Az üzleti modellek MDA-alapú transzformációja
Objektum- és architektúraszemléletű fejlesztési stratégia

DR. RAFFAI MÁRIA
Széchenyi István Egyetem, Műszaki Tudományi Kar, Informatika Tanszék
egyetemi docens, főiskolai tanár
raffai@sze.hu

ABSTRACT

With regards to the IS/IT strategies and the new IS requirements the application developing objectives are followed by, in descending order of importance, the corporate IT-intent to purchase off-the-shelf applications whenever possible, and the mandates to build systems capable to interoperate with legacy data and processes. But this is not only a primary demand as the application developing tools and the supply are mature enough for handling the multiple expectations for the truly distributed enterprise computing. The IT platform diversity is not an exception anymore, but it is already a rule to provide an easy and clear path to multi-tiered application development and deployment across the enterprise and beyond. In my article I intend

- *to discuss the newest solutions (paradigms, methods, tools, standards) helping us to create effective applications which are able to interoperate the different types and level of the legacy systems and the newly developed or purchased applications, and*
- *to describe the application development process supported by a framework which helps to model the business environment, the functionality and behavior of the system, the information flow and the artifacts of a development life cycle.*

Vállalati alkalmazásintegráció

Az információrendszerek vonatkozásában az alkalmazásfejlesztési munka, új IT-támogatott rendszerek építése napjaink egyik legnagyobb kihívása, különösen akkor, ha az információt erőforrásként tekintjük, és tudatában vagyunk annak a ténynek, hogy a vállalkozások tevékenysége, versenyképessége és így sikere is a korszerű technológiák alkalmazásán alapul. Nem hagyhatjuk azonban figyelmen kívül azt a fontos szempontot sem, hogy a vállalkozások meg kell, hogy őrizzék a korábban felhalmozott értékeket, azt a tőkét, amit az informatikai rendszerek jelentenek. Az évtizedek alatt felgyülemlett tudás azonban nemcsak a dolgozók fejében, a lefektetett dokumentumokban és elért eredményekben van, hanem az informatikai rendszerekben, az adatbázisokban is. A tulajdonosok és a CEO tehát az információrendszer és az informatikai infrastruktúra vonatkozásában fontosnak tartja

- az IR/IT-tőke megőrzését,
- a különböző (örökölt és újonnan vásárolt, illetve fejlesztett) rendszerek együttműködésének a biztosítását vállalaton belül és a vállalaton kívüli partnerekkel egyaránt, legyenek azok azonos vagy különböző platformon működő alkalmazások,
- az alkalmazások közötti korrekt adatátvitelt, a különböző generációjú rendszerek koncepcióját,
- az alkalmazások elvárt minőségének a biztosítását,

– az újabb fejlesztésekhez, illetve adaptációkhoz gyors fejlesztést biztosító módszerek és technikák rendelkezésre állását és alkalmazását, valamint
– az új technológiák versenyelőny érdekében történő kihasználását.

Mivel a megváltozott elvárások mellett egyéb tényezők is új megoldások kidolgozását teszik szükségessé, ezért az IR/IT-rendszerek vonatkozásában az alábbi sajátosságokat kell figyelembe venni:

– A meglévő, még korrekten működő alkalmazások különböző elveken alapulnak, eltérő algoritmusok szerint dolgoznak, és alapvetően nem az alkalmazások együttműködésére épülnek.
– Az örökölt rendszerek adatbázisai szintaktikai és szemantikai szempontok szerint is eltérő adat- és rekordformátumokat alkalmaznak.
– A meglévő rendszereket általában régi, mára már elavult szoftverkörnyezetre (operációs rendszer, protokoll, programozási nyelvek, adatbázis-kezelők, köztesréteg-megoldások stb.) fejlesztették.
– Az eltérő platformok különbözőképpen tesznek eleget a rendelkezésre állással, a rendszerek kezelhetőségével és az átviteli biztonsággal szemben támasztott felhasználói elvárásoknak, és eltérően reagálnak az együttműködési igényekre is.

Az említett problémák arra késztetik az elemzőket és a programfejlesztőket, hogy a meglévő infrastruktúra nagy részének megőrzése érdekében új elveket alkossanak, a korábbiaktól eltérő módszereket fejlesszenek, hogy lehetővé tegyék a különböző alkalmazások és platformok közötti átjárhatóságot, az adatbázisok tartalmának, az adatoknak azonos módon történő értelmezését. Bár már több évtizede léteznek eredményesen használható technológiák (file transfer, EDI stb), egy, a különböző szoftverrendszerek, az eltérő platformok közötti átjárhatóságot megvalósító megfelelő megoldás mégis sokáig váratott magára. Olyan szabványokra és keretrendszerre van ugyanis szükség, amely általánosan tesz eleget a fenti elvárásoknak, és egységesen biztosítja a különböző platformokon a fejlesztett rendszerek együttműködését [19].

