

31 SZOLGÁLTATÁSORIENTÁLT SZOFTVERTERVEZÉS

TÉMAKÖRÖK

A fejezet célja, hogy bevezetést nyújtson a szolgáltatásorientált szoftverfejlesztésbe, amely egyre nagyobb jelentőséggel bír az üzleti alkalmazások fejlesztése kapcsán. Mire a fejezet végére érünk:

- megértjük a webszolgáltatásokat és annak szabványait, valamint azt, hogy ezek hogyan támogatják a szervezetek közötti számítástechnikai együttműködést;
- ismerni fogjuk a szolgáltatások tervezési folyamatát, amelynek célja újrafelhasználható webszolgáltatások előállítása;
- érteni fogjuk a szolgáltatáskompozíció fogalmát, ami a szolgáltatásorientált fejlesztés egyik eszköze;
- át fogjuk látni, hogyan használhatók az üzleti folyamatmodellek szolgáltatásorientált rendszerek tervezésének alapjául.

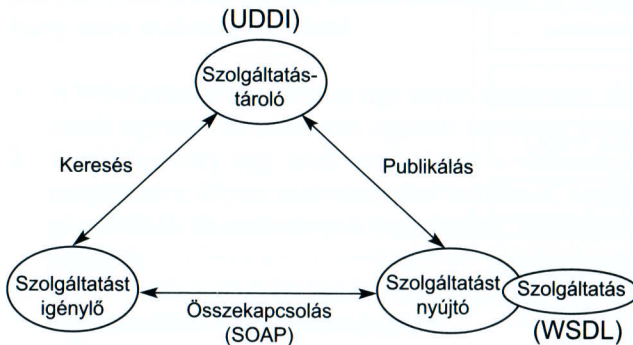
TARTALOM

- 31.1. Szolgáltatások mint újrafelhasználható komponensek
- 31.2. Szolgáltatások tervezése
- 31.3. Szoftverfejlesztés szolgáltatásokkal

A 12. fejezetben úgy vezettem be a szolgáltatásorientált architektúra fogalmát, mint olyan eszközt, amely képes leegyszerűsíteni a szervezetek közötti számítástechnikai együttműködést. A szolgáltatásorientált architektúra (service-oriented architecture, SOA) lényegében az elosztott rendszerek fejlesztésének módja, ahol a rendszerek komponensei különálló szolgáltatások. Ezek a szolgáltatások földrajzilag egymástól távol elhelyezkedő számítógépeken futhatnak. A szolgáltatások kommunikációjának és az adatcserének a támogatására számos szabványos protokoll született. Következésképpen a szolgáltatások függetlenek a platformtól és az implementációs nyelvtől. A szoftverrendszerek felépíthetők különböző gyártóktól származó szolgáltatásokból, és ezek a szolgáltatások zökkenőmentesen képesek együtt dolgozni.

A 31.1. ábra a webszolgáltatások használatát illusztrálja. A szolgáltatások gyártói megtervezik és implementálják a szolgáltatásokat, majd egy WSDL nevű

nyelven, amelyet később tárgyalok, specifikálják azokat. Ezenkívül a szolgáltatásokról információkat publikálnak egy általánosan elérhető tárolóban az UDDI publikációs szabvány segítségével. A szolgáltatások igénybe vevői (néha szolgáltatáskliensnek is nevezik őket), akik szeretnének használni egy szolgáltatást, keresést hajtanak végre az UDDI-tárolóban, hogy megtalálják azt, valamint annak nyújtóját. Ezután alkalmazásukat összekapcsolhatják az adott szolgáltatással, és kommunikálhatnak azzal. A kommunikációhoz általában a SOAP nevű protokollt használják.

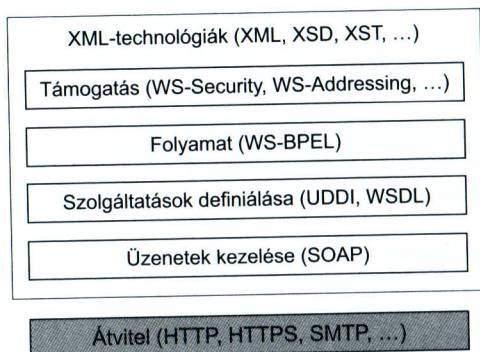


31.1. ábra. Szolgáltatásorientált architektúra

Napjainkban általánosan elfogadott, hogy a szolgáltatásorientált architektúra hatalmas fejlődést jelent, különösen az üzleti alkalmazásrendszerek szempontjából. Rugalmasságot biztosít, mivel a szolgáltatások elkészíthetők helyileg, illetve igénybe vehetők külső szolgáltatóktól. A szolgáltatások bármilyen nyelven implementálhatók. A működő rendszerek szolgáltatásokként történő becsomagolásával (21. fejezet) a cégek megőrizhetik befektetéseiket, és elérhetővé tehetik szolgáltatásaikat alkalmazások széles köre számára. A SOA lehetővé teszi, hogy a vállalat különböző részein esetlegesen használt különböző platformok és implementációs technológiák együttműködhesse. Ami a legfontosabb, a szolgáltatások segítségével a cégek és más szervezetek képesek az együttműködésre, valamint egymás üzleti funkcióinak használatára. Így azok a rendszerek, amelyekben széles körű információcsere folyik a vállalat határain keresztül, könnyen automatizálhatók. Ilyen rendszerek a szolgáltatási láncrendszerek, amelyekben az egyik cég termékeket rendel a másiktól.

A szolgáltatásorientált architektúrák sikerének titka talán az, hogy a technikai fejlesztéseket a kezdetektől aktív szabványosítási folyamat követte. A legnagyobb hardver- és szoftvergyártók elfogadták ezeket a szabványokat. Ennek köszönhetően a szolgáltatásorientált architektúrák nem szenvednek inkompatibilitási problémáktól, ami gyakori kísérője a technikai újításoknak, ha a különböző gyártók a saját fejlesztésű technológiát használják. Így az olyan problémák, mint az inkompatibilis komponensmodellek a 19. fejezetben tárgyalt komponensalapú szoftverfejlesztés esetén, nem jelentkeztek a szolgáltatásorientált rendszerfejlesztéssel kapcsolatban.

A 31.2. ábrán azok az alapvető szabványok láthatók, amelyek a webszolgáltatások támogatására születtek meg. Elvben a szolgáltatásorientált megközelítés más protokollok használata mellett is alkalmazható, a gyakorlatban azonban a webszolgáltatások dominanciája jelentős. Jóllehet, a webszolgáltatások függetlennek minden átviteli protokolltól, mégis gyakorlatban a HTTP és a HTTPS protokollok használata általános.



31.2. ábra. Webszolgáltatás-szabványok

A webszolgáltatás protokolljai minden szempontból lefedik a szolgáltatásorientált architektúrát, kezdve a szolgáltatások információcseréjének alapvető mechanizmusától (SOAP) a programnyelvi szabványokig (WS-BPEL). Ezen szabványok mindegyike XML-alapú, amely emberek és számítógépek számára egyaránt olvasható jelölés. Az XML lehetővé teszi strukturált adatok definícióját, ahol a szöveg beszédes címkékkel van ellátva. Az XML-t támogató technológiák széles tárháza jelent meg. Ilyen a sémadefiníciót támogató XSD, amely az XML-leírások bővítését és manipulálását szolgálja. Erl (Erl, 2004) kiváló összefoglalót nyújt az XML-technológiákról és azok szerepéről a webszolgáltatásokban.

A webszolgáltatás-orientált architektúrák szabványai röviden a következők:

1. **SOAP.** Ez egy üzenetcsere-szabvány, amely a szolgáltatások közötti kommunikációt támogatja. Definiálja a szolgáltatások üzeneteinek szükséges és opcionális komponenseit.
2. **WSDL.** Webszolgáltatás definíciós nyelv (Web Service Definition Language, WSDL) definiálja, hogyan kell a szolgáltatások gyártóinak elkészíteniük szolgáltatásaik interfészeit. Alapvetően ennek köszönhető, hogy a szolgáltatások interfészei (a szolgáltatás műveletei, a paraméterek és típusaik), valamint azok csatlakoztatása szabványos módon definiálható.
3. **UDDI.** Az Univerzális leírás, felfedezés és integráció (Universal Description, Discovery and Integration, UDDI) szabvány definiálja, hogy milyen komponenseket kell tartalmaznia a szolgáltatások specifikációinak, hogy könnyen felfedezhetők legyenek. Ez információkat tartalmaz a szolgáltatás gyártójáról, a szolgáltatásról, a szolgáltatás leírásának helyéről (általában WSDL-ben

kifejezve), valamint az üzleti kapcsolatokról. Az UDDI-tárolók segítségével fedezhetik fel a felhasználók az elérhető szolgáltatásokat.

4. **WS-BPEL.** Szabványos munkafolyamat nyelv, amely különböző szolgáltatásokat tartalmazó folyamatprogramok definiálására használatos. A folyamatprogramok fogalmát a 31.3. fejezetben tárgyalom.

A fő szabványok mögött számos támogató szabvány áll, amelyek mind a SOA speciálisabb aspektusaira koncentrálnak. Rengeteg támogató szabvány létezik, melyek különböző típusú alkalmazásokban hivatottak támogatni a SOA-t. Néhány ezen szabványok közül:

1. A **WS-Reliable Messaging** egy olyan szabvány, amely biztosítja, hogy az üzenetek egyszer, és pontosan egyszer kerülnek kézbesítésre.
2. A **WS-Security** egy szabványhalmaz a webszolgáltatások biztonságának támogatására. Olyan szabványokat tartalmaz, amelyek specifikálják a biztonsági politikák és szabványok definícióját. Idetartozik a digitális aláírások használata is.
3. A **WS-Addressing** azt határozza meg, hogy egy SOAP-üzenetben hogyan kell reprezentálni a címinformációkat.
4. A **WS-Transactions** az elosztott szolgáltatások közötti tranzakciók koordinációjának módját definiálja.

A webszolgáltatás-szabványok témaköre hatalmas, meghaladja ezen könyv kereteit. Áttekintésként Erl könyveit ajánlom (Erl, 2004; 2005). A szabványok részletes leírásai a weben is elérhetők ingyenes dokumentumként.

Ahogy azt majd a következő alfejezetben tárgyalom, a szolgáltatások felfoghatók egyszerű újrafelhasználható absztrakciókként, így ez a fejezet a 18. és 19. fejezetek kiegészítése, amelyekben a szoftver-újrafelhasználásról volt szó. A fejezet témái tehát a következők:

1. Szolgáltatások tervezése. Megbízható és újrafelhasználható szolgáltatások fejlesztésével foglalkozik. Alapvetően a feladat újrafelhasználható szoftverek fejlesztése.
2. Szoftverfejlesztés szolgáltatásokkal. Megbízható szoftverrendszerek fejlesztésével foglalkozik, amelyek vagy csak szolgáltatásokat használnak, vagy szolgáltatásokat más komponensekkel összefüggésben. Alapvetően a feladat újrafelhasználható szoftverek fejlesztése.

