8 RENDSZERMODELLEK

TÉMAKÖRÖK

A fejezet célja néhány, a követelménytervezési folyamat alatt kifejleszthető rendszermodell bemutatása. A fejezet elolvasása után:

- meg fogjuk érteni, miért fontos a rendszer határainak megállapítása és környezetének modellezése;
- érteni fogjuk a viselkedési modellezés, adatmodellezés és objektummodellezés fogalmait;
- ismerni fogunk néhány, a Unified Modeling Language-ben (UML) definiált jelölést, és tudni fogjuk, hogyan használhatók ezek a jelölések rendszermodellek fejlesztéséhez.

TARTALOM

A felhasználói követelményeket természetes nyelven ajánlott megírni, mivel azt azoknak kell megérteni, akik nem informatikai szakemberek. Ugyanakkor a részletesebb rendszerkövetelmények szakmai fogalmakkal is kifejezhetők. Széles körben használt technika, amikor a rendszer-specifikációt rendszermodellek halmazaként dokumentálják. Ezek a modellek olyan grafikus reprezentációk, amelyek leírják az üzleti folyamatokat, a megoldandó problémát és a kifejlesztendő rendszert. A grafikus reprezentáció használata miatt a modellek gyakran sokkal érthetőbbek, mint a rendszerkövetelmények részletes természetes nyelvi leírásai. Mi több, fontos hidat képeznek az elemzési és tervezési folyamatok között.

Modelleket használhatunk az elemzési folyamat során a cserére vagy fejlesztésre váró meglévő rendszer jobb megértésének kialakításához vagy a kívánt rendszer specifikálásához. Különböző modelleket fejleszthetünk ki azért, hogy velük a rendszert különböző szemszögekből reprezentálhassuk. Például:

1. Külső szemszögből a rendszer kontextusát vagy környezetét modellezzük.
2. Viselkedési szemszögből a rendszer viselkedését modellezzük.
3. Strukturális szemszögből a rendszer felépítését vagy a rendszer által feldolgozott adatok szerkezetét modellezzük.

Ezt a három szempontot vizsgáljuk a fejezet során, illetve tárgyaljuk az objektummodellezést, amely bizonyos mértékben ötvözi a viselkedési és a strukturális modellezést.

Egy rendszermodellnél a legfontosabb szempont, hogy részleteket hagy el. A rendszermodell a tanulmányozott rendszer absztrakciója, nem pedig a rendszer egy másik reprezentációja. Ideális esetben a rendszer *reprezentációja* minden információt kezel a reprezentált egyedről. Az *absztrakció* ezzel szemben szándékosan egyszerűsít, és kiemeli a legszembetűnőbb jellemzőket. Ha mondjuk bekövetkezne az a fölöttébb valószínűtlen esemény, hogy erről a könyvről cikksorozat jelenne meg egy újságban, a bemutatás ott a kulcsfogalmak absztrakciója lenne. Ha angolról olaszra fordítanák a könyvet, az egy másik reprezentáció lenne. A fordító szándéka az lenne, hogy az összes információt úgy tartsa meg, ahogyan az angol változatban szerepelt.

A rendszermodellek különböző típusai az absztrakció különböző megközelítésein alapulnak. Az adatfolyammodell (például) az adatok áramlására és az adatokon végzett funkcionális átalakításokra helyezi a hangsúlyt. Az adatszerkezetek részleteit elhagyja. Ezzel szemben az adategyedek és kapcsolataik modellje a rendszer adatstruktúráit dokumentálja, nem pedig annak funkcionalitását.

Az elemzési folyamat alatt különböző típusú rendszermodelleket állíthatunk elő:

1. *Adatfolyammodell.* Az adatfolyammodellek azt mutatják be, hogyan kerülnek feldolgozásra az adatok a rendszerben különböző szakaszokban.
2. *Kompozíciós modell.* A kompozíciós vagy aggregációs modell azt mutatja be, hogyan épülnek föl a rendszerbeli egyedek más egyedekből.
3. *Architekturális modell.* Az architekturális modellek a rendszert felépítő fő alrendszereket mutatják be.
4. *Osztálymodell.* Objektum osztály/öröklődési diagramok az egyedek közös jellemzőit mutatják be.
5. *Inger-válasz modell.* Az inger-válasz modellek vagy *állapotátmenet-diagramok* azt mutatják be, hogyan reagál a rendszer belső és külső eseményekre.

A modelleknek ezen típusairól szó lesz a fejezetben. Ahol csak lehetséges, a Unified Modeling Language (UML) jelöléseit fogom használni, amely mára szabványos modellezőnyelvvé vált az objektumorientált modellezés területén (Booch *és mások,* 1999; Rumbaugh *és mások,* 1999a). Ahol az UML nem tartalmaz megfelelő jelölést, ott egyszerű, könnyen érthető jelöléseket alkalmazok majd a modell leírásához. Már fejlesztés alatt áll az UML új verziója (UML 2.0), de akkor, amikor ezt a fejezetet írtam, még nem volt elérhető. Ugyanakkor az itt használt UML-jelölés vélhetőleg kompatibilis lesz az UML 2.0-val.

8.1. KÖRNYEZETI MODELLEK

A követelmények feltárási és elemzési folyamata korai szakaszában ajánlott döntenünk a rendszer határairól. Ez magában foglalja a rendszer kulcsfiguráival való együttműködést azért, hogy szétválasszuk, mi tartozik magához a rendszerhez és mi annak környezetéhez. A rendszer költségének és az elemzéshez szükséges időnek a csökkentése érdekében ezeket a döntéseket még a folyamat kezdetén célszerű meghozni.

Néhány esetben a rendszer és környezete közötti határ viszonylag tisztán kirajzolódik. Például egy meglévő kézi vezérlésű vagy számítógépesített rendszert automatizáltra cserélnek le, az új rendszer környezete rendszerint megegyezik a meglévő rendszer környezetével. Más esetekben nagyobb a rugalmasság, magunk szabjuk meg a követelménytervezési folyamat alatt, mi képezi a rendszer és környezete határait.

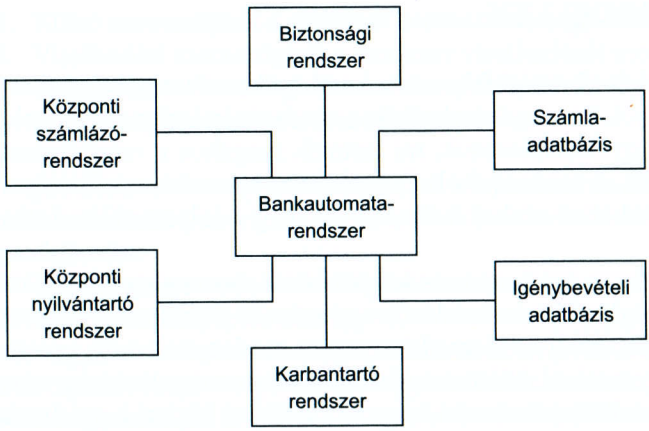
Mondjuk, hogy a LIBSYS könyvtári rendszer specifikációját fejlesztjük. Emlékezzünk, hogy a rendszer jogvédett anyagok elektronikus másolatait juttatná el a felhasználók számítógépére. A felhasználók ezután kinyomtathatnák saját másolataikat az anyagról. A rendszer specifikációjának kifejlesztésekor el kell döntenünk, hogy más könyvtári adatbázisrendszerek, mint például a könyvtári katalógusok, a rendszer határain belül vannak-e. Ha igen, elérhetővé kellhet tenni a rendszert a katalógus felhasználói felületén keresztül; különben kellemetlenségeket okozhat a felhasználóknak a mozgás a rendszerek között.

A rendszer határainak meghatározása nem elhanyagolható döntés. Szociális és szervezeti megfontolások azt okozhatják, hogy a rendszer határának helyét esetleg nem technikai tényezők is megszabhatják. Például a rendszer határa kialakítható úgy, hogy az elemzési folyamat egyetlen helyszínen elvégezhető legyen; megválasztható úgy, hogy ne kelljen konzultálni valamelyik különösen nehéz természetű vezetővel; kialakítható úgy, hogy megnövekedjen a rendszer költsége, és a rendszerfejlesztést ki kelljen terjeszteni a rendszer tervezésére és implementálására is.