Igény az integrációra

A vállalati szintű alkalmazásintegráció (EAI: Enterprise Application Integration) koncepcióját 20. század végén dolgozták ki. Az EAI valójában egy olyan IT-elv, amely szerint az alkalmazásfejlesztési módszereket, technikákat és eszközöket egyesíteni kell annak érdekében, hogy valós vállalati környezetben alkalmas legyen az örökölt és az új alkalmazások integrálására. Tisztában kell azonban lenni azzal, hogy a szakemberek egy ilyen megoldást akkor tudnak igazán eredményesen használni, ha az szabványos, és ha az alkalmazók többsége él a szabvány nyújtotta lehetőséggel.

A fenti problémák megoldására a vezető szoftverfejlesztő cégek egy általánosan használható, új megoldás kifejlesztésén kezdtek el dolgozni, és a szabványosításért felelős OMG (Object Management Group Task Force) a múlt évben elfogadta az integrációs szabványt. Az MDA (Model Driven Architecture) egy innovatív megközelítés a vállalati rendszerek absztrakt modelljének megalkotására, az üzleti igények platform- és technológiafüggetlen specifikálására. Az MDA keretrendszer egyértelműen szétválasztja az üzleti funkciókat azok számítógépes megvalósításától [1], és ezáltal gyors alkalmazásfejlesztést tesz lehetővé [14]. Az üzleti modell megalkotását követően egyértelműen meghatározhatók az interfészek, és a platformfüggetlen modell egy megvalósítható, rendszerfüggő modellé, szoftverarchitektúrává transzformálható [5].

Az MDA azonban, mint egy, az elvárásoknak megfelelő szabvány, követi és magában foglalja a legtöbb létező ipari szabványt, biztosítva a fejlesztett szoftvertermék rugalmasságát és továbbfejleszthetőségét, valamint a későbbi fejlesztésekhez történő egyszerű adaptálhatóságát. A legfontosabb azonban talán az a tény,

hogy az MDA-elvek követésével az üzleti szakemberek és a szoftverfejlesztők egyaránt az üzleti problémákra, a kibocsátásra, az üzleti kockázatokra tudnak koncentrálni, és a legjobb megoldást a meglévő technológiák leghatékonyabb kihasználásával érik el.

Mi is az MDA szabvány?

A ’90-es évek elején az OMG a programjának a középpontjába az együttműködő elemekből álló, integrált számítógépes környezet megvalósítását állította. Csaknem egy évtized telt el azóta, hogy számos, a céloknak eleget tevő szabványos megoldás kidolgozása és elfogadása után, az OMG egyértelműen definiálta, hogy a szoftverfejlesztési folyamatban a fejlesztőknek a támogatandó rendszer viselkedésére és funkcionalitására kell fókuszálniuk. Az MDA kimondja, hogy az üzleti feladatok modellezését el kell választani a megvalósítás részleteitől, vagyis a fejlesztést a szakterület modellnézeteinek kialakításával kell kezdeni, figyelmen kívül hagyva ebben a folyamatrészben az implementációs technológiai környezetet [4]. Bár ez, a rendszerfejlesztési munka kezdeteitől igaz elv *nem új dolog,* mégis csak most érkezett el az idő, hogy ez a nagyon fontos szempont valóban érvényesülni tudjon. Olyan megoldásokra volt ugyanis szükség (például XML/SOAP), amelyek lehetővé tették, hogy az azonos viselkedésű és funkcionalitású üzleti modellek elemeit más fejlesztésekben is fel lehessen használni. A probléma megoldását a köztesréteg-szoftverek jelentik, amelyek felelősek a különböző platformokon működő rendszerek (együttműködő szervezeti egységek, szállítók, ügyfelek rendszerei) kooperációjának a biztosításáért [17].

Az MDA-alapú megközelítésben a modell egy adott rendszernek az elemeire, a struktúrájára, a viselkedésére és a funkcionalitására vonatkozó információkat reprezentálja. A támogatott rendszer modelljének és a működést implementáló számítógépes rendszer részleteinek a szétválasztására az MDA két modellcsomagot definiál:

1. Az egyik a megvalósítás eszközeitől és módjától független ún. *platformfüggetlen modell* (PIM: Platform Independent Model),

2. a másik pedig a megoldás módját figyelembevevő, a technikai részleteket leíró, az implementációt megvalósító *platformspecifikus modell* (PSM: Platform Specific Model).