A szolgáltatásorientált architektúrák és a szolgáltatásorientált szoftverfejlesztés jelenleg is „forró téma”. A szolgáltatásorientált megközelítés alkalmazását hatalmas üzleti érdeklődés övezi, azonban a könyv írásának időpontjában igen csekély mértékű gyakorlati tapasztalattal rendelkezünk a témával kapcsolatban. Az ilyen forró témák mindig földtől elrugaszkodott víziókat sugallnak, és gyakran többet ígérnek, mint amit teljesíteni képesek. Például a SOA-ról szóló könyvükben Newcomer és Lomow (Newcomer és Lomow, 2005) a következőket jelentik ki:

A szolgáltatásorientált szemléletet a legfontosabb technológiák és a webszolgáltatások univerzális alkalmazhatóságának egyesülése vezérli. Ez a cégek agilitásának javításával, az új termékek és szolgáltatások piacra kerülésének meggyorsításával, az IT-költségek csökkentésével és a működési hatékonyság növelésével kecsegtet.

A szolgáltatásorientált szoftvertervezés valóban legalább olyan fontos fejlődést jelent, mint az objektumorientált szoftvertervezés. Azonban nem szabad elfelejtenünk, hogy sok évnek el kell még telnie addig, amíg a SOA víziója valósággá válik. Mivel a szolgáltatásorientált szoftverfejlesztés még gyerekcipőben jár, nincsenek jól kipróbált szoftvertervezési módszereink az ilyen típusú rendszerekhez. Ezért a hangsúlyt a szolgáltatások tervezésének és implementációjának általános kérdéseire helyezem, valamint arra, hogyan lehet a rendszereket szolgáltatásokból felépíteni.

31.1. SZOLGÁLTATÁSOK MINT ÚJRAFELHASZNÁLHATÓ KOMPONENSEK

A 19. fejezetben bemutatam a komponensalapú szoftverfejlesztést (CBSE), ahol a szoftverrendszereket néhány szabványos komponensmodellén alapuló komponensből állítjuk össze. A szolgáltatások a komponensek természetes továbbfejlődései, itt a komponensmodell a webszolgáltatásokhoz kapcsolódó szabvány-együttes. A szolgáltatás fogalma tehát az alábbi módon definiálható:

Lazán csatolt, újrafelhasználható szoftverkomponens, amely olyan diszkrét funkcionalitást zár be, amely elosztható, és programozói eszközökkel elérhető. A webszolgáltatás olyan szolgáltatás, amely szabványos internetes és XML-alapú szabványokkal érhető el.

A szolgáltatások és a 19. fejezetben definiált szoftverkomponensek közötti alapvető különbség az, hogy a szolgáltatások függetlenek és lazán csatoltak. Azaz mindig ugyanúgy kell viselkedniük, tekintet nélkül a végrehajtási környezetre. Az interfészük „biztosított interfész”, amely elérést biztosít a szolgáltatás funkcionalitásához. A szolgáltatásokkal szemben támasztott egyik legfontosabb igény, hogy legyenek függetlenek és felhasználhatók különböző kontextusokban. Emiatt nem rendelkeznek „szükséges interfésszel”, ami CBSE esetén azt definiálja, hogy a komponensnek milyen más komponensekre van szüksége.

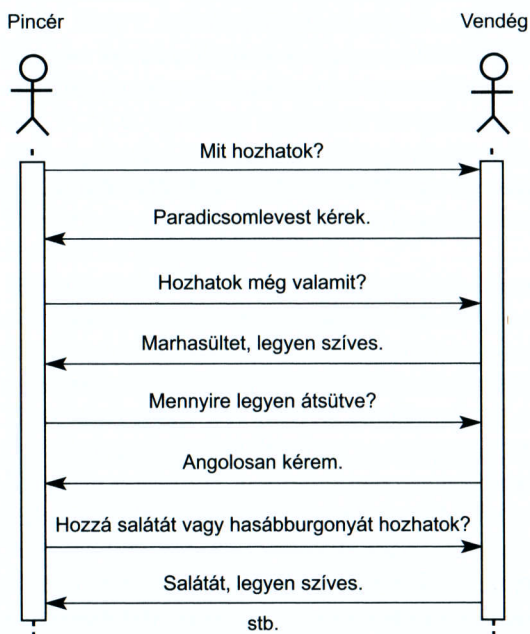
A szolgáltatások az interneten is eloszthatók. Ekkor XML-üzenetek cseréjével kommunikálnak, és ezeket az üzeneteket olyan internetes átviteli protokollok segítségével továbbítják, mint a HTTP és a TCP/IP. A szolgáltatás úgy definiálja, mit vár egy másik szolgáltatástól, hogy igényeit felsorolja egy üzenetben, és elküldi azt a szolgáltatásnak. A megszólított szolgáltatás feldolgozza az üzenetet, végrehajtja a számítást, majd ha ezt befejezte, választ küld az igénylő szolgáltatásnak. Ez a szolgáltatás azután feldolgozza a választ, és kinyeri belőle a szükséges infor-

mációkat. A komponensekkel ellentétben, a szolgáltatások nem hívják meg más szolgáltatások módszereit.

A módszerhívással történő kommunikáció és az üzenetküldéssel történő kommunikáció közti különbség illusztrálásához képzeljünk el egy olyan szituációt, amelyben ételt rendelünk egy étteremben. Mikor beszélgetünk a pincérrel, egy, a rendelésünket definiáló, szinkron interakció-sorozat részesei vagyunk. Ez hasonló a szoftverrendszerek interakciót folytató komponenseihez, amikor egy komponens meghívja egy másik valamely módszerét. A pincér ezután lejegyzi a rendelésünket a társaságunkban lévő személyek igényeivel együtt, majd a rendelés részleteit tartalmazó cetlit leadja a konyhán elkészítésre. Tulajdonképpen a következő történik: a pincér szolgáltatás üzenetet küld a konyha szolgáltatásnak, amelyben definiálja az elkészítendő ételt.

A fentieket a 31.3. és a 31.4. ábrán mutatom be. Az első a szinkron rendelési folyamatot szemlélteti, míg a másodikon egy hipotetikus XML-üzenet látható, amely reményeim szerint nem kíván magyarázatot. Az üzenet egy közös asztalnál ülő három vendég rendelését definiálja. A különbség nyilvánvaló – a pincér egy interakció-sorozaton keresztül rögzíti a rendelést, amelyben minden egyes interakció a megrendelés egy részét definiálja. Azonban a konyhával csak egyetlen interakciót folytat a pincér, mégpedig akkor, amikor leadja a teljes rendelést tartalmazó üzenetet.

Ha egy webszolgáltatást akarunk használni, ismernünk kell annak helyét (az URI-ját) és interfészének részleteit. Ezeket egy, az XML-alapú WSDL nyelven kifejezett, szolgáltatásleírásban találhatjuk meg. A WSDL-specifikáció három dol-



31.3. ábra. Szinkron interakció ételrendelés során


```

<starter>
  <dish name = „soup” type = „tomato” />
  <dish name = „soup” type = „fish” />
  <dish name = „pigeon salad” />
</starter>
<main course>
  <dish name = „steak” type = „sirloin” cooking = „medium” />
  <dish name = „steak” type = „fillet” cooking = „rare” />
  <dish name = „sea bass” />
</main/>
<accompaniment>
  <dish name = „french fries” portions = „2” />
  <dish name = „salad” portions = „1” />
</accompaniment>

```

31.4. ábra. Éttermi rendelést tartalmazó XML-üzenet

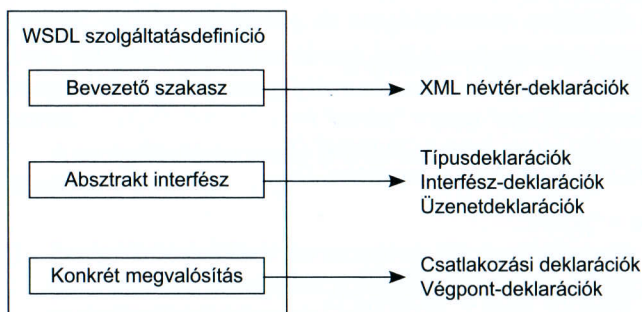
got definiál a webszolgáltatással kapcsolatban. Definiálja, hogy *mit* csinál a szolgáltatás, *hogyan* kommunikál és *hol* található meg:

1. A WSDL-dokumentum „mit” része, azaz az interfész, specifikálja a szolgáltatás által támogatott műveleteket, valamint definiálja a szolgáltatásnak küldhető és a szolgáltatás által elfogadott üzenetek formátumát.
2. A WSDL-dokumentum „hogyan” része, melyet csatlakozásnak is neveznek, leképezi az interfészt egy konkrét protokollhalmazra. A csatlakozás technikai részleteket tartalmaz arra nézve, hogyan kell kommunikálni a webszolgáltatással.
3. A WSDL-dokumentum „hol” része, amit (zavaróan) szolgáltatásnak neveznek, leírja, hol található meg egy specifikus webszolgáltatás implementációja.

A WSDL fogalmi modellje (31.5. ábra) a szolgáltatásleírás minden részét tartalmazza. Ezek mindegyike XML nyelven van kifejezve, és akár különböző állományokban is elhelyezkedhetnek. Ezek a részek a következők:

1. Egy bevezető rész, amely általában definiálja a használt XML-névtereket, és amely opcionálisan tartalmazhat egy dokumentációs szakaszt is, amelyben további információk adhatók a szolgáltatásról.
2. Opcionális leírás a szolgáltatás által küldött üzenetekben használt típusokról.
3. A szolgáltatás interfészének leírása, azaz az általa biztosított műveletek leírása.
4. Azon üzenetek leírása, amelyeket a szolgáltatás küld, illetve elfogad.
5. A szolgáltatás által használt csatlakozás, azaz az üzenetek küldésére és fogadására használt protokoll leírása. A SOAP az alapértelmezett, de más csatlakozások is specifikálhatók. A csatlakozás meghatározza, hogyan kell az üzeneteket egyetlen üzenetbe csomagolni, valamint specifikálja a kommunikációs protokollokat. A csatlakozás azt is meghatározza, hogy a támogató információk, mint a biztonsági tanúsítványok és a tranzakcióazonosítók, hogyan jelennek meg az üzenetekben.

6. A szolgáltatás fizikai helyét meghatározó, URI-ként kifejezett végpont-specifikáció – az erőforrás címe, amely az interneten keresztül elérhető.



31.5. ábra. Egy WSDL-specifikáció modellje

Az XML-ben íródott teljes szolgáltatásleírások hosszú, részletes és fárasztó olvasmányok. Általában tartalmaznak XML névtér-definíciókat. A névterekkel a neveket minősítik. A névtér-azonosító bármilyen azonosító előtt szerepelhet az XML-leírásban. Ez azt jelenti, hogy meg tudjuk különböztetni az azonos névvel az XML-leírás különböző részein definiált azonosítókat. Nem bocsátkozom részletekbe a névterekkel illetően. Jelen fejezet megértéséhez elég annyit tudnunk, hogy a nevek előtt szerepelhet névtér-azonosító, és hogy a névtér:név párnak egyedinek kell lennie.