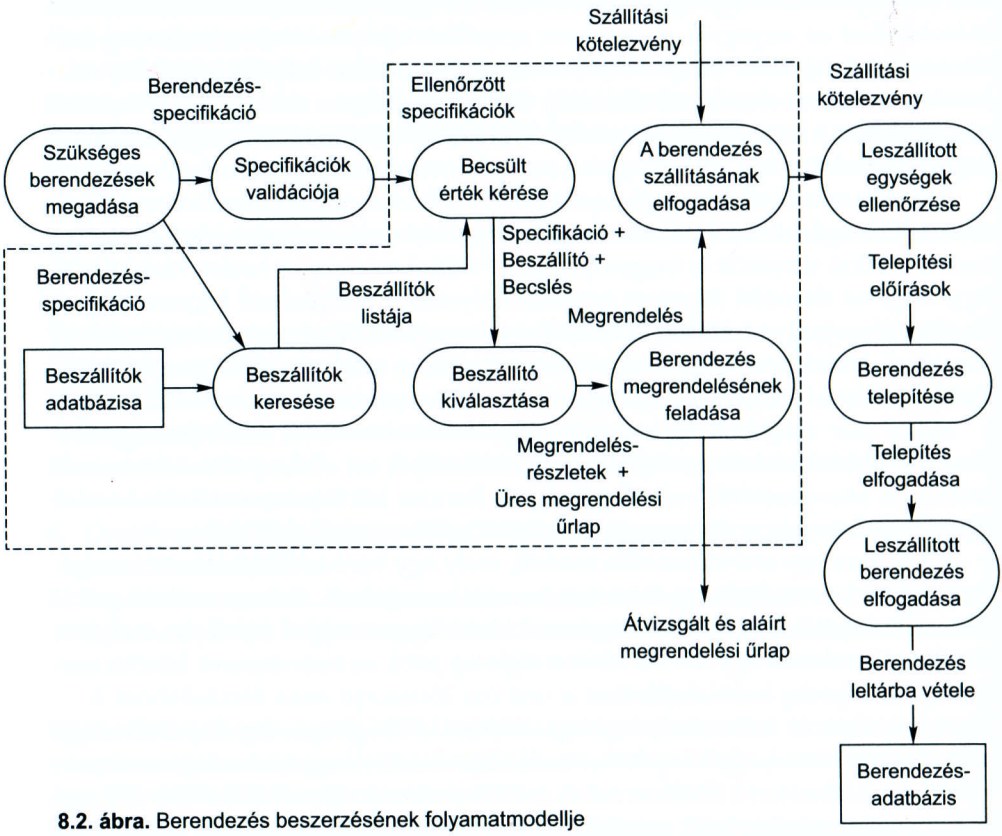
Mikor már meghoztunk bizonyos döntéseket a rendszer határaira vonatkozóan, az elemzési tevékenység részeként definiáljuk ezt a környezetet, és a rendszernek a környezetétől való függőségeit. Rendes körülmények között az első lépés e tevékenység során egyszerű architekturális modell előállítása.

A 8.1. ábra egy architekturális modell, mely egy banki ATM-hálózatot magában foglaló információs rendszer szerkezetét szemlélteti. A magas szintű architekturális modelleket általában egyszerű blokkdiagramokkal fejezik ki, melyben minden alrendszert egy névvel ellátott téglalap jelöl, az alrendszerek közötti asszociációkat pedig vonalak jelzik.

A 8.1. ábráról leolvashatjuk, hogy minden ATM össze van kapcsolva egy számlaadatbázissal, egy központi számlázórendszerrel, egy biztonsági rendszerrel és egy karbantartó rendszerrel. A rendszer szintén összeköttetésben áll egy igénybevételi adatbázissal, amely az ATM hálózatának használatát figyeli, valamint egy központi nyilvántartó rendszerrel. Ez a nyilvántartó rendszer olyan szolgáltatásokat biztosít, mint a biztonsági mentés és a nyomtatás. Ezeket tehát nem szükséges magának az ATM-rendszernek tartalmaznia.



8.1. ábra. ATM-rendszer környezete



8.2. ábra. Berendezés beszerzésének folyamatmodellje

Az architekturális modellek a rendszer környezetét írják le. Ugyanakkor nem mutatják meg a környezet egyéb rendszerei és az éppen specifikált rendszer közötti kapcsolatokat. Külső rendszerek előállíthatnak adatot a rendszer számára, és kaphatnak adatot a rendszertől. Megoszthatnak adatot a rendszerrel, összeköttetésben állhatnak egymással közvetlenül, hálózaton keresztül, vagy az is lehet, hogy sehogyan sem. Fizikai elhelyezésük szempontjából lehetnek egy helyen, vagy lehetnek külön épületekben. Ezek az összefüggések mind hatással lehetnek a definiált rendszer követelményeire, tehát számolnunk kell velük.

Ezért az egyszerű architekturális modelleket rendszerint más modellekkel is kiegészítik, melyek a rendszer által támogatott folyamattevékenységeket mutatják be. A (következő alfejezetben leírt) adatfolyammodellek szintén használhatók az adatátvitel bemutatására a rendszer és a környezetében lévő más rendszerek között.

A 8.2. ábra egy berendezés beszerzésének szervezeten belüli folyamatmodelljét mutatja be. Tartalmazza a szükséges berendezés meghatározását, beszállítók keresését és kiválasztását, a berendezés megrendelését, a berendezés leszállítását és az átadás utáni tesztelést. Ha ehhez a folyamathoz számítógépes támogatást írunk elő, el kell döntenünk, mely tevékenységekhez lesz tényleges támogatás ezek közül. A többi tevékenység a rendszer határain kívül esik. A 8.2. ábrán szaggatott vonal zárja közre a rendszer határain belül lévő tevékenységeket.

8.2. VISELKEDÉSI MODELLEK

A viselkedési modelleket a rendszer átfogó viselkedésének leírására használják. A viselkedési modellek két típusát tárgyalom: az adatfolyammodelleket, melyek a rendszer adatfeldolgozását modellezik, és az állapotátmenet-modelleket, melyek azt modellezik, hogyan reagál a rendszer az eseményekre. Ezek a modellek használhatók együtt vagy külön, a fejlesztett rendszer típusától függően.

A legtöbb üzleti rendszer elsősorban adatvezérelt. Vezérlésük adatok bevitelével történik, és viszonylag kevés külső eseményt dolgoznak fel. Az adatfolyammodell mindazt biztosítja, ami ezeknek a rendszereknek a reprezentálásához szükséges. Ezzel ellentétben a valós idejű rendszerek gyakran eseményvezéreltek, igen kevés adatfeldolgozással. Ezek viselkedését leghatékonyabban az állapotátmenet-modellel (8.2.2. alfejezet) reprezentálhatjuk. A rendszerek más osztályai lehetnek egyszerre adat- és eseményvezéreltek. Ezeknél mindkét modelltípust érdemes kifejleszteni.

8.2.1. Adatfolyammodellek

Az adatfolyammodellek intuitív módon mutatják be, hogyan dolgozza fel az adatokat a rendszer. Elemzési szinten annak modellezésére ajánlatos használni az ilyen modelleket, hogyan kerülnek feldolgozásra az adatok a meglévő rendszerben. Az adatfolyammodellek elemzési célra történő használata DeMarco

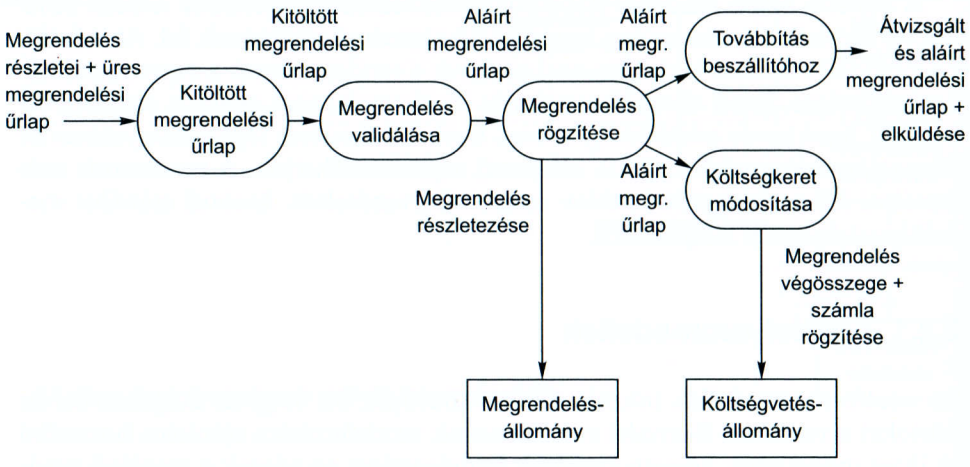
strukturált rendszerelemzésről szóló könyvének (DeMarco, 1978) megjelenése után vált elterjedtté. Ezek lényeges részei az ebből a műből kifejlődött strukturált módszereknek. Az ezekben a modellekben használt jelölések funkcionális feldolgozást (lekerekített téglalapok), adattárolást (téglalapok) és a függvények közötti adatmozgást (címkézett nyilak) reprezentálnak.

Az adatfolyammodelleket annak bemutatására használják, hogyan áramlanak végig az adatok a feldolgozási lépések sorozatán. Például feldolgozási lépés lehet a kettőzött rekordok kiszűrése a megrendelő adatbázisában. Az adat minden lépésben átalakításon megy keresztül, mielőtt a következő szakaszba kerül. Ezek a feldolgozási lépések vagy transzformációk szoftverfolyamatokat vagy programfüggvényeket jelentenek, ha az adatfolyam-diagramokat szoftverterv dokumentálásához használják. Ezzel szemben egy elemzési modellben a feldolgozást emberek vagy számítógépek végezhetik.