Egy vállalati alkalmazásfejlesztési munkánál tehát mindenekelőtt létre kell hozni azt a PIM-modellt, amely metaadatokat tartalmaz a valós rendszer elemeiről, ezek kapcsolatairól, struktúrájáról, az elemek viselkedéséről, feladatairól és interfészeikről, valamint az elemek között fennálló függőségi viszonyról. A PIM--modell létrehozása egy iteratív folyamatban történik, amelyben az elemek sajátosságainak fokozatos finomításával (definiálás, felülvizsgálat, kiértékelés, javítás) jutunk el a végső modellhez. A folyamat végeredménye korrekt kell legyen, hiszen ez képezi alapját annak a modellnek, amely a rendszer működését az alkalmazandó technológiai környezetben ténylegesen megvalósítja.

A PIM-modell PSM-modellé történő transzformációja során a modellelemekhez hozzárendeljük az implementáció modelljeit, majd a működtetésre vonatkozó információkat, létrehozva ezzel az üzemeltetés modelljét is. Az átalakítási munkához (technológiafüggetlen, konceptuális modellből számítógépes környezeti lehetőségek által meghatározott implementációs és működési modellt transzformálunk) számos algoritmus és szabványos megoldás áll rendelkezésre [20]. Az egyik, az 1997-ben szabványként elfogadott egységesített modellező nyelv, az UML (Unified Modeling Language), amely kiválóan alkalmas az üzleti domén sajátosságainak szemléletes specifikálására. A másik a doménspecifikus modellinformációkat leíró szabványos metamodell, a MOF (Meta Object Facility), a harmadik pedig a különböző adatbázisok egységes kezelését megvalósító adattárház-modell, a CWM (Common Warehouse Model).

Egy tipikus EAI-megoldásban a PSM-modell különböző specifikációk, feltételek, algoritmusok, forráskódok, interfészek, futtatási leírások formájában üzleti és platformspecifikus részleteket tartalmaz. Ha a fejlesztést egy már meglévő számítógépes környezetből kiindulva végezzük, akkor a fejlesztés kiindulási pontja a PSM-modell. Vannak megoldások, amelyekkel implementációk visszafejthetők. A *reverse engineering* lehetőséget ad arra, hogy működő alkalmazásokból meghatározzuk a támogatott rendszer üzleti funkcionalitását és viselkedését, vagyis PSM-modellekből PIM-modelleket generáljunk, majd szintén az iterativitás elvét alkalmazva az így transzformált PIM-modellt finomítsuk és a valós működéshez igazítsuk.

A PIM-modellé történő leképezés azonban nem egyszerű feladat, igazán csak akkor lehetséges, ha ehhez megfelelő technológiák állnak rendelkezésre, mint például kódvisszafejtés, elemspecifikálás, elemek kapcsolatrendszerét és a viselkedést modellező diagramok generálása forráskódból stb.

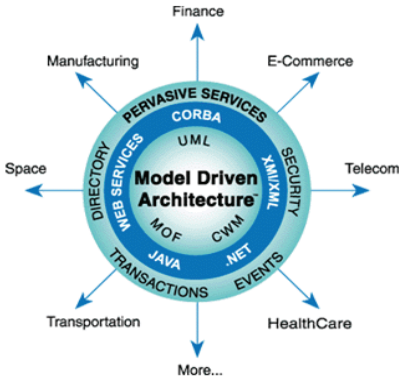
Összefoglalva megállapíthatjuk, hogy bár az MDA nem új elvet specifikál, érdeme mégis elvitathatatlan, hiszen kimondja a szoftverfejlesztés egyik legfontosabb paradigmáját. miszerint *külön kell választani a PIM- és PSM-modelleket, és a PSM-modellt az előzetesen tudatosan kialakított PIM-modellből transzformációval kell létrehozni*. Az MDA keretrendszert követve olyan rendszer fejleszthető, amelyben

- integrálhatók a múltbeli, a jelenlegi és a jövőbeli fejlesztések,
- a megvalósítást szolgáló, technológiafüggő modell a rendszer előzetesen megtervezett konceptuális modelljéből származik (PIM → PSM),
- a megvalósítás folyamatosan változó infrastruktúrája (hardverelemek, szoftvermegoldások, adatbázis-megvalósítások) nincs hatással az üzleti modellre (megvalósul a fizikai modellfüggetlenség), és
- meghosszabbodik a szoftver élettartama, alacsonyabbak az előállítás és a karbantartás költségei, gyorsabb a szoftvertermék fejlesztési/beszerzési ráfordításainak megtérülése (ROI: Return of Investment).