A könyv weblapján elhelyeztem egy teljes szolgáltatásleírást. Azonban ez nagyon hosszú, így most csak az absztrakt interfész definíciójára koncentrálok. A WSDL ezen része megfelel a szoftverkomponensek „biztosított” interfészének. A 31.6. ábra egy egyszerű szolgáltatás interfészének részleteit tartalmazza. A szolgáltatás, ha adott egy dátum és egy hely (város vagy ország), visszaadja az adott napon az adott helyen mért legmagasabb és legalacsonyabb hőmérsékletet. Ezek a hőmérsékletek, attól függően, hogy hol mérték őket, Celsiusban vagy Fahrenheitben értendők.

A 31.6. ábrán szereplő definíció első része a szolgáltatásspecifikációban használt típusok és elemek definícióit tartalmazza. Ez definiálja a `PlaceAndDate`, `MaxMinTemp` és `InDataFault` elemeket. Itt csak a `PlaceAndDate` elem definícióját szerepeltettem. Ezt úgy képzelhetjük el, mint egy rekordot város, ország és dátum mezőkkel. A `MaxMinTemp` és az `InDataFault` definíciója hasonlóképpen történhet.

A leírás második része a szolgáltatás interfészének definícióját mutatja be. Ebben a példában a `weatherInfo` szolgáltatásnak egyetlen művelete van, azonban nem létezik semmiféle korlátozás a megengedett műveletek számát illetően. A `weatherInfo` művelethez egy in-out minta tartozik, ami azt jelenti, hogy egy bemeneti üzenetet vár, és egy kimeneti üzenetet generál. A WSDL 2.0 specifikáció több különböző üzenetcsere mintát engedélyez. Ezek: in-only, in-out, out-only, in-optional-out, out-in stb. Ezután a típus szekcióban definiált típusokat használó üzeneteket definiáljuk.

Néhány felhasznált típus definiálása. Feltételezzük, hogy a „ws” névtér-előtag az XML-séma URI névtérét hivatkozza, és hogy ezen definícióhoz rendelt névtér-előtag a weathns.

```
<types>
  <xs: schema targetNamespace = "http://.../weathns"
    xmlns: weathns = "http://.../weathns" >
    <xs:element name = "PlaceAndDate" type = "pdrec" />
    <xs:element name = "MaxMinTemp" type = "mmtrec" />
    <xs: element name = "InDataFault" type = "errmess" />

    <xs: complexType name = "pdrec"
      <xs: sequence>
        <xs:element name = "town" type = "xs:string"/>
        <xs:element name = "country" type = "xs:string"/>
        <xs:element name = "day" type = "xs:date" />
      </xs:complexType>

```

MaxMinType és InDataFault definíciója

```
</schema>
</types>
```

Definiáljuk az interfészt és műveleteit. Ebben az esetben egyetlen művelet van, ami a maximális és minimális hőmérsékletet adja vissza.

```
<interface name = "weatherInfo">
  <operation name = "getMaxMinTemps" pattern = "wsdl:ns: in-out">
    <input messageLabel = "In" element = "weathns: PlaceAndDate" />
    <output messageLabel = "Out" element = "weathns:MaxMinTemp" />
    <outfault messageLabel = "Out" element = "weathns:InDataFault" />
  </operation>
</interface>
```

31.6. ábra. Részlet egy webszolgáltatás WSDL-leírásából

A WSDL legnagyobb problémája az, hogy a szolgáltatás-interfész definíciója nem tartalmaz semmiféle információt a szolgáltatás szemantikáját és nemfunkcionális tulajdonságait (teljesítmény, megbízhatóság stb.) illetően. Csak a szolgáltatás szignatúráját írja le, annak eldöntése pedig, hogy valójában mi a szolgáltatás feladata, és hogy az üzenetek egyes mezői mit jelentenek, a felhasználóra marad. Ezen segíthetnek a beszédes nevek és a szolgáltatás dokumentációja, azonban így is gyakran előfordulnak félreértések és az ezekből eredő téves használat.

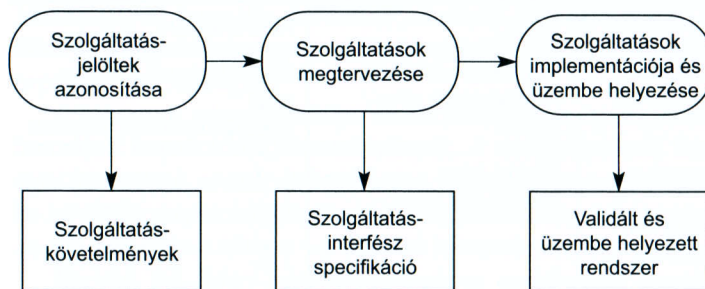
31.2. SZOLGÁLTATÁSOK TERVEZÉSE

A szolgáltatástervezés az a folyamat, melynek során szolgáltatásorientált alkalmazásokban újrafelhasználható szolgáltatásokat fejlesztünk ki. Sok mindenben hasonlít a komponenstervezéshez. A szolgáltatások tervezőinek biztosítaniuk

kell, hogy a szolgáltatás egy olyan újrafelhasználható absztrakciót reprezentál, amely különböző rendszerek számára is hasznos lehet. Az absztrakcióhoz kapcsolódó olyan általánosan használható funkcionalitást kell tervezniük és fejleszteniük, amely robusztus, és megbízhatóan működik a különféle alkalmazásokban. Továbbá dokumentálniuk kell a szolgáltatást, hogy a lehetséges felhasználók megtalálhassák és megérthessék azt. A következőkben az állomásokat vesszük sorra.

A szolgáltatástervezés folyamatának három logikai állomása van (31.7. ábra). Ezek:

1. Szolgáltatásjelöltek azonosítása. Itt megkeressük azokat a lehetséges szolgáltatásokat, amelyeket a későbbiekben implementálhatunk, és definiáljuk a szolgáltatások követelményeit.
2. Szolgáltatások megtervezése. Ebben a szakaszban tervezzük meg a logikai és a WSDL szolgáltatás-interfészeket.
3. Szolgáltatások implementációja és üzembe helyezése. Itt implementáljuk és teszteljük a szolgáltatásokat, majd a felhasználók számára hozzáférhetővé tesszük őket.



31.7. ábra. A szolgáltatástervezés folyamata

31.2.1. Szolgáltatásjelöltek azonosítása

A szolgáltatásorientált számítástechnika alapelve, hogy a szolgáltatásoknak az üzleti logikát kell támogatniuk. Minden szervezet rengeteg folyamatot működtet, így az implementálható szolgáltatások száma is igen nagy. A szolgáltatásjelöltek azonosítási folyamatának feladata, hogy megértse és elemezze a szervezet üzleti folyamatait, majd eldöntse, milyen szolgáltatások szükségesek a folyamatok támogatásához.

Erl a szolgáltatások három alapvető típusát különbözteti meg:

1. *Segédszolgáltatások.* Ezek olyan általános funkcionalitást implementálnak, amit különböző üzleti folyamatok használhatnak. Segédszolgáltatásra példa egy

- pénznemátváltó szolgáltatás, amelynek feladata egy pénznem (például dollár) átváltása egy másikra (például euró).
2. *Üzleti szolgáltatások.* Ezek a szolgáltatások egy bizonyos üzleti folyamathoz kapcsolódnak. Egy egyetemen üzleti szolgáltatás lehetne például a hallgatók regisztrálása egy kurzusra.
 3. *Koordinációs vagy folyamatszolgáltatások.* Ezek a szolgáltatások sokkal általánosabb üzleti folyamatot támogatnak, amelyben több különböző aktor és tevékenység is megjelenhet. Koordinációs szolgáltatásra jó példa egy vállalat rendelési szolgáltatása, amelyben az ügyfelek leadhatják rendeléseiket, fizethetnek stb.

Erl emellett azt is felveti, hogy a szolgáltatások lehetnek feladatorientáltak vagy entitásorientáltak. A feladatorientált szolgáltatások valamilyen tevékenységhez kapcsolódnak, míg az entitásorientáltak olyanok, mint az objektumok – néhány üzleti entitáshoz köthetők, mint egy munkaúrlap. A 31.8. ábra feladat- és entitásorientált szolgáltatások példáit tartalmazza. A segéd- és üzleti szolgáltatások mindkét típushoz tartozhatnak, a koordinációs szolgáltatások mindig feladatorientáltak.

	Segéd	Üzleti	Koordinációs
Feladat	Pénznemátváltás Alkalmazott keresés	Igénylőlap validálása Hitelbesorolás ellenőrzése	Kiadási igények feldolgozása Külső szolgáltatók kifizetése
Entitás	Dokumentumstílus-ellenőrző Webes űrlapok konvertálása XML-be	Kiadási űrlap Diákfelvételi űrlap	

31.8. ábra. Szolgáltatások osztályozása

A szolgáltatásjelöltek azonosítása során célunk olyan szolgáltatások kiválasztása, amelyek logikai egységet alkotnak, függetlenek és újrafelhasználhatók. Ebben Erl osztályozása hasznos segítség lehet, ugyanis útmutatást ad ahhoz, hogyan fedezzük fel az újrafelhasználható szolgáltatásokat az üzleti entítások és üzleti tevékenységek megvizsgálásával. Azonban a jelöltek kiválasztása még így is legalább annyira bonyolult, mint az objektumok és komponensek azonosításának folyamata. Először ki kell gondolnunk néhány lehetséges jelöltet, majd számos kérdést kell feltennünk velük kapcsolatban, hogy megtudjuk, valóban hasznosak-e. Az újrafelhasználható komponensek felismerését az alábbiakhoz hasonló kérdések segíthetik:

1. Entitásorientált szolgáltatás esetén megkérdezhetjük a következőket: a szolgáltatáshoz egyetlen logikai entitás tartozik, amit különböző üzleti folyamatokban használnak? Milyen műveletek hajthatók végre helyes működés esetén az entitáson, amelyeket támogatni kell?

2. Feladatorientált szolgáltatás esetén: a feladatot a szervezet több, különböző dolgozója vetette fel? Képesek lesznek beletörődni az elkerülhetetlen szabványosításba, amely a közös támogató szolgáltatás megvalósításának velejárója?
3. Független a szolgáltatás? Másképp: milyen mértékben támaszkodik más szolgáltatások elérhetőségére?
4. Szüksége van a szolgáltatás műveleteinek állapotok kezelésére? Ha igen, az állapotok kezeléséhez adatbázist használ? Általánosságban azok a szolgáltatások, amelyek belső állapotokra támaszkodnak, kevésbé újrafelhasználhatók, mint azok, amelyek állapotaikat külsőleg tárolják.
5. A szolgáltatás használható vállalaton kívüli kliensekről is? Például egy katalógus böngészésére szolgáló entitásorientált szolgáltatás elérhető-e kívülről és belülről is?
6. Lehetséges, hogy a szolgáltatást igénybe vevő különböző felhasználóknak eltérő nemfunkcionális követelményeik vannak? Ha igen, akkor talán érdemes a szolgáltatás több verzióját implementálni.