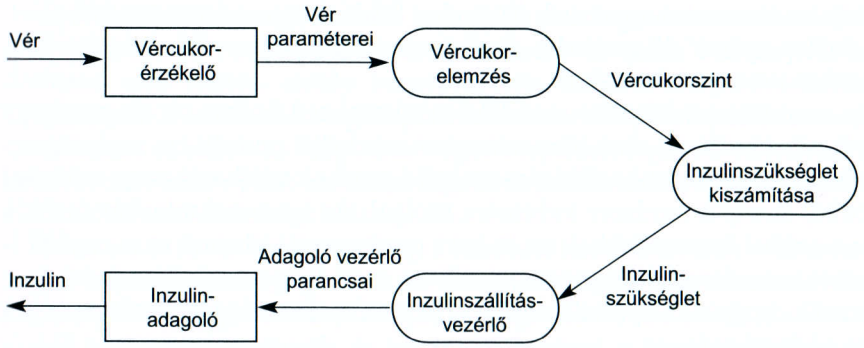
Adatfolyam-diagramot szemléltet a 8.3. ábra, amely egy árurendelés (például számítógépes berendezés) feldolgozásának lépéseit mutatja be egy szervezetnél. Ez a modell leírja az adatok feldolgozását az általános folyamatmodell **Berendezés megrendelésének feladása** tevékenységénél. A modell bemutatja, hogyan mozog az áruk rendelése az egyik folyamattól a másikig. Szintén bemutatja a folyamat során használt adattárolókat (**Megrendelésállomány** és **Költségvetés-állomány**).

Az adatfolyammodelleknek az a jelentősége, hogy a rendszeren keresztüli egyes folyamatmozgásokhoz társított adatok dokumentálása és nyomon követése segíti az elemzőket megérteni, mi is történik. Az adatfolyam-diagramok előnye néhány más modellezési jelöléssel szemben, hogy egyszerűek és intuitívak. Ezek általában a rendszer potenciális felhasználóinak is elmagyarázhatók, akik így részt vehetnek az elemzés validálásában.

Elvben az adatfolyam-diagramokhoz hasonló modellek fejlesztését fentről lefelé kellene végezni. Ez a példa utal rá, hogy először az általános beszerzési fo-



8.3. ábra. A megrendelés feldolgozásának adatfolyam-diagramja



8.4. ábra. Egy inzulinadagoló rendszer adatfolyam-diagramja

lyamatot ajánlott elemeznünk. Ezután folytatjuk az alfolyamatok (mint például a megrendelés) elemzését. A gyakorlatban az elemzés sosem így történik. Egy időben több szintről szerzünk ismereteket. Megtehetjük, hogy előbb az alacsonyabb szintű modelleket fejlesztjük ki, majd azokat absztrahálva egy általánosabb modellt alkotunk meg.

Az adatfolyammodellek funkcionális szemszögből mutatják a rendszert, ahol minden átalakítás egyetlen függvényt vagy folyamatot reprezentál. Ezek különösen hasznosak a követelmények elemzésekor, mivel velük a feldolgozás elejétől végéig bemutatható a rendszerben. Vagyis ezek a teljes műveletsorozatot bemutatják, amely során egy bemenetből a rendszer válaszaként a megfelelő kimenet előállítódik. A 8.4. ábra egy adatfolyam-diagram ilyen használatára mutat példát. Ez a diagram a 3. fejezetben bemutatott inzulinadagoló rendszerben történő adatfeldolgozást szemlélteti.

8.2.2. Állapotátmenet-modellek

Az állapotátmenet-modellek azt írják le, hogyan reagál egy rendszer belső vagy külső eseményekre. Az állapotátmenet-modell azokat a rendszerállapotokat és eseményeket mutatja be, melyek egy állapotból egy másikba történő átmenetet okoznak. Az adatok áramlását a rendszeren belül nem mutatja be. Ezt a modellt gyakran használják valós idejű rendszerek modellezésére, mert ezeket a rendszereket gyakran a rendszer környezetéből érkező ingerek vezérlik. Például a 13. fejezetben tárgyalt valós idejű riasztórendszer a mozgásérzékelőkből, ajtónyitásérzékelőkből stb. származó ingerekre válaszol.

Az állapotátmenet-modellek gyakran képezik részét olyan valós idejű tervezési módszereknek, mint amiket Ward és Mellor (Ward és Mellor, 1985) és Harel (Harel, 1987; 1988) vezetett be. Harel módszere *Állapotdiagramok*nak nevezett jelölést használ, és ezek voltak az UML-ben lévő állapotátmenet-modellezési jelölés alapjai.

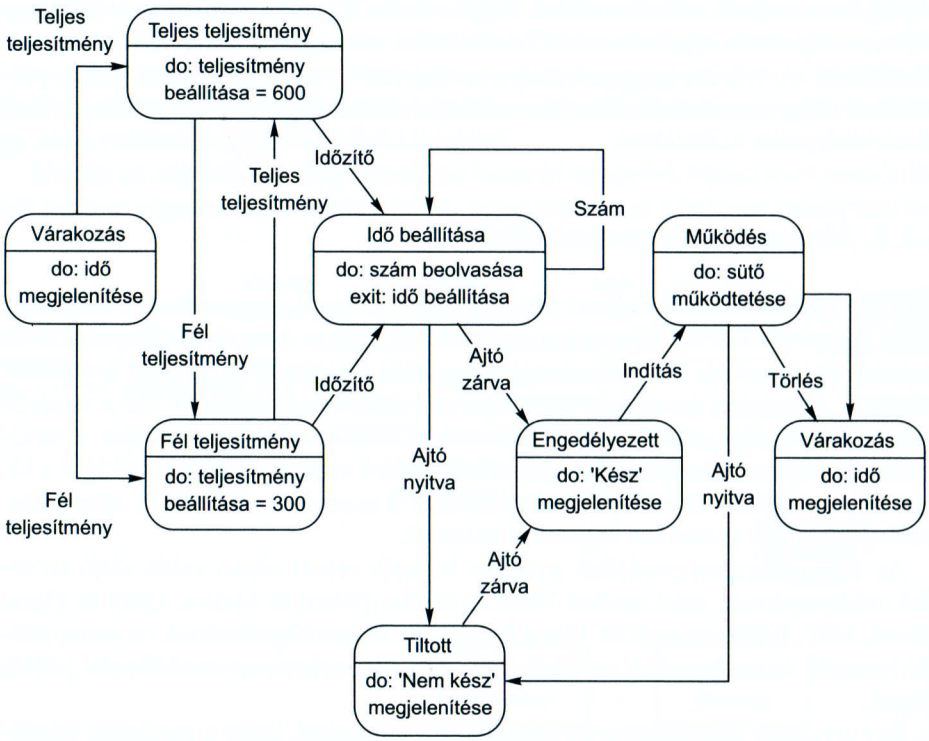
Egy rendszer állapotátmenet-modellje azt feltételezi, hogy a rendszer, tetszőleges időpillanatban, a számos lehetséges állapot egyikében van. Inger érkezése

kiválthatja az átmenetet egy másik állapotba. Például egy szelepet vezérlő rendszer a „Szelep nyitva" állapotból a „Szelep zárva" állapotba válthat, ha üzemeltetői utasítás (az inger) érkezik.

Ezt a rendszer-modellezési szemléletet tükrözi a 8.5. ábra. A diagram egy egyszerű mikrohullámú sütő állapotátmenet-modelljét mutatja be, melyen csupán a teljesítményfokozat beállítására szolgáló gombok találhatók meg, valamint egy időzítő, amely a rendszer indítására szolgál. Az igazi mikrohullámú sütők valójában sokkal összetettebbek az itt leírt rendszernél. Viszont ez a modell is tartalmazza a rendszer lényeges összetevőit. A modell egyszerűsítése érdekében feltételezzük, hogy a mikrohullámú sütő használatakor végzett tevékenységek sorozata a következő:

1. Az teljesítményfokozat kiválasztása (fél vagy teljes teljesítmény).
2. A melegítési idő megadása.
3. Az indítógomb megnyomása, majd az étel megmelegítése a megadott időre.

Biztonsági okokból a sütőt nem szabad nyitott ajtóval üzemeltetni, és a melegítés befejeztekor egy csengő szólal meg. A sütő egy igen egyszerű alfanumerikus kijelzővel rendelkezik, amely a különféle vészjelzések és figyelmeztető üzenetek kijelzésére szolgál.



8.5. ábra. Egy egyszerű mikrohullámú sütő állapotátmenet-diagramja

Az UML-jelölést, melyet én is használok az állapotátmenet-modellek leírására, objektumok viselkedésének modellezésére tervezték. Mindemellett ez egy általános célú jelölés, amely bármilyenfajta állapotátmenetes modellezésnél használható. A lekerekített téglalapok a rendszer állapotait jelölik a modellben. Tartalmazza az abban az állapotban végzett tevékenységek rövid leírását (a „do" után). A címkézett nyilak azokat az ingereket jelölik, amelyek egy állapotból egy másikba történő átmenetet idéznek elő.

Tehát a 8.5. ábrán láthatjuk, hogy a rendszer kezdetben csak a fél teljesítmény vagy a teljes teljesítmény gombra ad választ. A felhasználó, miután kiválasztotta valamelyiket, meggondolhatja magát, és megnyomhatja a másik gombot. Ezután az idő beállítása következik, és ha az ajtó bezárul, a Start gomb engedélyezve lesz. A gomb megnyomása működésbe hozza a sütőt, ami az ételt a megadott ideig melegíti.