Az MDA architektúrája

Az MDA-szabvány egy keretrendszerben egyesíti a korábban elfogadott modellezési és köztesréteg-szabványokat, nyitva hagyva a lehetőséget a később megjelenő szabványok beágyazására is. Az alapvető cél, hogy modellszinten biztosítsa a különböző elemek együttműködését, függetlenül az alkalmazott technológiáktól és alkalmazási szintektől [18]. Tekintettel arra, hogy az MDA magját az osztott vállalati informatikai infrastruktúra technológiafüggetlen modelljének a meghatározása képezi, ezért a jelenleg működő különböző architektúrák elveihez igazodik.

Az 14. ábra az MDA elemeit és struktúráját szemlélteti. Látható, hogy a kulcsszabványok az UML, a MOF és a CWM, amelyek a köztesréteg-szabványokkal (CORBA, XMI/XML stb.) kapcsolódnak a különböző alkalmazások együttműködését biztosító rendszerszolgáltatásokhoz. Az alkalmazási szoftverek (például termelésirányítási rendszer, betegfigyelő rendszer, banki ügyfélkiszolgáló rendszer, elektronikus értékesítési rendszer) alapját a szakterület funkcióit leíró platformfüggetlen PIM-modell képezi. A PIM-modellben az alkalmazások által nyújtott szolgáltatásokat, az egyik alkalmazástól a másik alkalmazásig vezető útvonalakat egymással kapcsolatban lévő modellrészletek fejezik ki, hidat képezve ezzel a megvalósításban különböző platformokon futó implementációk között. Az MDA-elvet követve tehát az alkalmazások szolgáltatásait és képességét, valamint az együttműködési igényt a PIM-modellben kell specifikálni, az implementációs modellt pedig ebből a platformfüggetlen modellből származtatjuk.



14. ábra Az MDA architektúrája

Annak érdekében, hogy megértsük az MDA lényegét, és ki tudjuk használni a benne rejlő lehetőségeket, fontos ismerni a komponens-szabványokat (UML, MOF, CWM, XMI, CORBA), valamint az MDA által nyújtott szolgáltatásokat.

Az MDA kulcsszabványai

Az UML mint modellező szabvány

Az UML modellező nyelvet a vezető szoftverfejlesztő cégek dolgozták ki annak érdekében, hogy egy széleskörűen alkalmazható, különböző szakterületeket képviselő emberek által is könnyen érthető, grafikus szimbólumokat használó, vizuális megoldást szolgáltassanak az üzleti folyamatoknak és a rendszer viselkedésének a modellezéséhez [13]. Az UML-t az OMG 1997-ben fogadta el szabványként. Az UML architektúráját három fő részben specifikálták: (1) alapvető modellezési elveket és szemantikát kifejező *modellelemek,* (2) az elemek ábrázolására szolgáló *grafikus szimbólumok* és (3) a használatra vonatkozó *szabályrendszer* [2].

Mivel az első szabványverziónak (UML Version 1.4) nagyon sok hiányossága volt, ezért egy alapos felülvizsgálatra, átdolgozásra és kiegészítésre volt szükség [13],[8]. A kidolgozásért felelős U2 Partners Consortium az UML 2.0 változatot már az MDA-szabványhoz igazította, és egy olyan verziót fejlesztett ki, amely mind a felhasználók, mind pedig a szoftverfejlesztők számára egyszerűbb használatot és könnyebb megértést, kommunikációt jelent. A nyelvben végrehajtott legfontosabb változások a szemantikai szabályokra, a szimbólumok használatára és a kiterjesztési mechanizmusban definiált lehetőségekre egyaránt vonatkoznak [9]. A legfontosabb változtatásokat az alábbiakban összegezhetjük:

– Az architektúramodellezési koncepció illeszkedik a MOF-szabványhoz, lehetővé téve a különböző modellnézeteknek, így a use case-eknek és a viselkedést leíró modelleknek (szekvencia, együttműködés, aktivitás és állapotátmenet), valamint komponenseknek és a fizikai megvalósítás csomópontjainak az összekapcsolását [4].