A fenti kérdésekre kapott válaszok segíthetnek a szolgáltatásként implementálandó absztrakciók megkeresésében és finomításában. Mindezek ellenére nem létezik olyan formális módszer, amellyel eldönthetnénk, melyek a legmegfelelőbb szolgáltatások, így a szolgáltatásjelöltek kiválasztásához szakképzettség és tapasztalat szükséges.

A folyamat végeredménye az azonosított szolgáltatások halmaza, valamint a hozzájuk kapcsolódó követelmények. A szolgáltatások funkcionális követelményei hivatottak annak definiálására, mi a szolgáltatás feladata. A nemfunkcionális követelmények közé tartoznak többek között a biztonságra, a teljesítményre és az rendelkezésre állásra vonatkozó követelmények.

Tegyük fel, hogy egy számítógépes eszközöket árusító nagyvállalat speciális árakat rendelt az ügyfelek számára jóváhagyott konfigurációkhoz. Az automatikus rendelés leegyszerűsítésére a cég olyan katalógusszolgáltatást kíván előállítani, amely lehetővé teszi az ügyfelek számára, hogy kiválasszák a nekik szükséges eszközöket. Azonban a szokásos ügyfélkatalógusoktól eltérően a rendeléseket nem közvetlenül, egy katalógus-interfészen keresztül kell leadni, hanem az egyes cégek beszerzési rendszerén keresztül. A legtöbb vállalat saját költségvetéssel rendelkezik, és saját folyamataikat kell követniük a rendelések jóváhagyásában és a rendelések feladásában.

A katalógusszolgáltatás jó példa üzleti műveleteket támogató entitásorientált szolgáltatásra. A katalógusszolgáltatás funkcionális követelményei az alábbiak:

1. A szolgáltatást használó minden vállalat számára egy specifikus katalógust kell nyújtani. Ez tartalmazhatja az adott vállalat dolgozói által megrendelhető eszközöket és konfigurációkat, valamint a katalógus tételeinek korábban egyeztetett árait.
2. A dolgozóknak lehetővé kell tenni a katalógus egy változatának letöltését offline böngészésre.

3. A katalógusnak biztosítania kell, hogy a felhasználók összehasonlíthassák akár hat katalóguselem specifikációját és árát.
4. A katalógusban lehetővé kell tenni, hogy a felhasználók böngészhessék azt, és kereshessenek benne.
5. A felhasználóknak képesnek kell lenniük arra, hogy megtekintsék néhány katalógusbeli tétel kézbesítésének várható dátumát.
6. A felhasználók számára lehetővé kell tenni, hogy „virtuális rendeléseket” adhassanak le. Virtuális rendelés esetén az igényelt tételeket 48 órára félreteszik. A virtuális rendelést egy beszerzési rendszerben elhelyezett valódi rendeléssel meg kell erősíteni. Ennek a virtuális rendelést követő 48 órában meg kell történnie.

A fenti funkcionális követelmények mellett számos nemfunkcionális követelmény is felmerül a katalógussal kapcsolatban:

1. A katalógusszolgáltatást csak az akkreditált szervezetek dolgozói érhetik el.
2. A szervezetek számára nyújtott árakat bizalmasan kell kezelni, azt más szervezetek dolgozói nem láthatják.
3. A katalógusnak elérhetőnek kell lennie megszakítás nélkül reggel 7 óra és 11 óra között, greenwichi idő szerint.
4. A szolgáltatásnak képesnek kell lennie másodpercenként akár 10 kérés feldolgozására is.

Vegyük észre, hogy a katalógusszolgáltatás válaszüdejével kapcsolatban nem említettünk semmilyen nemfunkcionális követelményt. A válaszüdő a katalógus méretétől és az egyszerre jelen lévő felhasználók becsült számától függ. Mivel ez nem időkritikus szolgáltatás, a tervezés jelen fázisában nincs szükségünk ennek specifikálásra.

31.2.2. Szolgáltatás-interfész tervezése

Ha a szolgáltatásjelölteket kiválasztottuk, a szolgáltatástervezés folyamatának következő lépése a szolgáltatás-interfészek megtervezése. Ehhez hozzátartozik a szolgáltatáshoz kapcsolódó műveletek és paramétereik definiálása. Azt is alaposan át kell gondolnunk, hogyan tervezhetők meg az üzenetek és műveletek úgy, hogy minimalizáljuk a szolgáltatáskérések teljesítéséhez szükséges üzenetcsere számát. Ahelyett, hogy szinkron interakciókat követelnénk meg a szolgáltatástól, inkább azt kell biztosítanunk, hogy a lehető legtöbb információt küldjük el minden egyes üzenetben.

Arra is emlékeznünk kell, hogy a szolgáltatások állapot nélküliek, így a szolgáltatásspecifikus alkalmazások állapotainak kezelése a felhasználó felelőssége, nem magáé a szolgáltatásé. Vagyis az ezeket az állapotokat leíró információkat is el kell helyezni a szolgáltatás bemenő és kimenő üzeneteiben.

A szolgáltatás-interfész tervezésének három lépése a következő:

1. Az interfész logikai tervezése során azonosítjuk a szolgáltatásokhoz tartozó műveleteket, meghatározzuk a műveletek bemeneteit és kimeneteit, továbbá a műveletekhez tartozó kivételeket.
2. Az üzenetek megtervezése az a lépés, amelyben eldől, milyen lesz az üzenetek struktúrája, és hogy a szolgáltatás milyen üzeneteket küld és fogad.
3. A WSDL-fejlesztés során a logikai és üzenetterveinket átültetjük egy WSDL nyelven íródott absztrakt interfészleírásba.

Az első lépés, a logikai interfész tervezése, a szolgáltatás követelményeiből indul ki, és ezek alapján definiálja a szolgáltatáshoz kapcsolódó műveletek és paraméterek neveit. Szintén itt kell definiálnunk azokat a kivételeket, amelyek bekövetkezhetnek a szolgáltatás műveleteinek meghívása közben. A 31.9. és a 31.10. ábrák tartalmazzák azokat a műveleteket, amelyek implementálják a katalógus követelményeit, valamint itt szerepelnek a műveletek bemenetei, kimenetei és kivételei. Ebben a lépésben nem szükséges ezeket részletesebben definiálni – a részletek a tervezési folyamat következő szakaszában jelennek meg.

Különösen fontos definiálni a kivételeket, valamint azt, hogy a felhasználók hogyan szereznek tudomást a kivételekről. A szolgáltatások tervezői nem ismerik szolgáltatásuk jövőbeli felhasználásának követelményeit, azt pedig, hogy a felhasználók teljesen érteni fogják a szolgáltatást, értelmetlen feltételezni. Előfordulhat, hogy helytelen bemenő üzenet érkezik, így olyan kivételeket kell definiálnunk, amelyek értesítik erről a felhasználót. Az újrafelhasználható komponensek fejlesztésénél bevett gyakorlat, hogy a kivételek kezelését teljes mértékben a komponens használójára hagyják – a fejlesztőnek nem kell a felhasználóra kényszeríteni saját elképzeléseit arról, hogyan kell a kivételeket kezelni.

Művelet	Leírás
MakeCatalogue	Egy adott ügyfél számára személyreszabott katalógust készít. Tartalmaz egy opcionális paramétert, amely segítségével a katalógus PDF-formátumban is elkészíthető.
Compare	Legfeljebb négy katalóguselem legfeljebb hat tulajdonságának (például ár, dimenziók, processzorsebesség stb.) összehasonlítására szolgál.
Lookup	Megjeleníti az egy adott elemre vonatkozó összes információt.
Search	Ez a művelet egy megadott logikai kifejezésnek megfelelő bejegyzéseket keres a katalógusban. Kilistázza az összes olyan tételt, amely eleget tesz a logikai kifejezéssel megadott feltételnek.
CheckDelivery	Visszatér azzal a dátummal, amelyre a tétel kézbesítésre kerülne, ha aznap megrendelnénk.
MakeVirtualOrder	Lefoglal annyi tételt, amennyit az ügyfél megrendelt, és információkat szolgáltat azokról az ügyfél beszerzési rendszerének.

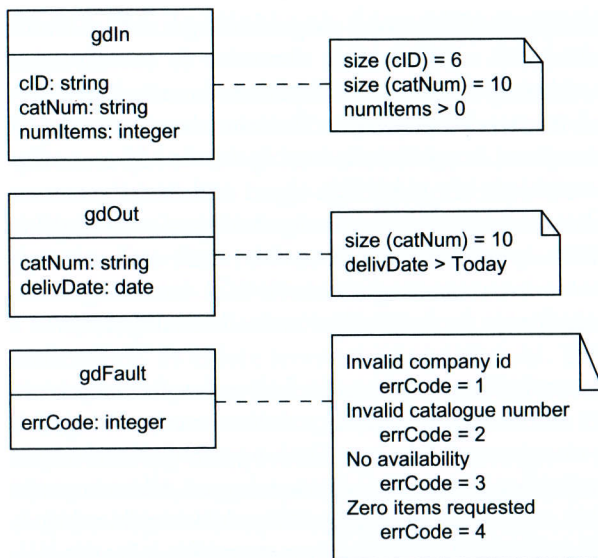
31.9. ábra. A katalógusszolgáltatás műveleteinek funkcionális leírása

Művelet	Bemenetek	Kimenetek	Kivételek
MakeCatalogue	<i>mcIn</i> Cégazonosító PDF-jelző	<i>mcOut</i> A vállalat katalógusának URL-je	<i>mcFault</i> Érvénytelen cégazonosító
Compare	<i>compIn</i> Cégazonosító Jellemzők (legfeljebb 6) Katalógusszám (legfeljebb 4)	<i>compOut</i> Az összehasonlítási táblát tartalmazó oldal URL-je	<i>compFault</i> Érvénytelen cégazonosító Érvénytelen katalógusszám Ismeretlen jellemző
Lookup	<i>lookIn</i> Cégazonosító Katalógusszám	<i>lookOut</i> A tétel információit tartalmazó oldal URL-je	<i>lookFault</i> Érvénytelen cégazonosító Érvénytelen katalógusszám
Search	<i>searchIn</i> Cégazonosító Keresési sztring	<i>searchOut</i> A keresési eredményeket tartalmazó oldal URL-je	<i>searchFault</i> Érvénytelen cégazonosító Rosszul formált keresési sztring
CheckDelivery	<i>gdIn</i> Cégazonosító Katalógusszám A kért tételek száma	<i>gdOut</i> Katalógusszám Várható kézbesítés dátuma	<i>gdFault</i> Érvénytelen cégazonosító Érvénytelen katalógusszám Nem állnak rendelkezésre elemek A kért elemek száma nulla
PlaceOrder	<i>poIn</i> Cégazonosító A kért elemek száma Katalógusszám	<i>poOut</i> Katalógusszám A kért elemek száma Várható kézbesítési dátum Egységár becslése Teljes költség becslése	<i>poFault</i> Érvénytelen cégazonosító Érvénytelen katalógusszám Az igényelt tételek száma nulla

31.10. ábra. A katalógus interfészterve

Ha rendelkezésünkre áll egy nem hivatalos logikai leírás a szolgáltatás feladatairól, a következő lépés a be- és kimenő üzenetek struktúrájának meghatározása, és az üzenetekben használt típusok definiálása. Az XML alkalmatlan ennek leírására. Jobb ötletnek látszik az üzeneteket objektumokként reprezentálni, és vagy UML-ben, vagy valamilyen programozási nyelven (ami lehet például a Java) definiálni. A 31.11. ábrán egy UML-diagram látható, amely a katalógusszolgáltatás *getDelivery* műveletének be- és kimenő üzeneteit mutatja be.