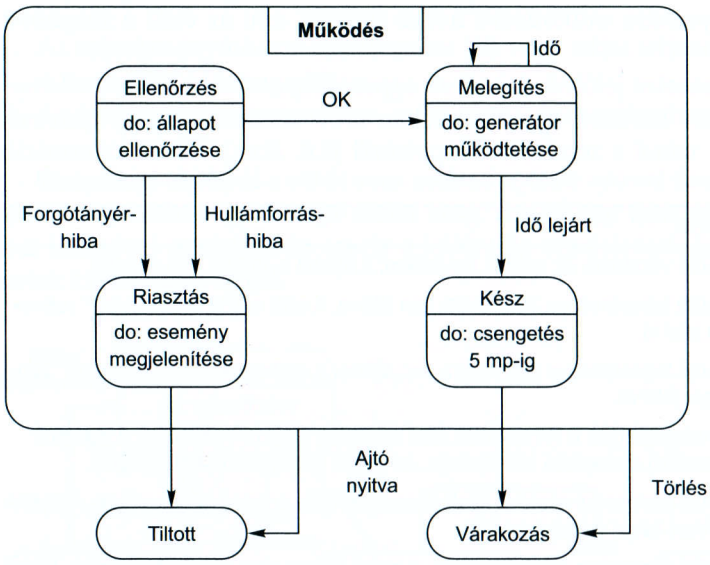
Az UML-jelölés szerint jelölnünk kell az egyes állapotokon belüli tevékenységeket. Egy részletes rendszer-specifikációban több részletet kell nyújtanunk mind az ingerekről, mind a rendszer állapotairól (8.6. ábra). Ez az információ

Állapot	**Leírás**
Várakozás	A sütő várakozik az adatok bevitelére. A kijelző a pontos időt mutatja.
Fél teljesítmény	A sütő teljesítménye 300 wattra van állítva. A sütő a „Fél teljesítmény" szöveget jelzi ki.
Teljes teljesítmény	A sütő teljesítménye 600 wattra van állítva. A sütő a „Teljes teljesítmény" szöveget jelzi ki.
Idő beállítása	A melegítési idő a felhasználó által megadott értékre van állítva. A kijelző a választott melegítési időt mutatja, és az idő beállítása után frissítődik.
Tiltott	A sütő biztonsági okokból tiltott állapotban van. A belső lámpa világít. A kijelző a „Nem kész" feliratot mutatja.
Engedélyezett	A sütő üzemkész állapotban van. A belső lámpa nem világít. A kijelző a „Kész" feliratot mutatja.
Működés	A sütő működés alatt áll. A belső lámpa világít. A kijelző a visszaszámlált időt mutatja. A melegítés befejezésekor a csengő öt másodpercig jelez. A belső lámpa világít. A kijelző a „Melegítés elvégezve" feliratot jeleníti meg a csengő hangjelzése mellett.
Inger	**Leírás**
Fél teljesítmény	A felhasználó megnyomta a fél teljesítmény gombot.
Teljes teljesítmény	A felhasználó megnyomta a teljes teljesítmény gombot.
Időzítő	A felhasználó megnyomta az egyik időzítőgombot.
Szám	A felhasználó számgombot nyomott meg.
Ajtó nyitva	A sütő ajtaja nyitva van.
Ajtó zárva	A sütő ajtaja be van zárva.
Indítás	A felhasználó megnyomta a start gombot.
Törlés	A felhasználó megnyomta a törlés gombot.

8.6. ábra. Egy mikrohullámú sütő állapotainak és ingereinek leírása

karbantartható adatszótárban vagy enciklopédiában (8.3. alfejezet), amit a rendszermodell megalkotásához használt CASE-eszköz tart karban.

Az állapotátmenetes megközelítés problémája, hogy a lehetséges állapotok száma gyorsan növekszik. Nagyméretű rendszerek modelljeinél ezért az állapotmodellek valamilyen strukturálása szükséges. Ennek egyik módja a több különálló állapotot magában foglaló szuperállapotok alkalmazása. A szuperállapot egy magasabb szintű modellben egyetlen állapotként látszik, viszont egy külön diagramon már részletesebben van kifejtve. Ennek az elképzelésnek a szemléltetéséhez tekintsük a 8.5. ábra **Működés** állapotát. Ez egy szuperállapot, amely a 8.7. ábrán látható formában fejthető ki.



8.7. ábra. Mikrohullámú sütő működése

A **Működés** állapot több alállapotot tartalmaz. Megmutatja, hogy a működés állapot-ellenőrzéssel kezdődik, és bármilyen probléma felfedezése esetén a rendszer riaszt és a működés leáll. A melegítés együtt jár azzal, hogy a mikrohullámgenerátor a megadott ideig működik, és befejezéskor a csengő megszólal. Ha az ajtó működés közben kinyílik, a rendszer tiltott állapotba kerül, ahogyan azt a 8.5. ábra mutatja.

8.3. ADATMODELLEK

A legtöbb nagyméretű szoftverrendszer az információk tárolásához nagyméretű adatbázis használatát teszi szükségessé. Néhány esetben ez az adatbázis a szoftverrendszertől független. Más esetekben azt a fejlesztett rendszer számára hozzák létre. A rendszerek modellezésének fontos része a rendszer által feldolgozott adatok logikai szerkezetének meghatározása. Ezeket *szemantikus adatmodelle*knek is hívják.

A leginkább elterjedt adatmodellezési technika az egyed-tulajdonság-kapcsolat modellezés (ETK-modellezés), amely az egyedeket, a hozzájuk tartozó tulajdonságokat és az ezen egyedek közötti kapcsolatokat mutatja meg. Ezt a modellezési megközelítést először Chen (Chen, 1976) javasolta a 70-es évek közepén, és számos változatát fejlesztették ki azóta (Codd, 1979; Hammer és McLeod, 1981; Hull és King, 1987), mindegyiknek ugyanaz az alapformája.

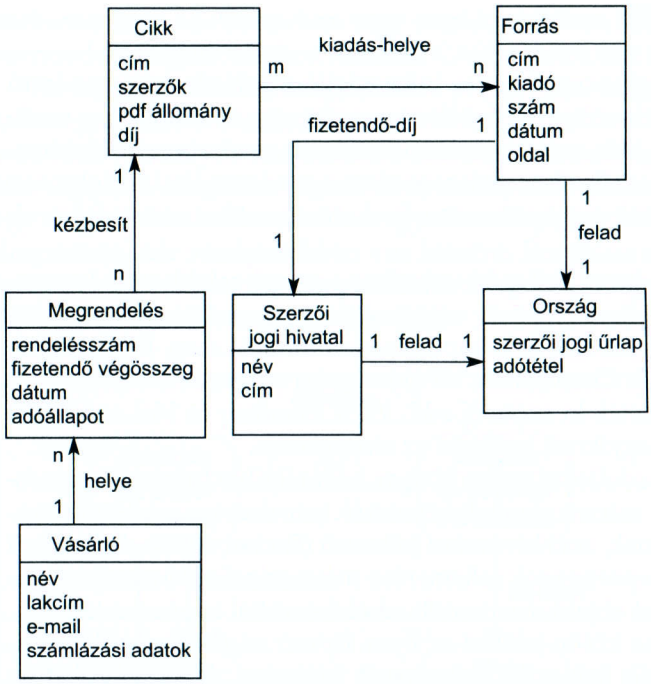
Az egyed-kapcsolat modelleket széles körben használják adatbázis-tervezésnél. Az ebből a modellből származó adatbázissémák természetes módon harmadik normálformában vannak, ami kívánatos jellemző (Barker, 1989). Az explicit típusosság és az al- és szupertípusok felismerése miatt szintén előremutató dolog, ha ezeket a modelleket objektumorientált adatbázisokkal implementáljuk.

Az UML nem tartalmaz külön jelölést az ilyen típusú adatbázis-modellezésre, mivel objektumorientált fejlesztési folyamatot feltételez, és az adatokat az objektumok és a köztük lévő kapcsolatok segítségével modellezi. Ugyanakkor az UML-t használhatjuk szemantikus adatmodell ábrázolására. Az ETK-modell egyedeit egyszerűsített osztályoknak tekinthetjük (amiknek nincsenek műveleteik), a tulajdonságokat osztályattribútumoknak, a kapcsolatokat pedig az osztályok közötti nevesített asszociációknak.

A 8.8. ábra a korábbi fejezetekben bemutatott LIBSYS könyvtári rendszer részét képező adatmodellre nyújt példát. Emlékezzünk, hogy a LIBSYS célja, hogy magazinokban és folyóiratokban közzétett jogvédett cikkek másolatait kézbesítse, és az ezek után járó díjat beszedje. Így az adatmodellnek információt kell hordoznia a cikkről, a szerzői jog tulajdonosáról és a cikk vásárlójáról. Feltettem, hogy a kifizetések nem közvetlenül, hanem nemzeti, szerzői jogokat felügyelő hivatalok közreműködésével történnek.

A 8.8. ábrán látható, hogy egy Cikk-nek a címet, a szerzőket, a cikk PDF-állományának nevét és a fizetendő díjat reprezentáló attribútumai vannak. Ez össze van kapcsolva a Forrás-sal, amely a cikket kiadta, és a kiadás országának Szerzői jogi hivatal-ával. Mind a Szerzői jogi hivatal, mind a Forrás az Ország-hoz kapcsolódik. A publikálás országa azért fontos, mert a szerzői jogi törvények országonként változnak. A diagramról szintén leolvasható, hogy a Vevő-k Megrendelés-t adnak fel a Cikk-ekre.