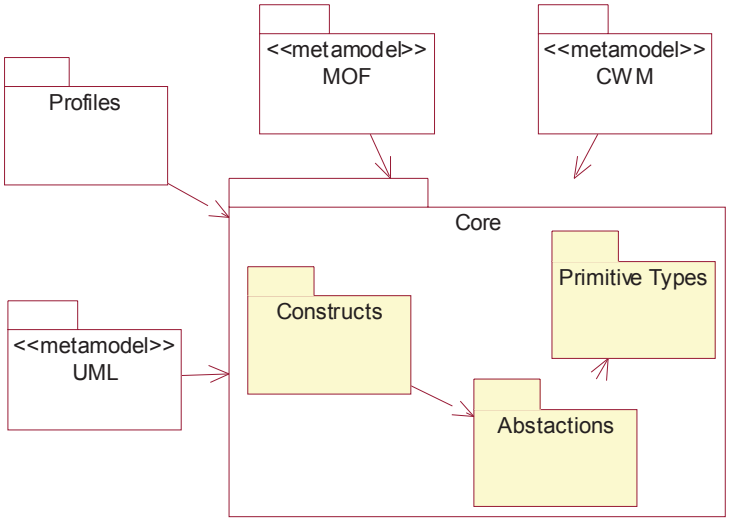
– Az aktivitásdiagram szemantikáját egyértelműen szétválasztották az állapotátmeneti diagram szemantikájától. Új szabályokat vezettek be, miszerint a szekvenciadiagramban modellezhetők az alternatív útvonalak és a párhuzamosan végzendő műveletek, az állapotdiagramban pedig szemléltethető az általánosítás művelete.

– A komponenstervezést interfészalapokra helyezték, ami azt jelenti, hogy a komponensekhez az inputigényeket és az outputszolgáltatásokat egyaránt diagramszinten lehet hozzárendelni, és megoldások vannak a beágyazott komponensek modellezésére is.

– A kiterjesztési mechanizmust a négyrétegű metamodellhez illesztették.

  - Az UML-specifikáció magját a szintaktikai és a szemantikai specifikációk képezik [8], pontosan definiálva
  - a modellnézetek kapcsolatát (UML CORBA facility, XMI DTD),
  - a nyelvi kiterjesztéseket (UML Standard Profiles), valamint
  - a feltételeket (OCL: Object Constraint Language).

A felülvizsgálati folyamatot négy különböző területre bontva végezték: *Infrastructure, Superstructure, OCL* és *Diagram Interchange* [14]. Néhány további specifikáció arra szolgál, hogy az UML-t a rendszer dinamizmusának és a funkcionalitásnak a precízebb modellezése érdekében testreszabhatóvá tegye. Ez az alábbiakat jelenti: az aktivitás modellezésére új szemantikát dolgoztak ki, az UML-editor programokhoz lehetővé tették a szöveges megjegyzések beillesztését és az egyes modellnézetek korábbiaknál szabadabb manipulálását. Az UML 2.0 verziót vizsgálva nem szabad megfeledkezni a szabványos SPE metamodellről *(Software Process Engineering Metamodel),* amely egy, a módszertanok közötti átjárhatóságot biztosító keretrendszert specifikál a fejlesztéshez (lásd 15. ábra).



15. ábra Az UML új architektúrája [11]

Az MDA-ban az UML nemcsak a meglévő PIM- és PSM-modellnézetek definiálására szolgál, de alkalmas arra is, hogy az egyes modellnézeteket a folyamatosan változó körülményekhez/környezethez illessze. Az UML-nek a technológiai környezethez való illesztésével (lásd EDOC, EJB vagy CORBA) kapcsolatos előírásokat az *UML Profiles* tartalmazza, amely három különböző komponensből, és egy negyedik, a valósidejű rendszerek tervezéséhez segítséget nyújtó elemből áll:

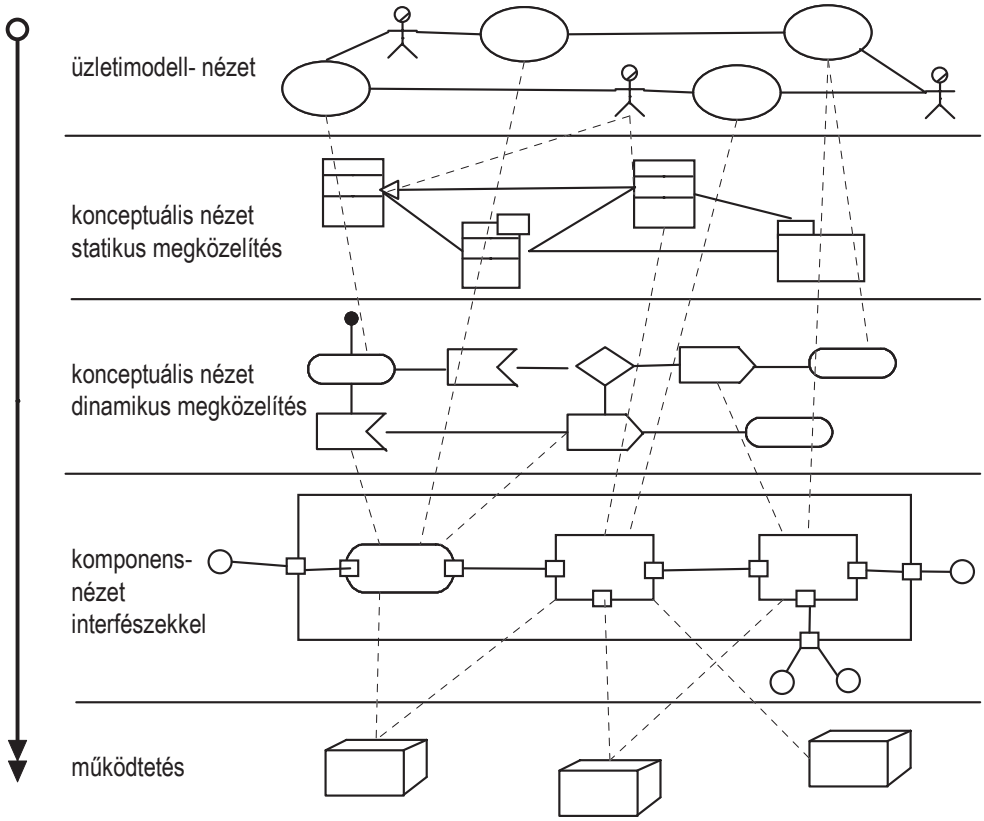
– A *Profile for EDOC* komponens a vállalati alkalmazások PIM-modellnézeteinek a fejlesztéséhez tartalmaz előírásokat, így definiálja az entitások, az események, a folyamatok, a szervezeti kapcsolatok modellezési szabályait, és mintákat is szolgáltat a modellezéshez. Mivel a PIM-modell az implementáció alapja, ezért külön fejezetekben határozza meg a PSM-leképezés módját is:

– A *Profile for EAI* komponens az üzenetalapú rendszerek fejlesztésének, míg

– a *Profile for CORBA* a PIM-modellek CORBA--specifikus PSM-modellé alakításának a szabályait írja elő.

– A *Profile for schedulability, performance and time* megoldást szolgáltat a valósidejű rendszerek modellezéséhez, pontos specifikációt adva az ütemezési és a futtatási problémák megoldásához, valamint a kritikus időtényezők kvantitatív elemzéséhez.

Az UML 2.0, amelyet a felhasználók és a fejlesztők széleskörű bevonásával, véleményük figyelembevételével dolgoztak ki, a korábbi szabványverziótól eltérően már nagymértékben kielégíti a vele szemben támasztott, újonnan felmerült és a jövőben várhatóan jelentkező elvárásokat. Az új UML-szabvány modellnézeteinek architektúráját a 16. ábra szemlélteti.



16. ábra Különböző modellnézetek a doménmodelltől a megvalósításig [11]