Figyeljük meg, hogy az UML-diagramot korlátozásokat tartalmazó megjegyzésekkel ellátva, részleteket adtam a leíráshoz. Ezek definiálják a vállalatot és a katalógustételeket reprezentáló sztringek hosszát, specifikálják, hogy a tételek száma nem lehet nulla, valamint azt, hogy a kézbesítés dátumának a mai napnál



31.11. ábra. Bemenő és kimenő üzenetek UML-definíciója

későbbre kell esnie. A megjegyzésekből látszik az is, hogy milyen hibákhoz milyen hibakód tartozik.

A szolgáltatástervezés folyamatának végső lépése a szolgáltatás-interfész terv átültetése WSDL-be. Ahogyan korábban tárgyaltam, a WSDL-reprezentáció hosszú és részletes, így ebben a szakaszban könnyen hibázhatunk. A legtöbb szolgáltatásorientált fejlesztést támogató környezet (például az ECLIPSE környezet) tartalmaz olyan eszközöket, amelyek a logikai interfészleírást lefordítják a megfelelő WSDL-reprezentációba.

31.2.3. Szolgáltatások implementációja és üzembe helyezése

Miután definiáltuk a szolgáltatásjelölteket és megterveztük azok interfészét, a szolgáltatástervezés folyamatából már csak a szolgáltatások implementációja van hátra. A megvalósításban olyan szabványos programozási nyelvek vehetnek részt, mint a C# és a Java. Mindkét nyelv tartalmaz könyvtárakat a szolgáltatás-fejlesztés széles körű támogatására.

Emellett a szolgáltatások kifejleszthetők már létező komponensek, vagy ahogy később tárgyalom, ősrendszerek felhasználásával. Ez azt jelenti, hogy az eddig felhalmozott hasznos szoftvervagyon sokkal szélesebb körben tehető hozzáférhetővé. Az ősrendszerek esetén ez azt is jelentheti, hogy a rendszer-funkcionalitáshoz hozzáférhetnek az új alkalmazások is. Új szolgáltatások definiálhatók létező szolgáltatások kompozíciójával is. A szolgáltatások kompozíciójával történő fejlesztése a 31.3. fejezet témája.

Az implementáció után, az üzembe helyezést megelőzően, a szolgáltatást tesztelnünk kell. Ehhez hozzátartozik a bemenetek elemzése és osztályozása (23. fejezet), a kombinációkat tükröző üzenetek elkészítése, és annak ellenőrzése, hogy a megfelelő kimenő üzenetek érkeztek-e. Érdeemes megpróbálnunk kivételeket generálni minden tesztben, hogy lássuk, a szolgáltatás képes-e megbirkózni az érvénytelen bemenetekkel. Ma már több olyan eszköz is beszerezhető, amely alkalmas a szolgáltatások vizsgálatára és tesztelésére, valamint képes teszteseteket generálni WSDL-specifikációból. Azonban ezek csak azt tudják ellenőrizni, hogy a szolgáltatás-interfész megfelel-e a WSDL-leírásnak. Nem tudják tesztelni azt, hogy a szolgáltatás funkcionális viselkedése eleget tesz-e a specifikációjának.

A folyamat befejező lépése a szolgáltatások üzembe helyezése. Ez azt jelenti, hogy a szolgáltatást használatra elérhetővé tesszük egy webszerveren. A legtöbb szerverszoftver esetén ez nagyon egyszerűen megy. Csak egy megadott könyvtárba kell telepítenünk a végrehajtható szolgáltatást tartalmazó állományt, és az automatikusan használhatóvá válik. Ha a szolgáltatást publikusnak szánjuk, akkor még egy UDDI-leírást is készítenünk kell, hogy a potenciális felhasználók megtalálhassák azt. Erl (Erl, 2004) könyve hasznos összefoglalója az UDDI-nak.

Ma számos nyilvános tároló áll rendelkezésünkre, és a cégek fenntarthatják saját belső UDDI tárolóikat. Egy UDDI-leírás különböző típusú információkat tartalmaz:

1. Részleteket a szolgáltatás gyártójáról. Ez főként bizalmi szempontokból lényeges. Egy szolgáltatás felhasználóinak garancia kell arról, hogy a szolgáltatás megfelelően fog működni. A szolgáltatás gyártójáról szóló információk lehetővé teszik, hogy a felhasználók ellenőrizzék annak tanúsítványait.
2. A szolgáltatás funkcionalitásának informális leírása. Ez abban nyújt segítséget a lehetséges felhasználóknak, hogy eldöntsék, erre van-e szükségük. Azonban mivel ez a leírás természetes nyelven íródik, ezért nem lehet egyértelmű leírása a szolgáltatás feladatának.
3. Információ arról, hol található meg a szolgáltatás WSDL-specifikációja.
4. Feliratkozási információk, amelyek segítségével a felhasználók regisztrálhatnak a szolgáltatás frissítésével kapcsolatos információkra.

Az UDDI egyik lehetséges problémája az, hogy a szolgáltatás funkcionális viselkedését informálisan, természetes nyelven írja le. Ahogy azt a szoftverkövetelményekről szóló 6. fejezetben említettem, a természetes nyelvű szövegek ugyan könnyen olvashatók, de ugyanilyen egyszerűen félreértelmezés tárgyává válhatnak. Széles körű kutatás folyik a probléma megoldására. A szemantika specifikációjának legígéretesebb megközelítése az ontológiaalapú leírásra épül, ahol a kifejezések specifikus jelentése egy ontológiában található meg. A web-szolgáltatás-ontológiák leírására az OWL-S nyelvet fejlesztették ki (OWL_Services_Coalition, 2003). A könyv írásakor ezek a technikák még igen fejletlenek, de a következő néhány évben széles körű elterjedésük várható.

31.2.4. Szolgáltatások ösrendszerekből

A 18. fejezetben bemutattam, hogyan implementálhatunk úgy komponenseket, hogy interfészeket készítsünk létező ösrendszerekhez. Így újrafelhasználhatóvá válhat az ösrendszerek funkcionalitása. A komponens implementációja mindössze annyit jelentett, hogy egy általános interfészt készítettünk a rendszerhez. A szolgáltatások egyik legfontosabb felhasználási területe az ilyen „csomagolók” gyártása ösrendszerekhez. Ezek a rendszerek azután elérhetők weben keresztül, és integrálhatók más alkalmazásokkal.

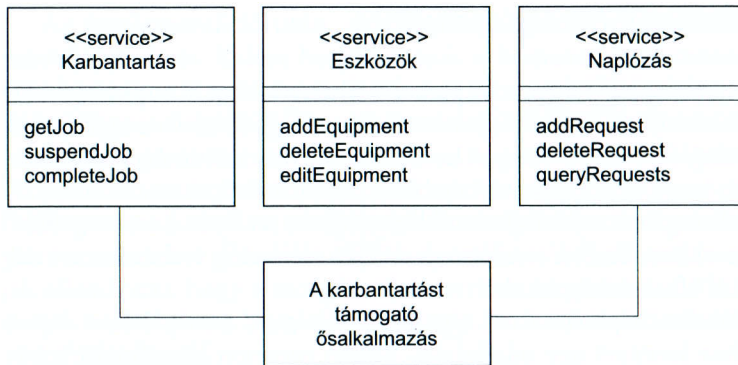
Ennek illusztrálására képzeljünk el egy nagyvállalatot, amely leltárt vezet eszközeiről, és ehhez fenntart egy adatbázist. Ebben nyomon követhetők a következők: milyen nyilvántartási kérelmek érkeztek az egyes eszközökre, milyen szokásos karbantartásokat ütemeztek be, mikor történtek meg ezek a karbantartások, mennyi időbe került ez a karbantartás stb. Az ösrendszert eredetileg arra fejlesztették ki, hogy napi feladatlistákat generáljon a karbantartó személyzetnek, de idővel újabb és újabb lehetőségeket adtak a rendszerhez. Ezek információkat szolgáltatnak az egyes eszközök karbantartási költségeiről, és segítenek megbecsülni a külső vállalkozók által végrehajtandó munkálatok költségeit. A rendszer kliens–server rendszerként fut, eléréséhez egy speciális célú kliens PC-s szoftver használható.

A cég most valós idejű hozzáférést szeretne biztosítani a karbantartó személyzet által használt hordozható terminálokról. Ők fogják közvetlenül betölteni az adatbázisba az egyes eszközök karbantartására fordított idő és egyéb erőforrás mennyiségét, valamint lekérdezhetik, mi a következő karbantartási munka, amit el kell végezniük. Emellett a telefonos ügyfélszolgálati személyzetnek hozzá kell férnie a rendszerhez, hogy naplózhassa a karbantartási kérelmeket és ellenőrizze állapotukat.

Azonban a gyakorlatban lehetetlen a rendszert úgy továbbfejleszteni, hogy támogassa a fenti követelményeket, így a vállalat úgy dönt, hogy új alkalmazásokat készít a karbantartó és a telefonos ügyfélszolgálati személyzet számára. Ezek az alkalmazások az ösrendszerre támaszkodnak, amely számos szolgáltatás alapjául fog szolgálni. Ez látható a 31.12. ábrán, ahol egy UML-sztereotípiát használtam a szolgáltatások jelölésére. Az új alkalmazások egyszerűen üzeneteket cserélnek ezekkel a szolgáltatásokkal, hogy elérjék az ösrendszer funkcionalitását.

A biztosított szolgáltatások közül néhány:

1. *Karbantartási szolgáltatás.* Műveletei lehetővé teszik, hogy feladatazonosító alapján lekérdezzünk egy feladatot, prioritást vagy földrajzi elhelyezkedést, illetve hogy egy végrehajtott karbantartás részleteit betöltsük az adatbázisba. Emellett támogat egy olyan műveletet, amellyel az elkezdett, de még be nem fejezett munkák felfüggeszthetők.
2. *Eszközszoftvertartás.* Olyan műveletei vannak, amellyel hozzáadhatunk vagy törölhetünk egy eszközt, és módosíthatjuk az egyes eszközökről tárolt információkat.



31.12. ábra. Az ősszkalmazás elérését biztosító szolgáltatások

3. *Naplózó szolgáltatás.* Műveletei segítségével új szolgáltatási kérelem adható le, karbantartási igény törölhető, továbbá lehetőséget ad a végrehajtás alatt álló munkák státuszának lekérdezésére.