Mint minden más grafikus modell, az adatmodellek is híján vannak a részleteknek, így ajánlott kiegészíteni őket a modellben szereplő egyedek, kapcsolatok és tulajdonságok részletesebb leírásaival. Ezekről a részletesebb leírásokat is összegyűjthetünk egy tárolóban vagy adatszótárban. Az adatszótárak általános



8.8. ábra. A LIBSYS-rendszer szemantikus adatmodellje

körben használhatók rendszermodellek fejlesztésére, és használhatók még bármilyen típusú rendszermodell összes információjának kezelésére.

Az adatszótár, nagyon leegyszerűsítve, a rendszermodellekben szereplő nevek betűrend szerinti listája. Akárcsak a névnek, a szótárnak is ajánlott tartalmaznia az elnevezett egyedhez kapcsolódó leírást, és ha a név összetett objektumot tartalmaz, akkor az összetétel egy leírását is. Egyéb információk, mint a létrehozás időpontja, a létrehozó és az egyed reprezentációja, szintén feltüntethetők, a fejlesztett modell típusától függően. Az adatszótárak használatának előnyei:

1. *Mechanizmust ad a nevek kezeléséhez.* Nagyméretű rendszermodell fejlesztése során számos személy ad nevet egyedeknek és kapcsolatoknak. Fontos ezek konzisztens használata, illetve hogy a nevek ne ütközzenek egymással. Az adatszótár szoftvere ellenőrzi a nevek egyediségét, és figyelmezteti a követelményelemzőket a kettőzött nevekről.
2. *Szervezeti információk tárakénti szolgál.* A rendszer fejlesztésekor az elemzéshez, tervezéshez, implementációhoz és evolúcióhoz kapcsolható információk az adatszótárba kerülnek azért, hogy egy egyedről minden információ egy helyen legyen.

A 8.9. ábrán látható adatszótár-bejegyzések a LIBSYS-rendszer szemantikus adatmodelljének (8.9. ábra) neveit definiálják. Egyszerűsítettem a példa bemutatását azzal, hogy elhagytam pár nevet, és rövidítettem a hozzákapcsolt leírásokon.

Név	Leírás	Típus	Dátum
Cikk	A LIBSYS-t használók által megrendelhető publikált cikk részletei.	Egyed	2002. 12. 30.
szerzők	A cikk szerzőinek neve, akik majd a díj egy részét kapják.	Tulajdonság	2002. 12. 30.
Vásárló	A cikk másolatát megrendelő személy vagy vállalat.	Egyed	2002. 12. 30.
fizetendő-díj	1:1 kapcsolat a Cikk és a Szerzői jogi hivatal között, aminek a díjat kell fizetni.	Kapcsolat	2002. 12. 29.
lakcím(Vásárló)	A vásárló lakcíme. Minden szükséges számlázási információhoz ez lesz használva.	Tulajdonság	2002. 12. 31.

8.9. ábra. Példa adatszótár-bejegyzésekre

Fontos, hogy a rendszer minden nevét, legyen az egyedek, típusok, kapcsolatok, tulajdonságok vagy szolgáltatások neve, fel kell venni a szótárba. A szótár létrehozását, karbantartását és lekérdezését rendszerint szoftver végzi. Ezeket a szoftvereket más eszközökbe is integrálni szokták azért, hogy a szótár létrehozása részben automatizált legyen. Például a rendszermodellezést támogató CASE-eszközök általában tartalmaznak támogatást adatszótárakhoz, és a modellben először használt neveket felveszik a szótárba.

8.4. OBJEKTUMMODELLEK

Az objektumorientált megközelítések használata mára általánossá vált a szoftverfejlesztési folyamatban, különösen az interaktív rendszerek fejlesztésében. Ez azt jelenti, hogy a rendszerkövetelményeket objektummodellel fejezzük ki, a tervezést objektumok segítségével végezzük, a rendszert pedig objektumorientált programozási nyelven, például Javában vagy C++-ban fejlesztjük.

A követelményelemzés során kifejlesztett objektummodelleket mind a rendszer adatainak, mind azok feldolgozásának reprezentálására fel lehet használni. Ebben a vonatkozásban az adatfolyam- és a szemantikus modellek bizonyos használatai módjait egyesítik magukban. Arra is felhasználhatók, hogy megmutassák, hogyan osztályozhatók a rendszer egyedei és hogyan építhetők fel más egyedekből.

A rendszerek néhány osztálya számára az objektummodellek természetes módon tükrözik az általuk manipulált valós világbeli egyedeket. Ez különösen igaz akkor, ha a rendszer olyan kézzelfogható egyedekről dolgoz fel információt, mint az autók, légi járművek vagy könyvek, amiknek világosan azonosítható attribútumaik vannak. Az absztraktabb, magasabb szintű egyedeket, mint egy könyvtár, egy orvosi nyilvántartó rendszer vagy egy szövegszerkesztő, nehezebb objektumosztályokként modellezni. Ezeknek nincs szükségszerűen egyszerű, egymástól független attribútumokból és műveletekből álló interfészük.

A követelményelemzés során objektummodellek kifejlesztésével általában egyszerűsíthetjük az objektumorientált tervezés és programozás közötti átmenetet. Ugyanakkor úgy tapasztaltam, hogy a rendszerek végfelhasználói az objektummodelleket gyakran természetellenesnek és nehezen érthetőnek találják. Gyakran könnyebben befogadják a funkcionálisabb, adatfeldolgozáson alapuló bemutatást. Ezért olykor segíthet, ha az objektummodelleket adatfolyammodellekkel egészítjük ki, amely bemutatja a rendszer egymáshoz kapcsolódó adatfeldolgozási lépéseit.

Az objektumosztály a közös attribútummal rendelkező objektumok halmazának (mint a szemantikus adatmodellekben) és az összes objektum által nyújtott szolgáltatásoknak és műveleteknek egy absztrakciója. Az objektumok végrehajtható egyedek, az objektumosztály attribútumaival és szolgáltatásaival. Az objektumok az objektumosztály példányai, és egy osztályból számos különböző objektum készíthető. Általánosságban az elemzés segítségével fejlesztett modellek az objektumosztályokra és azok kapcsolataira helyezik a hangsúlyt.

Az objektumorientált követelményelemzés során a valós világbeli egyedeket objektumosztályokkal ajánlott modelleznünk. Nem szabad az egyes objektumok (osztályok példányai) részleteit belevennünk a rendszerbe. Az objektummodelleknek különféle típusait állíthatjuk elő, amelyek megmutatják, hogyan viszonyulnak egymáshoz az objektumosztályok, hogyan aggregálnak objektumok más objektumokat, hogyan érintkeznek objektumok más objektumokkal és így tovább. Ezek mind egyedi információt adnak a specifikált rendszerről.

Az objektumok és objektumosztályok azonosítására szolgáló elemzési folyamatot az objektumorientált fejlesztés egyik legnehezebb területének tartják. Az objektumazonosítás alapvetően ugyanaz az elemzésnél és a tervezésnél. Használhatók az objektumazonosításnak az objektumorientált tervezésről szóló, 14. fejezetben ismertetett módszerei. Itt néhány olyan objektummodellre helyezzük a hangsúlyt, amelyek az elemzési folyamat során generálhatók.

Az objektumorientált elemzésnek különféle módszerei keletkeztek a 90-es években (Coad és Yourdon, 1990; Rumbaugh *és mások*, 1991; Jacobsen *és mások*, 1993; Booch, 1994). Ezekben a módszerekben igen sok közös volt, ezért a legfőbb fejlesztők közül hárman (Booch, Rumbaugh és Jacobsen) elhatározták, hogy egységesítik megközelítéseiket, és létrehoznak egy egységes módszert (Rumbaugh *és mások*, 1999b). Az így létrejött Unified Modeling Language (Egységes modellezőnyelv, UML) az objektummodellezés szabványává vált. Az UML különböző típusú rendszermodellekhez is tartalmaz jelöléseket. Korábbi fejezetekben már láttunk használatieset-modelleket és szekvenciadiagramokat, e fejezet korábbi részében pedig állapotátmenet-modelleket.

Az objektumosztályokat UML-ben, ahogyan azt a 8.10. ábrán szereplő példa is mutatja, három részre osztott, álló téglalappal jelölik:

1. A felső részben áll az objektumosztály neve.
2. Az osztály attribútumai a középső részben vannak.
3. Az objektumosztályhoz kapcsolt műveletek a téglalap alsó részében helyezkednek el.