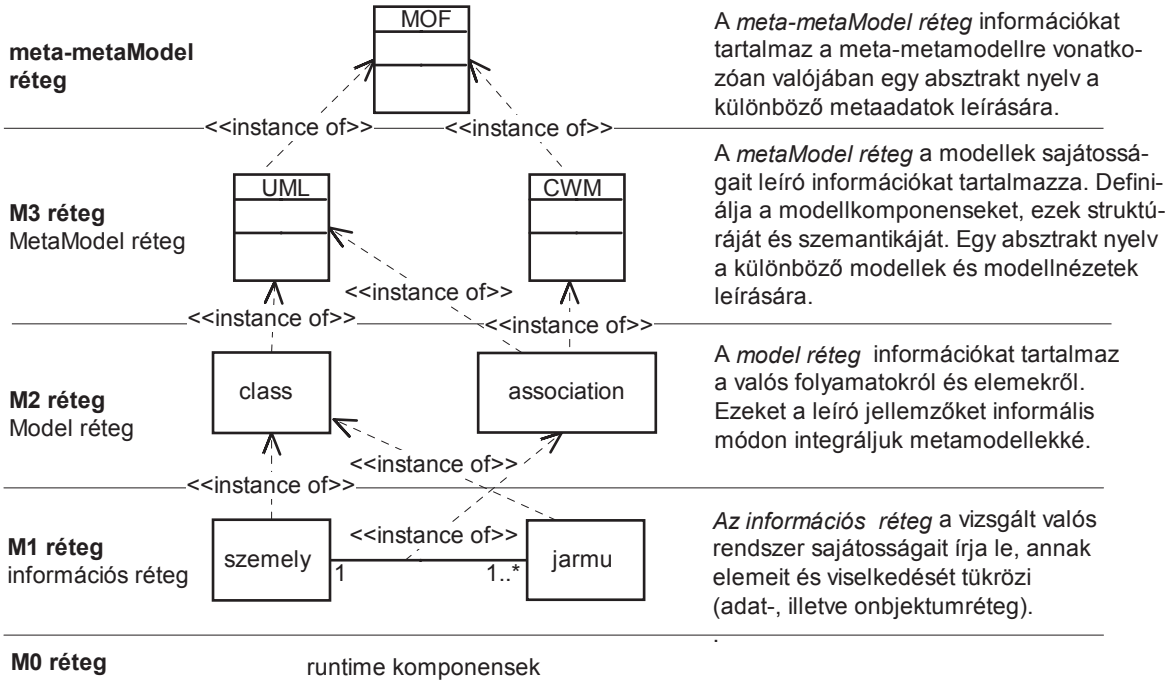
A MOF-szabvány

A Meta Object Facility szabvány valójában egy technológiai megoldás metaadatok definiálására olyan fejlesztésekben, amelyek a doménmodellt objektumosztályokká és különböző viselkedésmodellekké képezik le. A metaadat nem egyéb, mint egy általános specifikáció a rendszert leíró modellek sajátosságainak a kifejezésére, vagyis olyan információk összességéről van szó, amelyek a modelljellemzőket tükrözik. A MOF-nak, amely a modellekre vonatkozó információkat egy repositoryban (fejlesztési/modelladatbázis) tárolja, a korábbinál szélesebb jelentősége van, amennyiben a metaadatok

- fontos szerepet töltenek be a közös szemantikai keretrendszerben,
- teljesen illeszkednek a modell struktúráját és konzisztenciáját meghatározó absztrakciós szintaxishoz, és
- tartalmaznak a metamodellre jellemző információkat is.

A MOF-szabvány alapvető célja, hogy az együttműködésre kényszerülő metamodellek manipulálásához biztosítsa a CORBA-interfészeket, támogatva ezáltal a CORBA-alapú osztott környezet alkalmazásainak együttműködését. A közös metamodellek specifikálásával lehetővé válik a különböző platformon

futó alkalmazások kommunikációja [10]. Egy adott rendszerben azonban számos meta-adattípus lehetséges, amelyeket egy egységes metamodellnek kell kezelni tudni. A MOF egy olyan, metaadat-keretrendszerben (metadata framework) definiált, absztrakt szintaxist kínáló megoldás [10], amely négyrétegű architektúrát ad a különböző modellszintek információinak az integrálására (lásd 17. ábra).



17. ábra A MOF Metaadat architektúrája

CWM, az adattárház-metamodell

Az adattárház-koncepció a különböző formátumban tárolt adatok átalakításán és egységes kezelésén alapul. A cél egy olyan, egységesen kezelhető és karbantartható adattár létrehozása, amely hatékonyan elégíti ki a vállalati információigényeket, lehetővé teszi a különböző szintű döntési folyamatok támogatását, és egyfajta üzleti intelligenciaként nagymértékben hozzájárul az eredményes szervezeti működéshez. Az alapelv a metaadatok kezelésén és az adatok kezeléséhez, elemzéséhez kifejlesztett eszközökön nyugszik [3]. A Common Warehouse Metamodel egy ipari szabvány az adatrepositoryk integrációjához, ami azt jelenti, hogy a CWM egységesíti az adatbázismodell-sémákat, a sémaátalakítás módját, az adatok on-line elemzését (OLAP: On-Line Analitical Processing), valamint az adatkeresési megoldásokat (data mining). Valójában egy olyan keretrendszerről van szó, amely az adattárházban lévő különböző adatforrásokra, céladatokra, az átalakítás és az elemzés módjára, valamint az adattárház-adatok kezelésére és az adatműveletekre vonatkozó metaadatokat tartalmazza. A CWM--metamodel az alábbi mezőket tartalmazó *almetamodellekből* áll:

- A *Data Resources* relációs és objektumorientált rekordok, valamint multidimenzionális és XML-adatforrások metainformációit tartalmazza.
- A *Data Analysis* metamodellek az adattranszformációs, az OLAP, az adatbányászati, az infomrációmegjelenítési és az üzleti/szakterületi adatokat, míg

– a *Warehouse Management* metamodellek az adattárház-műveleteket és azok eredményeit reprezentálják.

