prud'hommeaux, A. Seaborne (Eds.), W3C Working Draft, http://www.w3.org/TR/rdf-sparql-query/, 2005.

[2] RDF Vocabulary Description Language 1.0: RDF Schema, D. Brickley, R.V. Guha (Eds.), W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-schema, 2004.

[3] SKOS Core Vocabulary Specification, A. Miles, D. Brickley (Eds.), W3C Working Draft, http://www.w3.org/TR/swbp-skos-core-spec, 2005.

[4] Web Services technology: http://www.w3.org/2002/ws/

[5] Juhász Zoltán, Sipos Gergely, Zsemlye Tamás: Szolgáltatás-orientált technológiák és ezek hatása a jövő üzleti alkalmazásaira: http://pds.irt.vein.hu/files/publications/Juhasz_Neumann.pdf

[6] Sun Microsystems SOA honlapja: http://www.sun.com/products/soa/index.jsp

[7] Thomas Earl - Service-Oriented Architecture: Concepts, Technology, and Design

[8] OMG CORBA honlapja: http://www.omg.org/gettingstarted/corbafaq.htm

[9] OMG hivatalos honlapja: http://www.omg.org/

[10] The Jini Specification http://www.sun.com/software/jini/jini_technology.html

[11] Web Service Description Language: http://www.w3.org/TR/wsdl

[12] Universal Description, Discovery and Integration: http://www.uddi.org/

[13] Simple Object Access Protocol: http://www.w3.org/TR/soap/

[14] W3C: Web Services Architecture. http://www.w3.org/TR/ws-arch (2004)

[15] Fielding, R.T.: Principled design of the modern web architecture. In: Proceedings of the 2000 International Conference on Software Engineering. (2000)

[16] Cabral, L., Domingue, J., et al: Approaches to semantic web services: an overview and comparisons. In Bussler, C., Davies, J., Fensel, D., Studer, R., eds.: ESWS. Volume 3053 of Lecture Notes in Computer Science., Springer (2004) 225–239

[17] W3C: Semantic Web Recommendations and Specifications: RDF/S, OWL, etc. (http://www.w3.org/sw)

[18] Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, UNIVERSITY OF CALIFORNIA (2005) http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html

[19] http://msdn2.microsoft.com/en-us/arcjournal/bb491121.aspx

[20] http://ianlumb.wordpress.com/2007/03/31/on-the-use-of-informal-ontologies-in-the-delivery-of-service-oriented-architectures-soas/

[21] http://km.aifb.uni-karlsruhe.de/ws/swese2006/final/korotkiy_full.pdf

[22] http://weblog.infoworld.com/realworldsoa/archives/2006/11/managing_soa_se.html

[23] http://www.opengroup.org/projects/soa-ontology/uploads/40/12153/soa-ont-06.pdf

[24] http://www.w3.org/2006/Talks/0318-Budapest-IH/cikk.html

[25] http://www.w3c.hu/forditasok/OWL/REC-owl-ref-20040210.html

[26] http://hu.wikipedia.org/wiki/Ontol%C3%B3gia_(mesters%C3%A9ges_intelligencia)

[27] http://en.wikipedia.org/wiki/Ontology_(computer_science)

[28] http://www.w3.org/2006/Talks/0318-Budapest-IH/

[29] http://www.w3.org/2001/sw/WebOnt/

[30] http://www.w3.org/RDF.

[31] http://www.xml.com/pub/a/ws/2001/04/04/soap.html

[32] http://www.oreillynet.com/pub/wlg/3005

[33] http://www.ibm.com/developerworks/architecture/library/ar-soastyle/

[34] http://en.wikipedia.org/wiki/Martin_Fowler

[35] http://www.martinfowler.com/bliki/ServiceOrientedAmbiguity.html

[36]http://msdn2.microsoft.com/en-us/architecture/bb833022.aspx#_Acknowledgements

[37] http://www.w3.org/2003/Talks/0521-hh-wsa/slide8-0.html

[38] http://www.service-architecture.com/web-services/articles/serviceoriented_architecture_soa_definition.html

[39] http://www.w3.org/2005/rules/.

[40] http://www.w3.org/2004/OWL/.

[41] http://people.inf.elte.hu/santa/oktatasi-anyagok/segedletek-pdf/segedlet5.pdf

[42] *OIL:* http://www.cordis.lu/ist/projects/99-10132.htm

[43] http://www.w3.org/TR/daml+oil-reference