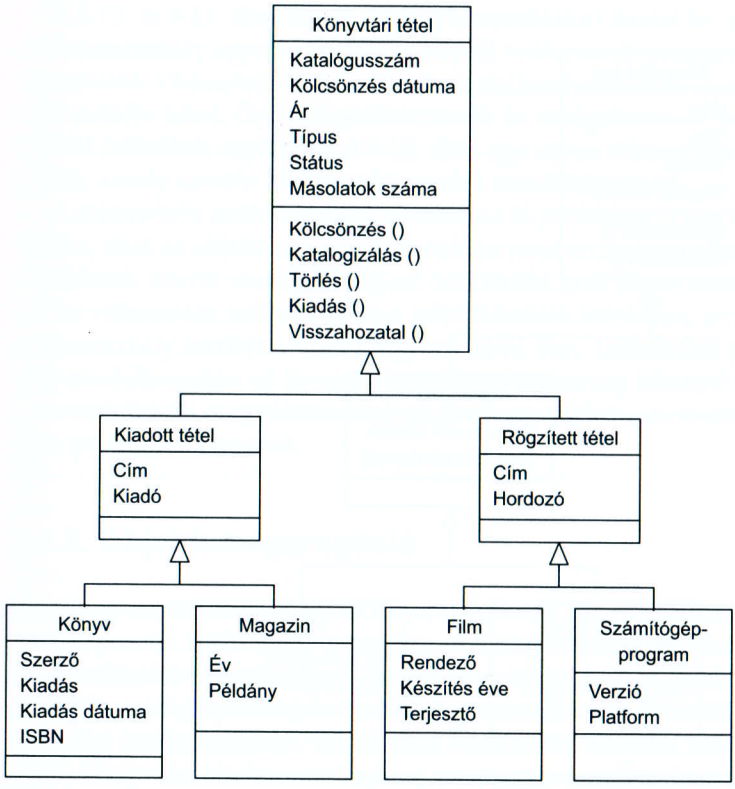
A könyv terjedelme nem elegendő arra, hogy itt a teljes UML-t ismertessem, így most háromfajta modellt emelek ki: az objektummodelleket, amelyek megmutatják, hogyan osztályozhatók az objektumok, valamint hogyan örökölnek attribútumokat és műveleteket más objektumoktól; az aggregációs modelleket, amelyek azt mutatják meg, hogyan kezelhetők összetett objektumok; továbbá egyszerű viselkedési modelleket, melyek az objektumok interakcióit mutatják be.

8.4.1. Öröklődési modellek

Az objektumorientált modellezés magában foglalja a tanulmányozott szakterület fontos objektumosztályainak azonosítását. Ezeket utána taxonómiába szervezzük. A taxonómia egy osztályozási séma, amely megmutatja, hogyan kapcsolódik egy osztály más osztályokhoz közös attribútumokon és szolgáltatásokon keresztül.

A taxonómia megjelenítéséhez az osztályokat öröklődési hierarchiába szervezzük, ahol a hierarchia csúcsán a legáltalánosabb objektumosztályok állnak. A specializáltabb objektumok öröklik ezek attribútumait és szolgáltatásait. Ezeknek a specializált objektumoknak saját attribútumaik és szolgáltatásaik is lehetnek.

A 8.10. ábrán egy könyvtár modelljének a leegyszerűsített osztályhierarchiájának egy része látható. Ez a hierarchia a könyvtárban tárolt tételekről közöl in-



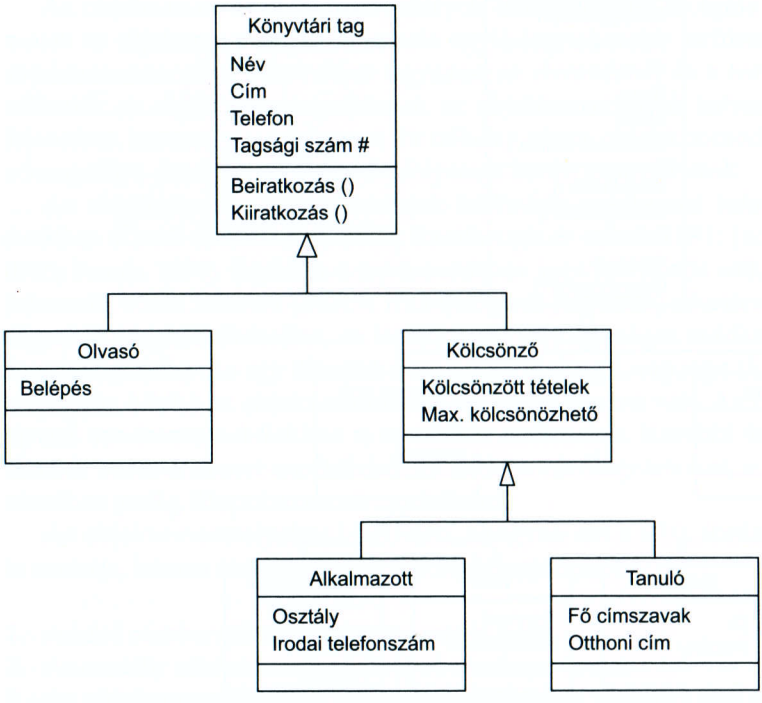
8.10. ábra. Egy könyvtár osztályhierarchiájának részlete

formációt. A könyvtárban különböző típusú tételeket tárolnak: könyveket, zenét, filmfelvételeket, magazinokat, folyóiratokat stb. A 8.10. ábrán a legáltalánosabb tétel a fa gyökerében áll, és az összes könyvtári tételben közös attribútumokat és szolgáltatásokat tartalmazza. Ezeket a Kiadott tétel és a Rögzített tétel osztályok öröklik, melyek hozzáteszik a saját attribútumaikat, amiket aztán az alacsonyabb szinten lévő osztályok örökölnek.

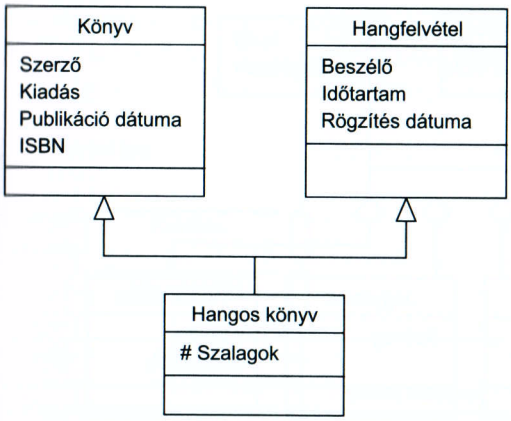
A 8.11. ábra egy másik olyan öröklődési hierarchiára mutat példát, amely a könyvtári modell részét képezheti. Ez a könyvtári tagokat mutatja. A tagoknak két osztálya van: azoké, akik könyvet kölcsönözhetnek, és azoké, akik csak a könyvtárban olvashatnak könyveket és nem vihetik ki azokat.

Az UML jelölésében az öröklődés „felfelé" mutat, nem pedig „lefelé", mint néhány más objektumorientált jelölésben, vagy mint a Javában, ahol az alosztályok örökölnek a szuperosztályoktól. Vagyis a nyílhegy (háromszög alakban) az attribútumokat és műveleteket öröklő osztálytól a szuperosztály felé mutat. Az *öröklődés* szó helyett az UML inkább a *generalizációs kapcsolat* elnevezést használja.

Az osztályhierarchiákat megtervezni nem könnyű, így az elemzőknek részletesen meg kell érteniük a szakterületet, melyben a rendszert telepíteni fogják. A gyakorlatban előforduló problémák bonyolultságára példaként tekintsük a könyvtári tételek hierarchiáját. Elsőre úgy tűnhet, hogy a Cím attribútumot a legáltalánosabb tételben kell tárolni, és az alacsonyabb szinten lévő tételek majd öröklik azt.



8.11. ábra. A könyvtári tagok osztályhierarchiája



8.12. ábra. Többszörös öröklődés

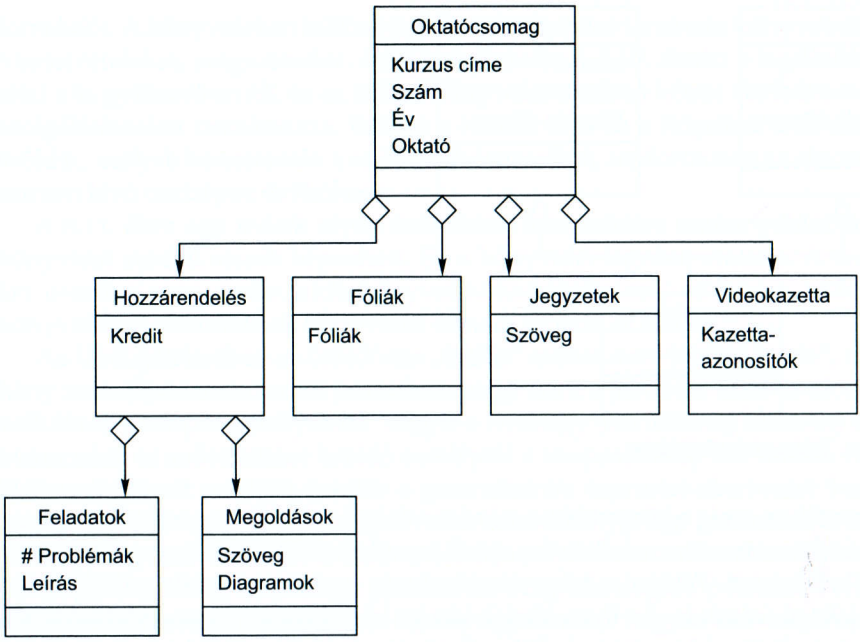
Ugyanakkor, amíg a könyvtárban mindennek kell valamilyen azonosítóval vagy regisztrációs számmal rendelkeznie, ebből nem következik az, hogy mindennek címe is kell legyen. Például a könyvtár tárolhatja egy visszavonult politikus személyes jellegű írásait is. Az ilyen tételek között lehet számos olyan, például levél, ami nincs explicit módon címmel ellátva. Ezeket egy másik osztályba fogjuk sorolni (amit itt nem mutatok be), amelynek különböző attribútumai vannak.