A létező ősszkalmazást nem egyetlen szolgáltatás reprezentálja. Ehelyett a kifejlesztett szolgáltatások koherensek, és a funkcionalitás egyetlen területét támogatják. Ennek köszönhetően kevésbé összetettek, átláthatóbbak és egyszerűen újrafelhasználhatók más alkalmazásokban. Mivel a szolgáltatások üzeneteinek részletes leírása igen helyigényes, így ez az olvasó feladata marad.

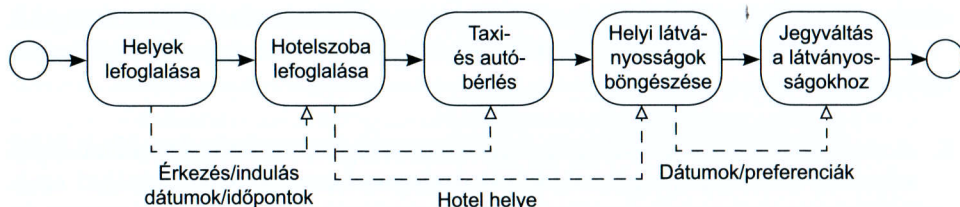
31.3. SZOFTVERFEJLESZTÉS SZOLGÁLTATÁSOKKAL

A szolgáltatásokat használó szoftverek fejlesztése arra az ötletre épül, hogy egyszerű szolgáltatásokat összeállítunk, majd konfiguráljuk azokat, és így egy összetettebb szolgáltatást hozunk létre. Ha webes alkalmazásra van szükségünk, integrálhatjuk ezeket egy webes felhasználói felülettel, vagy felhasználhatjuk őket más összetettebb szolgáltatások megalkotásához. A kompozícióban részt vevő szolgáltatások lehetnek olyanok, hogy speciálisan az adott alkalmazáshoz készültek, lehetnek a vállalatban belül kifejlesztett üzleti szolgáltatások, és lehetnek külső gyártók szolgáltatásai.

Sok cég foglalkozik ma azzal, hogy vállalatok alkalmazásait szolgáltatásorientált rendszerekre alakítja át. Ez megnyitja az utat a vállalatban belüli szélesebb körű használat előtt. A következő lépés a szervezetek közötti alkalmazások fejlesztése lesz, amelynek résztvevői megbízható szolgáltatók lesznek. A szolgáltatásorientált architektúrák víziójának végső kiteljesedése a „szolgáltatás piac” fejlődésére fog támaszkodni. Úgy vélem, valószínűtlen, hogy ezt a jelen könyv meg fogja élni, ugyanis írásának idején relatíve kevés nyilvánosan elérhető üzleti alkalmazás van.

Szolgáltatások kompozíciójával különálló üzleti folyamatokat integrálhatunk, hogy olyan folyamatot állítsunk elő, amely sokkal bővebb funkcionalitást kínál. Tegyük fel, hogy egy légitársaság komplett nyaralási csomagot akar az utasok rendelkezésére bocsátani. A repülőgépjáratok lefoglalása mellett az utasoknak lehetőségük van hotelszobák és taxik lefoglalására, autók bérletére, böngészhetik az utazási segédanyagokat, valamint belépőket foglalhatnak a helyi látnivalókhoz. Az alkalmazás elkészítéséhez a légitársaság felhasználja saját járatfoglalási szolgáltatását, továbbá egy hotelfoglalással foglalkozó iroda, különböző autókölcsönzők és taxis cégek szolgáltatásait, és a helyi látványosságokkal foglalkozó cégek által nyújtott szolgáltatásokat. A végeredmény egy olyan szolgáltatás, amely integrálja a különböző gyártók szolgáltatásait.

Ezt a folyamatot különálló lépések sorozataként képzelhetjük el. Ez látszik a 31.13. ábrán. Minden lépésből információ áramlik a következőbe – például az autókölcsönző-szolgáltatással tudatják, mikorra várható a járat érkezése. A lépések sorozatát munkafolyamatnak nevezik – olyan tevékenységek időben rendezett sorozata, amelyek közül mindegyik a munka egy részét végzi el. A munkafolyamatot tekinthetjük egy üzleti folyamat modelljeként – minden lépésben néhány olyan cél valósul meg, amely fontos az üzlet számára. Jelen esetben az üzleti folyamat a légitársaság által kínált vakációs szolgáltatás.



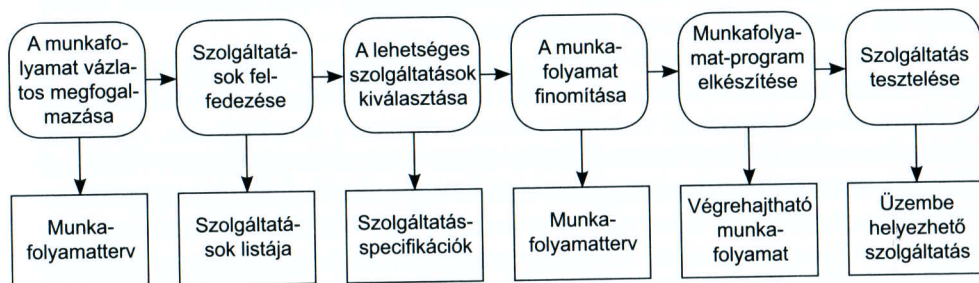
31.13. ábra. A vakáció csomaghoz tartozó munkafolyamat

A munkafolyamat egy egyszerű fogalom, és a vakáció lefoglalásának fenti forgatókönyve megfelelőnek tűnik. A valóságban a szolgáltatások kompozíciója sokkal összetettebb, mint ahogy mostani egyszerű példánk alapján gondolnánk. Például figyelembe kell vennünk annak lehetőségét, hogy a szolgáltatások hibásan működnek, és fel kell készülnünk az ilyen hibák kezelésére. Gondolnunk kell az alkalmazás felhasználóinak különleges igényeire is. Például egy mozgássérült személynek autó helyett kerekesszékre lehet szüksége.

Fel kell készülnünk az olyan szituációkra, amikor meg kell változtatnunk a munkafolyamatot, mert az egyik szolgáltatás normális végrehajtásának eredményeként olyan eredmény áll elő, amely nem kompatibilis valamely más szolgáltatás végrehajtásával. Ilyen eset például az, ha egy járatot úgy foglalnak le, hogy az június 1-jén induljon, és június 7-én térjen vissza. A munkafolyamat ezután a hotelfoglalási szakaszba lép. Azonban a szálloda június 2-ig egy fontos rendezvénynek ad otthont, így minden szobája foglalt. A hotelfoglalási szolgáltatás jelenti ezt a hiányt. Ez nem hiba, nagyon sokszor előfordul. Ezután vissza kell vonnunk a repülőgép-foglalást, és a hiányról értesítenünk kell a felhasználót. Ő ezután el-

döntheti, hogy megváltoztatja a repülés dátumát, vagy esetleg másik szállodát választ. A munkafolyamatok terminológiájával élve „kompenzációs tevékenységet” alkalmazunk a már végrehajtott műveletek visszavonásához.

A szolgáltatások tervezése már létező szolgáltatások kompozíciójával alapvetően szoftvertervezési folyamat újrafelhasználással (31.14. ábra). Az újrafelhasználással történő tervezés elkerülhetetlenül magával von néhány kompromisszumot a követelményekben. A rendszer „ideális” követelményhalmazát úgy kell módosítani, hogy az figyelembe vegye a rendelkezésre álló szolgáltatásokat, amelyek a költségvetésbe beleférnek, és minőségük megfelelő.



31.14. ábra. Szolgáltatások felépítése kompozícióval

A 31.14. ábrán a kompozícióval történő szolgáltatásépítés hat alapvető lépését mutatom be:

1. *A munkafolyamat vázlatos megfogalmazása.* A szolgáltatástervezés kezdeti lépésében az összetett szolgáltatás követelményeit használjuk az „ideális” szolgáltatás megtervezéséhez. Meglehetősen absztrakt tervet kell itt készítenünk, amelyet akkor egészítünk ki további részletekkel, ha többet tudunk az elérhető szolgáltatásokról.
2. *Szolgáltatások felfedezése.* A folyamat ezen szakaszában a szolgáltatástárókban keresünk, hogy megtudjuk, mely szolgáltatások léteznek, azoknak kik a gyártói, és mik a rendelkezésre bocsátás részletei.
3. *A lehetséges szolgáltatások kiválasztása.* A felfedezett lehetséges szolgáltatásjelöltek közül kiválasztjuk azokat a szolgáltatásokat, amelyek alkalmasak a munkafolyamat egyes tevékenységeinek implementálására. Kiválasztási kritériumaink között nyilvánvalóan szerepelni fog a nyújtott szolgáltatások funkcionalitása. Szintén szem előtt tarthatjuk a szolgáltatások költségét és minőségét (válaszadási képesség, rendelkezésre állás stb.). Azonos funkcionalitású szolgáltatások közül azt választhatjuk, amelyek az ár vagy a minőség tekintetében jobban megfelel a munkafolyamatunknak.
4. *A munkafolyamat finomítása.* A kiválasztott szolgáltatásokkal kapcsolatos információk tükrében finomítjuk a munkafolyamatot. Itt adhatunk részleteket az absztrakt leíráshoz, és lehetőségünk van munkafolyamat-tevékenységek törlésére és felvételére. Ezután megismételhetjük a felfedezési és kiválasztási lépéseket. Miután kiválasztottuk a szolgáltatások egy stabil halmazát és vég-

legesítettük a munkafolyamattervet, továbbléphetünk a folyamat következő szakaszába.

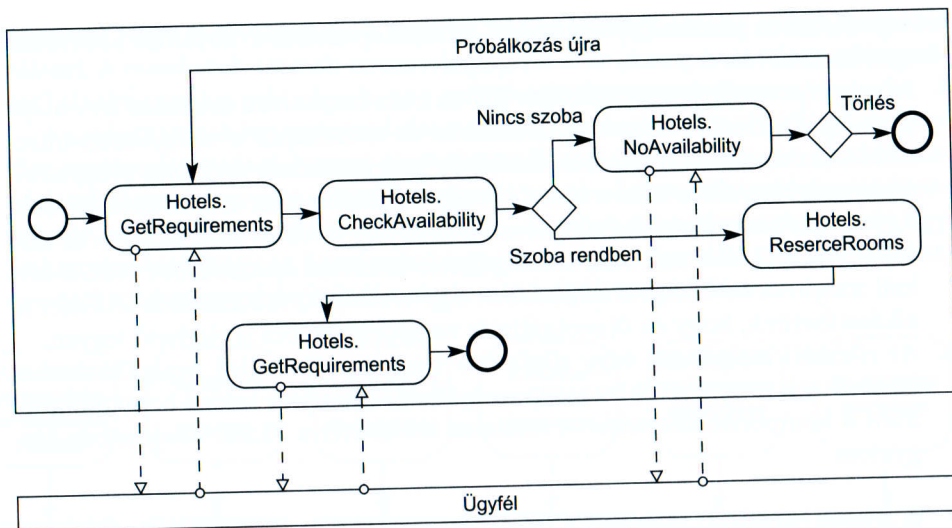
5. *Munkafolyamat-program elkészítése.* Ebben a szakaszban az absztrakt terv alapján végrehajtható programot készítünk, és definiáljuk a szolgáltatás-interfészt. A szolgáltatás implementációjához használhatunk szokásos programozási nyelveket, mint a Java és a C#, vagy használhatunk sokkal speciálisabb, úgynevezett munkafolyamatnyelvet. Ilyen nyelv a WS-BPEL. Ahogyan az előzőekben arról már szó volt, a szolgáltatás-interfész specifikációt WSDL-ben kell megírunk. Ebben a szakaszban egy webalapú felhasználói felületet is elkészíthetünk, hogy az új szolgáltatás webböngészőből is elérhető legyen.
6. *Az elkészült szolgáltatás vagy alkalmazás tesztelése.* Ha külső szolgáltatásokat használunk, az elkészült összetett szolgáltatás tesztelése sokkal bonyolultabb, mint a komponenstesztelés. A tesztelés témakörét a 31.3.2. alfejezetben tárgyalom.