A 8.10. és 8.11. ábra olyan osztályhierarchiákat mutat be, melyekben minden objektumosztály egyetlen szülőosztálytól örökli attribútumait és műveleteit. Létrehozhatók többszörös öröklődéssel rendelkező modellek is, ahol egy osztálynak több szülője lehet. Örökölt attribútumaik és szolgáltatásaik az egyes szülőosztályoktól örököltek együttese. A 8.12. ábra egy olyan többszörös öröklődésre mutat példát, amely szintén a könyvtári modell részét képezheti.

A többszörös öröklődéssel kapcsolatos fő probléma olyan öröklődési gráf tervezése, ahol az objektumok nem öröklik a nem szükséges attribútumokat. Egyéb problémák között szerepel még az öröklődési gráf átszervezésének nehézkessége, ha változtatás szükséges, és a névütközések feloldása, amikor két vagy több szuperosztály attribútumának egyező neve van, különböző jelentéssel. A rendszer modellszintjén az ilyen ütközéseket viszonylag könnyű feloldani az objektummodell kézi megváltoztatásával. Ezek az objektumorientált programozásban több problémát okoznak.

8.4.2. Objektumaggregáció

Bizonyos objektumok, hasonlóan ahhoz, hogy más objektumokkal való öröklődési kapcsolat útján attribútumokra és szolgáltatásokra tehetnek szert, más objektumokból is felépülhetnek. Vagyis egy objektum más objektumok halmazának *aggregátuma*. Az ilyen objektumokat reprezentáló osztályok objektumaggregációs modellel modellezhetők, amint az a 8.13. ábrán látható. Ebben a példában egy olyan könyvtári tételt modelleztem, amely egy egyetemi kurzus oktatócsomagja.



8.13. ábra. Egy tantárgyat reprezentáló aggregátor objektum

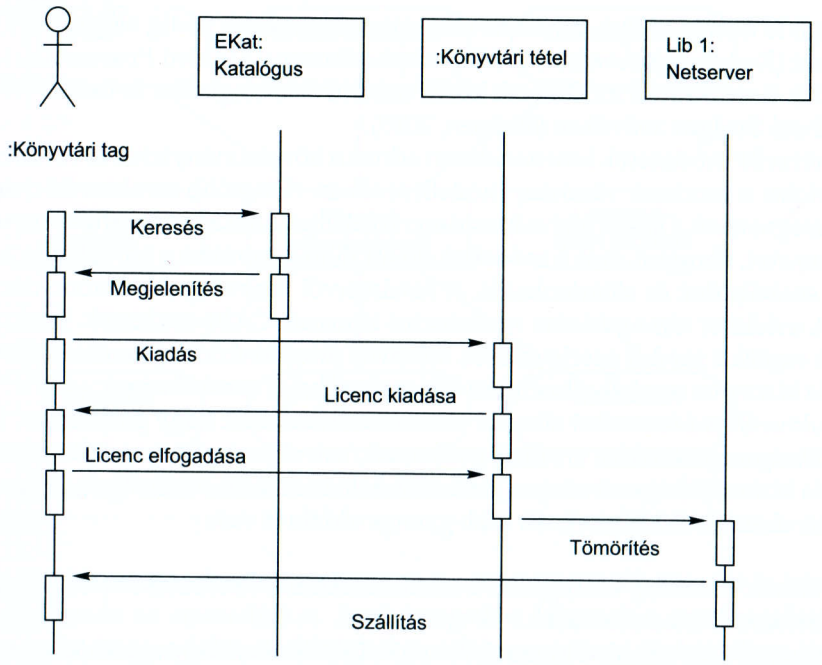
Ez az oktatócsomag tartalmazza a jegyzeteket, feladatokat, mintamegoldásokat, az órákon használt fóliák másolatait és videokazettákat.

Az UML jelölésében az aggregációt a kapcsolat forrásánál elhelyezett rombusz jelenti. Ezért a 8.13. ábra olvasható úgy, mint „Egy oktatócsomag, ami egy vagy több hozzárendelésből, fóliából, jegyzetből és videokazettából áll".

8.4.3. Objektumok viselkedésének modellezése

Az objektumok viselkedésének modellezéséhez be kell mutatnunk, hogyan használják az objektumok által biztosított műveleteket. Az UML-ben a viselkedést forgatókönyvek segítségével modellezzük, amiket az UML (7. fejezetben tárgyalt) használati eseteivel ábrázolunk. A viselkedés modellezésének egyik módja az UML-szekvenciadiagramok használata, amelyek bemutatják a használati esetben lezajló műveletsorozatot. Ezek hasonlóak a szekvenciadiagramokhoz, ezért ezekkel itt nem foglalkozom külön.

A 8.14. ábrán láthatjuk, hogyan használhatók a szekvenciadiagramok a viselkedés modellezésére. Az ábra egy LIBSYS-rendszerbeli használati esetet bont ki, melyben a tagok elektronikus formában kérnek le könyveket a könyvtárból. Képzeljünk el egy olyan szituációt, ahol a 8.13. ábrán látható oktatócsomagokat elektronikus formában gondozzák, és azok letölthetők a hallgató számítógépére.



8.14. ábra. Elektronikus tételek kiadása

Egy szekvenciadiagramon az objektumok és aktorok a diagram tetején egy sorban helyezkednek el. A címkézett nyilak műveleteket jelölnek; a műveletek sorban felülről lefelé következnek. Ebben a forgatókönyvben a könyvtári tag eléri a katalógust, hogy megnézze, hogy a kívánt tétel elektronikusan elérhető-e; ha igen, kéri a tétel elektronikus példányát. Szerzői jogi okokból ezt licencelni kell, így létrejön egy tranzakció a tétel és a tag között, amely során a tag a licencet elfogadja. A kiadandó tétel azután egy hálózati szerver objektumhoz küldődik, amely tömöríti azt, majd elküldi a könyvtári tagnak.

A 7.8. ábrán egy használati esetet kifejtő szekvenciadiagramra másik példát is találhatunk, amely egy cikk nyomtatásakor végbemenő műveletsorozatot ír le.

8.5. STRUKTURÁLT MÓDSZEREK

Egy strukturált módszer egy meglévő vagy megépítendő rendszer modelljei elkészítésének szisztematikus módja. A hetvenes években fejlesztettek ki először strukturált módszereket, a szoftverelemzés és -tervezés támogatására (Constantine és Yourdon, 1979; Gane és Sarson, 1979; Jackson, 1983). Később, a nyolcvanas és kilencvenes években az objektumorientált fejlesztést támogató strukturált módszereket fejlesztettek ki (Rumbaugh *és mások,* 1991; Robinson, 1992; Jacobsen *és mások,* 1993; Booch, 1994). Ezeket az objektumorientált módszereket egyesítették, szabványos modellezőnyelvként az UML-t ajánlva (Booch *és mások,* 1999;

Rumbaugh *és mások*, 1999a), hozzá strukturált módszerként pedig előbb a Unified Processt (Rumbaugh *és mások*, 1999b), majd a Rational Unified Processt (Krutchen, 2000). E strukturált módszerek közül számos összefoglalása és összevetése megtalálható Budgen művében (Budgen, 2003).

A strukturált módszerek keretrendszert adnak a követelményfeltárás és -elemzés részeként a rendszer részletes modellezéséhez. A legtöbb strukturált módszernek megvannak a maga kedvelt rendszermodelljei. Általában meghatároznak egy folyamatot, ahogyan ezek a modellek előállíthatók, továbbá a modellekre vonatkozó szabályokat és előírásokat is. A rendszerről szabványos dokumentáció készül. A módszer támogatására rendszerint léteznek CASE-eszközök. Ezek az eszközök segítik a modell szerkesztését, valamint programkód és jelentések generálását, és bizonyos modell-ellenőrzési képességekkel is rendelkeznek.

A strukturált módszereket sikerrel alkalmazták már sok nagy projektben. Jelentős költségmegtakarítást eredményezhetnek, mivel szabványos jelölést használnak, és biztosítják a szabványos tervezési dokumentáció elkészítését. Ugyanakkor a strukturált módszereknek több gyenge oldala is van:

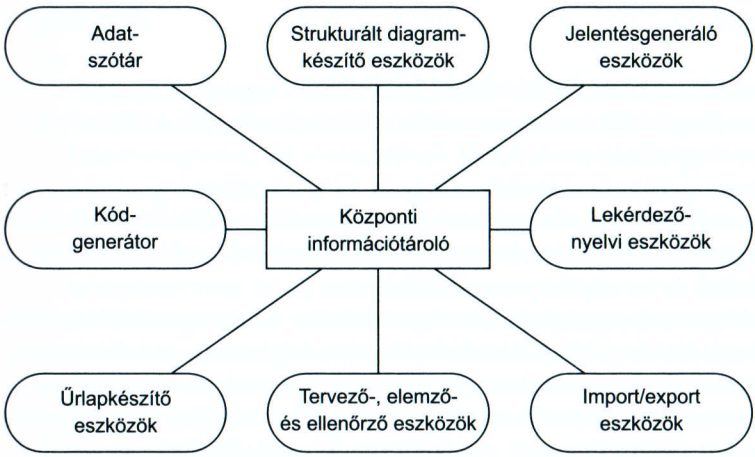
1. Nem adnak hatékony támogatást a nemfunkcionális rendszerkövetelmények megértésére és modellezésére.
2. Rendszerint nem adnak útmutatást annak eldöntésére, hogy egy módszer alkalmas-e egy adott problémára vagy sem. Ahhoz sem adnak tanácsot, hogy hogyan lehet azokat adott környezetben alkalmazni.
3. Gyakran túl sok dokumentációt állítanak elő. A rendszerkövetelmények lényeges elemei elsikkadnak a töménytelen részlet között.
4. Az elkészült modellek nagyon részletesek, és a használók számára gyakran nehezen érthetők. Ezért nem is tudják ezek hitelességét ellenőrizni.

A gyakorlatban ugyanakkor a követelménytervezők és a szoftvertervezők nem csupán egy adott módszer által javasolt modellekre szorítkoznak. Például az objektumorientált módszerek általában nem javasolják adatfolyammodellek elkészítését. Viszont tapasztalataim alapján ezek a modellek gyakran igen hasznos részei a követelményelemzési folyamatnak, mivel átfogó képet adnak az adatfeldolgozásról az elejétől a végéig. Szintén közvetlenül hozzájárulhatnak az objektumok (az áramló adatok) és műveleteik azonosításához (a transzformációk).

Az elemzési és tervezési CASE-eszközök támogatják a strukturált módszerekben használt grafikus jelölések létrehozását, szerkesztését és elemzését. A 8.15. ábra bemutatja azokat a komponenseket, amiket egy ilyen módszertámogató környezet magában foglalhat.

Ahogy a 8.15. ábrán látható, az átfogó módszertámogató eszközök rendszerint a következőket tartalmazzák:

1. *Diagramszerkesztők* objektummodellek, adatmodellek, viselkedési modellek stb. létrehozására. Ezek a szerkesztők nem csupán rajzolóeszközök, hanem figyelnek a diagrambeli egyedek típusára is. Ezekről az egyedekről információt gyűjtenek be, és azt a központi tárolóban tárolják.



8.15. ábra. Egy CASE-eszköz komponensei strukturált módszer támogatására

2. *Tervelemzési és ellenőrző eszközök,* melyek feldolgozzák a tervet és jelentik a hibákat és az anomáliákat. Ezek integrálva lehetnek a szerkesztőrendszerrel azért, hogy a felhasználói hibákra a folyamat korai szakaszában fény derüljön.
3. *Tárolólekérdező nyelvek,* amelyek lehetővé teszik, hogy a tervező a tárolóban terveket és kiegészítő tervezési információkat találjon.
4. Egy *adatszótár,* amely a rendszertervben használt egyedekről tárol információt.
5. *Jelentésgeneráló eszközök,* amelyek a központi tárolóból vesznek információt, és automatikusan generálnak dokumentációt a rendszerről.
6. *Űrlapkészítő eszközök,* amelyek lehetővé teszik képernyő- és dokumentumformátumok megadását.
7. *Import/export eszközök,* amelyek lehetővé teszik az információcserét a központi adatbázis és más fejlesztőeszközök között.
8. *Kódgenerátorok,* amelyek a központi tárolóból vett tervből automatikusan kódot vagy kódvázat generálnak.

A legátfogóbb CASE-eszközök lehetővé teszik program vagy programrészlet generálását a rendszermodell által biztosított információból. A CASE-eszközök sokszor több nyelvet is támogatnak, így ugyanabból a tervezési modellből C-, C++- vagy Java-programot is generálhatunk. Mivel a modellek nem tartalmaznak alacsony szintű részleteket, a tervezési eszközrendszerben lévő kódgenerátor rendszerint nem képes a teljes rendszer generálására. Némi kézi kódolás általában szükséges a generált program kiegészítéséhez.

Kulcsfogalmak

- A modell a rendszer absztrakt nézete, mely mellőzi a rendszer bizonyos részleteit. Egymást kiegészítő rendszermodellek is kifejleszthetők különböző információk bemutatására a rendszerről.
- A környezeti modellek bemutatják, hogyan helyezkedik el a modellezett rendszer más rendszerek és folyamatok környezetében. Kijelölik a rendszer határait. Környezeti modellként használhatók az architekturális modellek, a folyamatmodellek és az adatfolyammodellek.
- Adatfolyam-diagramokat használhatunk a rendszer által végzett adatfeldolgozás modellezésére. Ez a modellezett rendszert adattranszformációk halmazaként mutatja be, ahol az adatokon függvények végeznek tevékenységet.
- Az állapotátmenet-modelleket a rendszer belső, illetve külső eseményekre adott válaszaként megnyilvánuló viselkedésének modellezésére használják.
- A szemantikus adatmodellek a rendszerbe kerülő, illetve abból távozó adatok logikai szerkezetét írják le. Ezekben a modellekben leírhatók a rendszer egyedei, azok tulajdonságai és a kapcsolatok, melyekben részt vesznek.
- Az objektummodellek a rendszer logikai egyedeit, azok osztályozását és aggregációját írják le. Az adatmodellt a feldolgozási modellel ötvözik. A kifejleszthető lehetséges objektummodellek az öröklődési modellek, az aggregációs modellek és a viselkedési modellek.
- Egy rendszer aktorai és objektumai közötti interakciókat bemutató szekvenciamodellek a dinamikus viselkedés modellezésére szolgálnak.
- A strukturált módszerek keretrendszert biztosítanak a rendszermodellek fejlesztésének támogatásához. Rendszerint kiterjedt CASE-eszköz támogatással bírnak, beleértve a modellszerkesztést és -ellenőrzést valamint a kódgenerálást.

További irodalom

Software Design, 2nd ed. Bár ez a könyv elsősorban a szoftvertervezésről szól, a szerző számos strukturált módszert is tárgyal, amely a követelménytervezési folyamatban szintén használható. Nem csak az objektumorientált megközelítéssel foglalkozik. (Budgen, D., 2003, Addison-Wesley.)

Requirements Analysis and System Design. A könyv információs rendszerek elemzésével foglalkozik, és tárgyalja, hogyan használhatók a különböző UML-modellek az elemzési folyamatban. (Maciaszek, L., 2001, Addison-Wesley.)

Software Engineering with Objects and Components. Röviden, olvashatóan ismerteti az UML használatát a rendszer specifikációja és tervezése alatt. Bár nem annyira átfogó, mint a teljes UML-leírások, ez a könyv sokkal jobb a jelölés megtanulására és megértésére. (Stevens, P. és Pooley, R., 1999, Addison-Wesley.)

Feladatok

8.1. Rajzolja meg egy kórház betegnyilvántartó rendszerének környezeti modelljét. Az egyéb elérhető kórházi rendszerekről bármilyen ésszerű feltételezést megtehet, de a modellnek tartalmaznia kell egy betegfelvételi rendszert, egy röntgenfelvétel-tároló rendszert és más diagnosztikai rekordokat.

8.2. A banki ATM-ekről szerzett tapasztalata alapján rajzolja meg azt az adatfolyam-diagramot, amely azt a folyamatot modellezi, amely során az ügyfél készpénzt vesz fel az automatából.

8.3. Modellezze azt az adatfeldolgozást, mely egy elektronikus levelezőrendszerben mehet végbe. A levélküldési és levélfogadási folyamatot külön modellezze.

8.4. Rajzolja meg a következő rendszerek ellenőrző szoftvereinek állapotátmenet-modelljeit:

- egy automata mosógépét, amely a különböző típusú ruhákhoz különböző programokat biztosít;
- egy DVD-lejátszó szoftverét;
- egy üzenetrögzítőét, amely felveszi a beérkező üzeneteket és a számukat kijelzi egy LED-en. A rendszernek lehetővé kellene tennie azt is, hogy a telefontulajdonos a saját számát bárhonnan felhíva és egy (hangokként azonosított) számsort beütve a felvett üzeneteket távolról visszahallgathassa.

8.5. Egy szoftverrendszer ábrázolható irányított gráfként, ahol a csomópontok a rendszerbeli egyedek, az élek pedig az egyedek közötti kapcsolatok. A modell egyedei és kapcsolatai névvel és egyéb információval címkézhetők fel. Minden egyednek van típusa és kifejthető egy almodellben. Rajzoljon egy adatmodellt, amely leírja egy szoftverrendszer modelljének struktúráját.

8.6. Modellezze azokat az objektumosztályokat, melyek egy elektronikus levelezőrendszerben felhasználásra kerülhetnek. Ha foglalkozott a 8.3. feladattal, írja le az adatfeldolgozási modell és az objektummodell közötti hasonlóságokat és különbségeket.

8.7. A 8.8. ábrán szereplő rendszeradatok információit felhasználva rajzoljon szekvenciadiagramot, amely bemutatja egy új cikk katalogizálásának lehetséges műveletsorozatát a LIBSYS-rendszerben.

8.8. Fejlesszen ki egy osztályhierarchia-diagramot és egy aggregációs diagramot tartalmazó objektummodellt, amely bemutatja egy személyi számítógépes rendszer alapvető összetevőit és annak rendszerszoftverét.

8.9. Fejlesszen szekvenciadiagramot, amely bemutatja azokat az interakciókat, melyek egy hallgató egyetemi kurzusra való regisztrációjakor történnek. A kurzusokon létszámkorlát is lehet, így a regisztrációs folyamatnak a szabad helyek számára vonatkozó ellenőrzést is tartalmaznia kell. Tételezze fel, hogy a hallgató a felvehető kurzusokról egy elektronikus kurzuskatalóguson keresztül is érdeklődni tud.

8.10. Milyen körülmények között nem javasolná strukturált módszerek használatát a rendszerfejlesztéshez?