A fejezet hátralévő részében a munkafolyamat-tervezésre és -tesztelésre helyezem a hangsúlyt. A bevezetőben már említettem, hogy ez idáig nem alakult ki a szolgáltatások piaca. Jóllehet létezik néhány nyilvános UDDI-tároló, azonban ezek kevés bejegyzést tartalmaznak, és a szolgáltatásleírások olykor homályosak és nem teljesek. Ez az oka, hogy a szolgáltatások felfedezése nem igazán elterjedt. A legtöbb szolgáltatást a szervezetben belül fedezik fel belső tárolókban és a szoftvertervezők nem hivatalos beszélgetésein keresztül.

31.3.1. Munkafolyamat-tervezés és -implementáció

A munkafolyamat-tervezéshez hozzátartozik a létező és a tervezett üzleti folyamatok elemzése a folyamatok különböző szakaszainak megértése céljából. Ezután a megtervezendő folyamatot valamilyen munkafolyamattervek számára készített jelöléssel reprezentáljuk. Ez a folyamat végrehajtásának szakaszait és a szakaszok között áramló információkat szemlélteti. Azonban a létező folyamatok informálisak lehetnek, és függhetnek a bennük részt vevő emberek képességeitől és gyakorlatától – azaz lehet, hogy nincs a munkának „normális” menete. Ilyen esetekben a folyamatról megszerzett tudásunk felhasználásával olyan munkafolyamatot kell terveznünk, amely ugyanazt a célt valósítja meg, mint az aktuális üzleti folyamatok.

A munkafolyamatok üzleti folyamatok modelljeit reprezentálják, és általában olyan grafikus jelölésmódokkal ábrázoljuk őket, mint a BPMN (White, 2004) vagy a YAWL (van der Aalst és ter Hofstede, 2005). A könyv írásának idején a BPMN az a folyamatmodellezési nyelv, amely a legjobb úton halad a szabványosítás felé. Ez egy könnyen érthető grafikus nyelv. Olyan leképezéseket fejlesztettek ki, amelyek alacsonyabb szintre fordítják a nyelvet, mégpedig XML-alapú WS-BPEL-beli leírásokká. A BPMN tehát jól illeszkedik a 31.2. ábrán szereplő webszolgáltatás-szabványokhoz. Itt a BPMN jelölésrendszerét használom az üzleti folyamat programozásának szemléltetésére.



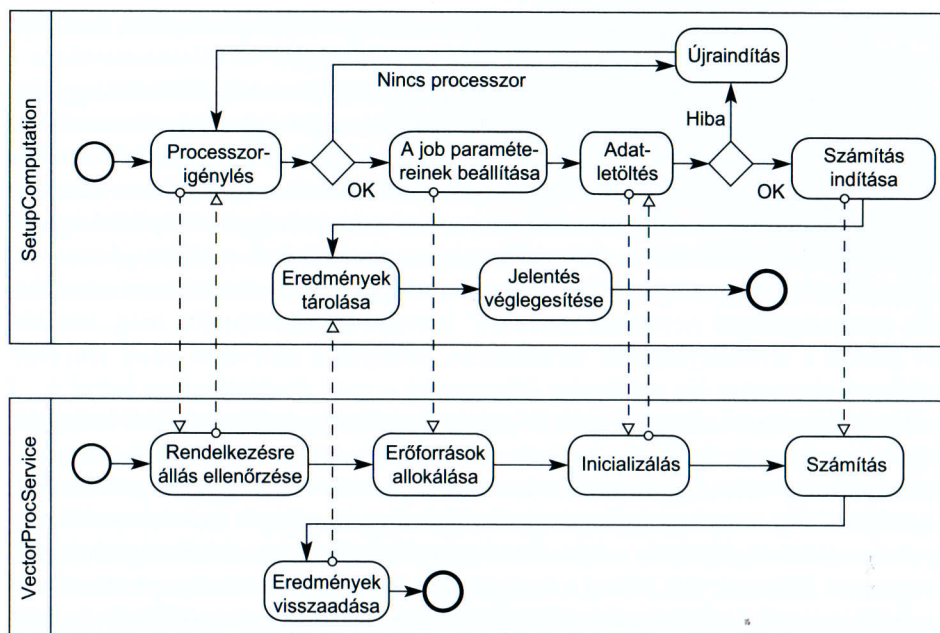
31.15. ábra. Hotelfoglalási munkafolyamat

A 31.15. ábra a korábban leírt vakáció csomag szolgáltatás egyszerű BPMN-modelljének része. A modellen a hotelfoglalás egyszerűsített munkafolyamata látható, ahol feltételezzük, hogy létezik egy Hotels szolgáltatás, amelyhez a következő műveletek tartoznak: GetRequirements, CheckAvailability, ReserveRooms, NoAvailability, ConfirmReservation és CancelReservation. A folyamat a következő: bekérjük a felhasználó igényeit, ellenőrizzük, hogy van-e rendelkezésre álló szoba, ha igen, lefoglalunk egyet a megadott dátumoknak megfelelően.

A modellen megjelenik a BPMN néhány fontos fogalma, amelyeket munkafolyamat-modellek létrehozásához használhatunk:

1. A tevékenységeket lekerekített sarkú téglalapok jelölik. A tevékenységeket végrehajthatja egy ember, vagy akár automatikus szolgáltatás is.
2. Az eseményeket körök reprezentálják. Egy esemény olyan valami, ami az üzleti folyamat során történik. Egyszerű kör jelöli a kiinduló eseményt, a befejező eseményt pedig sötétebb. A köztes események megjelenítésére dupla köröket (az ábrán ilyen nem szerepel) használunk. Az események lehetnek ütemezett, így lehetőségünk van a munkafolyamatok periodikus és időzített végrehajtására is.
3. Rombusszal jelöljük az átjárókat. Az átjáró az üzleti folyamat olyan lépése, amelyben valamilyen döntés történik. Például a 31.15. ábrán egy döntés történik az alapján, hogy van-e szabad szoba vagy sem.
4. Folytonos nyíllal jelöljük a tevékenységek végrehajtási sorrendjét; szaggatott vonal jelzi az üzenetek áramlását a tevékenységek között – a 31.15. ábrán ezek az üzenetek a hotelfoglalási szolgáltatás és az ügyfél között áramlanak.

A fenti eszközök általában elegendőek a legtöbb munkafolyamat lényegének leírására. Azonban a BPMN számos olyan további eszközt tartalmaz, amelyek



31.16. ábra. Együttműködő munkafolyamatok

ismertetése nem e könyv feladata. Ezek olyan információkat adnak az üzleti folyamatok leírásához, amelyek lehetővé teszik a végrehajtható formába történő átalakítást. Így azok a webszolgáltatások, amelyek BPMN-ben leírt szolgáltatások kompozíciójaként állnak elő, elkészíthetők egy üzleti folyamatmodellből.

A 31.15. ábra egy, a lefoglalási szolgáltatást kínáló cégnél futó szolgáltatást mutat. Azonban tudjuk, hogy a szolgáltatásorientált megközelítést egyik legnagyobb haszna, hogy támogatja a szervezetek közötti számítástechnikai együttműködést. Ez azt jelenti, hogy a teljes számításban különböző cégek szolgáltatásai vesznek részt. Ezt BPMN-ben úgy jeleníthetjük meg, hogy mindegyik szervezethez kifejlesztünk egy munkafolyamatot, és ábrázoljuk a köztük történő interakciókat.

Ennek illusztrálására egy újabb példát használok, amit a fűrtözéses technológia témaköréből vettem. A szolgáltatásorientált megközelítést arra találták ki, hogy elősegítse az erőforrások (mint a nagy teljesítményű számítógépek) megosztását. Ebben a példában feltételezzük, hogy egy kutatólaboratórium egy vektorszámítógépet (olyan számítógép, amely képes párhuzamos műveletek végrehajtására értékek tömbjein) kínál szolgáltatásként (VectorProcService). Ez egy másik szolgáltatáson, a SetupComputation szolgáltatáson keresztül érhető el. Ezek a szolgáltatások és a köztük történő interakciók a 31.16. ábrán láthatók.

Ebben a példában a SetupComputation szolgáltatás munkafolyamata hozzáférést igényel egy vektorprocesszorhoz, és ha megkapja azt, akkor a végrehajtott számítás eredményét továbbítja a feldolgozó szolgáltatáshoz. Ha a számítások befejeződtek, az eredmény mindig átkerül egy helyi számítógépre. A VectorProcService munkafolyamata ellenőrzi, hogy van-e szabad processzor, erőforrásokat

foglal le a számításhoz, inicializálja a rendszert, végrehajtja a számítást, és a végeredményt továbbítja a kliensszolgáltatáshoz.

A BPMN fogalmaival minden szervezet munkafolyamatát különálló egységként reprezentáljuk. Ezt grafikusán úgy jelenítjük meg, hogy a folyamat minden résztvevőjének munkafolyamatát egy téglalapba zárjuk, és a szolgáltatás nevét a téglalap bal oldali végébe írjuk függőlegesen. A különálló egységekben definiált munkafolyamatok koordinálása üzenetcserevel valósul meg; a különböző egységek tevékenységeiből álló szekvenciák nem megengedettek. Azokban az esetekben, amikor a munkafolyamatban egyazon szervezet különböző részei szerepelnek, ezt az egységek nevesített „sávokra” bontásával jeleníthetjük meg. Minden sáv azokat a tevékenységeket tartalmazza, amelyek a szervezet adott részéhez tartoznak.

Ha kifejlesztettünk egy üzleti folyamatmodellt, finomítanunk kell azt, attól függően, milyen szolgáltatásokat fedeztünk fel. Ahogyan a 31.14. ábra magyarázatánál is kifejtettem, egy modell számos iteráción áteshet, míg egy olyan modellt kapunk, amely annyi szolgáltatást hasznosít újra, amennyit csak lehet. Ha egy ilyen tervet elkészítettünk, a következő teendőnk az, hogy ebből végrehajtható programot állítsunk elő. Mivel a szolgáltatások függetlenek az implementációs nyelvtől, a megvalósítás történhet bármilyen nyelven. Ma mind a C#, mind a Java fejlesztői környezetek támogatják a webszolgáltatások kompozícióját.

A webszolgáltatások kompozíciójának megvalósításának közvetlen támogatására számos szabvány született. Ezek közül a WS-BPEL (Business Process Execution Language – üzleti folyamat végrehajtónyelv) a legismertebb. Ez a nyelv egy XML-alapú „programozási nyelv”, amely támogatja a szolgáltatások közötti interakciókat. Ez számos további szabványon keresztül valósul meg. Ilyenek a WS-Coordination (Cabrer és mások, 2005), amelyet annak specifikálására használnak, hogyan történjen a szolgáltatások koordinációja, és a WS-CDL (Choreography Description Language – koreográfia leírónyelv) (Kavantzás és mások, 2004), amely a résztvevők közötti üzenetcsere definíálásának eszköze (Andrews és mások, 2003).

Ezek mind XML-szabványok, azaz az így készült leírások terjedelmesek és nehezen olvashatók. A közvetlenül XML-jelölésrendszerben történő programírás igen bonyolult lenne, és sok hibát okozna. Ezek miatt úgy döntöttem, nem bocsátkozom részletekbe ezekkel kapcsolatban, mivel a munkafolyamat és a szolgáltatáskompozíció megértése szempontjából nem feltétlenül fontosak. Ahogy a szolgáltatásorientált technológiák egyre jobban terjednek, ezeket a leírásokat majd automatikusan lehet generálni. Olyan eszközök fognak megszületni, amelyek képesek a munkafolyamat-leírásokból végrehajtható szolgáltatáskompozíciókat előállítani.

31.3.2. Szolgáltatások tesztelése

A tesztelés minden rendszer fejlesztési folyamatának elhagyhatatlan része, ugyanis a tesztelés szolgál annak ellenőrzésére, hogy a rendszer megfelel-e funkcionális és nemfunkcionális követelményeinek, illetve segít a fejlesztés során

kialakult hiányosságok feltárásában. Ahogyan a 22–24. fejezetekben láthattuk, a szoftvertesztelés és validáció támogatására számos különböző megközelítési mód született. Ezek közül sok (például a programátvizsgálás és a lefedettség teszt) a szoftver forráskódjának elemzésére támaszkodik. Azonban ha a szolgáltatásokat külső gyártóktól rendeljük meg, a forráskódok nem állnak rendelkezésünkre. A szolgáltatásorientált rendszerek esetén tehát el kell vetnünk a jól bevált forráskódalapú technikákat.

A szolgáltatás működésének megértése mellett a tesztelők további problémákkal is szembesülhetnek szolgáltatások és szolgáltatáskompozíciók tesztelése közben:

1. A külső szolgáltatások nem a felhasználó irányítása alá tartoznak, hanem a gyártó felelős értük. A gyártó pedig bármikor kivonhatja a forgalomból ezeket a szolgáltatásokat, vagy megváltoztathatja őket, amely érvénytelenné teszi a korábban, a tesztelés során megszerzett eredményeket. Ezeket a problémákat a komponensek esetében úgy kezelik, hogy egy-egy komponensből több verziót tartanak fenn. Azonban jelenleg nincs olyan szabványjavaslat, amely a verziók kezelésével foglalkozik.
2. A szolgáltatásorientált architektúrákról szóló hosszú távú elképzelések szerint a jövőben a szolgáltatások hozzákapcsolása a szolgáltatásorientált alkalmazásokhoz dinamikusan fog történni. Ez azt jelenti, hogy nem biztos, hogy egy alkalmazás minden egyes futásánál ugyanazt a szolgáltatást használja. Így a tesztek sikeresek lehetnek abban az esetben, ha az alkalmazás egy bizonyos szolgáltatáshoz kapcsolódik, viszont nem garantálható, hogy az aktuális futás során is azt a szolgáltatást fogja választani.
3. Mivel a legtöbb esetben a szolgáltatásokat több felhasználó is elérheti, ezért a szolgáltatás nemfunkcionális viselkedése nem csak egyszerűen attól függ, hogy a tesztelt alkalmazás hogyan használja azt. Lehet, hogy egy szolgáltatás jó teljesítménnyel dolgozik a tesztelés során, mert nincs különösebb terhelés alatt. Gyakorlatban a megfigyelt viselkedés megváltozik a többi felhasználó igényei miatt.
4. A szolgáltatások kifizetési modellje költségessé teheti a tesztelést. Több különböző fizetési modell létezik – néhány szolgáltatás szabadon használható, másokért feliratkozás útján kell fizetni, szintén másokért pedig használat alapján fizetünk. Ha a szolgáltatás szabadon használható, akkor a szolgáltatás gyártója nem akarja, hogy a tesztelők túlterheljék azt. Ha feliratkozás szükséges, akkor a szolgáltatás felhasználója vonakodni fog a szolgáltatás tesztelése előtti szerződéskötéstől. Ha a fizetés használat alapján történik, a tesztelők túlzónak érezhetik a tesztelés költségeit.
5. Korábban említettem a kompenzációs tevékenységek fogalmát, amelyekkel kivétel bekövetkezése esetén visszavonhatjuk a már véglegesített eseményeket (mint a repülőgép-foglalás). Az ilyen tevékenységek tesztelésével az a probléma, hogy más szolgáltatások hibás működésétől függhetnek. Annak biztosítása, hogy ezek a szolgáltatások valóban hibáznak a tesztelés folyamán, igen bonyolult lehet.

Az előbbi gondok különösen akkor súlyosak, ha külső szolgáltatásokat használunk. Kevésbé rossz a helyzet, ha a szolgáltatások egyazon vállalaton belülről származnak, vagy ha az együttműködő cégek megbíznak a partnereik szolgáltatásaiban. Ilyen esetekben a forráskód is rendelkezésre állhat, és valószínűleg a fizetés módja sem okoz problémákat. A fenti akadályok leküzdése, a szolgáltatás-orientált alkalmazások tesztelését megkönnyítő eszközök és technikák kifejlesztése jelenleg is fontos kutatási téma.

Kulcsfogalmak

- A szolgáltatásorientált szoftverfejlesztés arra az elvre épül, hogy a programok felépíthetők olyan független szolgáltatások kompozíciójával, amelyek valamilyen újrafelhasználható funkcionalitást valósítanak meg. A szolgáltatások nyelvfüggetlenek, implementációjuk a széles körben elfogadott és használt XML-alapú szabványokra épül.
- A szolgáltatás-interfészek az XML-alapú WSDL nyelvvvel definiálhatók. A WSDL-specifikáció tartalmazza az interfész típusainak és műveleteinek leírását, a szolgáltatás által használt csatoló protokollt és azt, hol található meg a szolgáltatás.
- A szolgáltatások a következőképpen csoportosíthatók: segédszolgáltatások, amelyek valamilyen általános célú funkcionalitást valósítanak meg; üzleti szolgáltatások, amelyek egy üzleti szolgáltatás valamely részét valósítják meg; koordinációs szolgáltatások, amelyek a többi szolgáltatás végrehajtását koordinálják.
- A szolgáltatástervezés folyamata magában foglalja az implementálandó szolgáltatásjelöltek azonosítását, a szolgáltatás-interfész definiálását, továbbá a szolgáltatás implementálását, tesztelését és üzembe helyezését.
- Az olyan ősrendszerekhez, amelyek továbbra is hasznosak lehetnek a szervezet számára, szolgáltatás-interfészeket definiálhatunk. Ezután az ősrendszer funkcionalitása újrafelhasználható más alkalmazásokban.
- A szolgáltatások felhasználásával történő szoftverfejlesztés alapötlete az, hogy programokat úgy is előállíthatunk, hogy szolgáltatásokat összekombinálunk és konfigurálunk.
- Az üzleti folyamatmodellek az üzleti folyamatokban végbemenő tevékenységeket és információcserét definiálják. Az üzleti folyamat tevékenységeit implementálhatják szolgáltatások, így a modell egy szolgáltatáskompozíciót reprezentál.
- A szoftvertesztelés forráskódon alapuló technikái nem használhatók olyan szolgáltatásorientált rendszerek esetén, amelyek külső gyártók szolgáltatásai-ból épülnek fel.

További irodalom

A weben tömértelen mennyiségű bevezető anyag található a webszolgáltatások minden témakörével kapcsolatban. Azonban a kezembe került könyvek közül az alábbi kettő a legszínvonalasabb. Számos szerzővel ellentétben Earl a szoftvertervezés témaköreit is tárgyalja könyveiben.

Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Upper Saddle River, NJ: Prentice Hall.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology and Design*, Upper Saddle River, NJ: Prentice Hall.

Feladatok

- 31.1. Mik a szolgáltatások és a szoftverkomponensek közötti különbségek?
- 31.2. Magyarázza meg, miért elengedhetetlen, hogy a szolgáltatásorientált architektúrák szabványokra épüljenek.
- 31.3. Miért fontos, hogy minimalizáljuk a szolgáltatások üzeneteinek számát?
- 31.4. Magyarázza meg, miért kell a szolgáltatásoknak mindig tartalmazniuk egy kivétel interfészt, amely a hibákról és kivételekről értesíti a klienseket.
- 31.5. Ugyanazt a jelölést alkalmazva bővítse ki a 31.6. ábrát! adja meg a `MaxMinType` és az `InDataFault` definícióját. A hőmérsékleteket egészsként kell ábrázolni egy kiegészítő mezővel, amely azt határozza meg, hogy a fokot Celsiusban vagy Fahrenheitben kell érteni. Az `InDataFault` lehet egy egyetlen hibakódból álló egyszerű típus.
- 31.6. Definíáljon egy egyszerű interfészt a 31.8. ábrán szereplő pénznemátváltó és kreditellenőrző szolgáltatásokhoz.
- 31.7. Tervezze meg a 31.12. ábra szolgáltatásainak lehetséges üzeneteit. Ezeket megadhatja UML- és XML-jelölés használatával.
- 31.8. Soroljon föl két olyan fontos alkalmazástípust, amelyek valószínűleg nem alkalmazhatják a szolgáltatásorientált megközelítést. Válaszát indokolja.
- 31.9. A 31.2.1. alfejezetben bemutattam egy céget, amely kifejlesztett egy katalógusszolgáltatást, amelyet az ügyfelek webalapú beszerzési rendszerei használhatnak. BPMN használatával tervezzen olyan munkafolyamatot, amely ezt a szolgáltatást használja számítógépre leadott rendelések kérésére és ilyen rendelések leadására.
- 31.10. Magyarázza meg, mi a „kompenzációs tevékenység”, és példa használatával indokolja, miért fontos ilyen események szerepeltetése a munkafolyamatokban.
- 31.11. A vakáció foglalási szolgáltatáshoz tervezzen olyan munkafolyamatot, amely földi szállítást biztosít a reptérre érkező utasok számára. Az utasok utazhatnak taxival vagy bérelhetnek autót. Feltételezheti, hogy a taxis cégek és az autókölcsönzők webszolgáltatásokat nyújtanak ügyfeleiknek.
- 31.12. Példák felhasználásával részletezze, miért bonyolult a kompenzációs tevékenységeket tartalmazó szolgáltatások alapos tesztelése.