A CWM fejlesztését az a cél motiválta, hogy szabványos megoldást kínálva kihasználják a vállalatoknál felhalmozódott adatvagyonban rejlő lehetőségeket. Ezt úgy oldották meg, hogy maximalizálták az objektummodell UML-készletek formájában történő felhasználását, és lehetővé tették a közös modellelemek lehető legszélesebb körben történő alkalmazását. A CWM struktúráját a 18. ábra szemlélteti.

	adattárház-folyamatok			adattárház-műveletek		
menedzsment	transzformáció		OLAP	adat-bányászat	információ-megjelenítés	szakterületi megvalósítás
lehetőségek	objektum-modell	relációs modell	rekordok	többdimenziós megvalósítás		XML
szakterület	üzleti információk	adat-típusok	kifeje-zések	kulcsok, indexek	típus-leképezés	szoftverek futtatása
	objektumok architektúra- és viselkedésmodell nézetei					

18. ábra A CWM-modell struktúrája [3]

A CWM szabványosítja mindazokat a metamodelleket, amelyek a vállalatok különböző adatbázisainak az együttműködését, az adatbázisok közötti átjárhatóságot lehetővé teszik. Az OMG és az MDC (Meta-Data Coalition) együttműködéseként elfogadott szabvány egy olyan egységes szabályrendszer, amely az UML alkalmazásmodellezési koncepcióját követve egységesíti az adatmodellezés és az adatimplementáció szabályait, specifikálva az adatbázissémák leképezésének formáit és módját. Fontos említést tenni az internetmegoldásokra vonatkozó kiterjesztésekről. Az egyik a *CWM Web Services,* amely előírja a Webimplementációk API-t használó metaadattovábbítási szintaktikáját és szemantikáját, a másik pedig a metaadatforgalmat szabványosító minták készlete, a *CWM Metadata Interchange Patterns* (MIP).

XMI/XML-szabványú metaadatforgalom

Az XMI alapvető célja, hogy heterogén, osztott környezetben lehetővé tegye a metaadatok a modellező eszközök és a metaadat-repositoryk közötti forgalmát. Az XMI-technológia integrálja a W3C (World Wide Web Consortium) XML--szabványát, valamint az OMG UML- és MOF--szabványait annak érdekében, hogy lehetővé tegye osztott környezetű alkalmazások együttműködését, és hogy a fejlesztők között hozzáférhetővé tegye az Interneten keresztül elérhető objektummodelleket és egyéb metaadatokat [5]. Az XML Metadata Interchange által lehetővé válik az UML-modellek XML-leképezése, és a vállalat minden pontján történő felhasználása. Az dokumentumformátum és definiálásának (Document Type Definitions) szabványosításával az UML-modellekhez és az UML-metamodellekhez XML-alapú formátum generálható. Ehhez a művelethez az alábbiak állnak rendelkezésre:

– MOF-alapú átalakítási szabályokat alkalmazó *XML DTD-k,* amelyek meghatározzák az XMI--dokumentumok specifikációjának a szabályait, és amelyek az XMI-dokumentumok előállításához és érvényesítéséhez lehetővé teszik az általános XML-eszközök használatát,

– *XML-dokumentumkészletek* a metaadatok XML-kompatibilis formára történő visszafejtéséhez, és fordítva, XMI-dokumentumokból metaadatok rekonstruálásához, valamint

– tervezési előírások XMI-alapú DTD-k és XML-streamek, valamint konkrét UML és MOF DTD-k előállításához.

Az XMI-specifikáció olyan együttműködés eredménye, amelyben az objektumrepository-kat, az objektummodellező eszközöket, a webtechnológiát, valamint az osztott környezetben működő üzleti alkalmazásokat fejlesztő cégek és az alkalmazók egyaránt részt vettek. Az XMI jelentősége abban áll, hogy általánosan és egyes rendszerekre specifikusan is hangsúlyozza az osztott környezet metaadatkezelésének és a metaadatok közötti átjárhatóságnak a fontosságát.

A középrétegszabványok szerepe

A középréteg-technológia a különböző alkalmazások, adaterőforrások és felhasználói folyamatok osztott környezetben történő valós idejű integrálására szolgál, függetlenül attól, hogy azok milyen operációs rendszer alatt, milyen protokollokkal, milyen hálózati platformon futnak. A középrétegszoftverek alapvető rendeltetése, hogy lehetővé tegye az alkalmazások együttműködését, vagyis azt, hogy azok megértsék egymás üzeneteit [19].

Az osztott működésű rendszerek architektúrájában a középrétegszoftverek önálló rétegként helyezkednek el a hálózati protokollok és az alkalmazási réteg között (lásd 19. ábra). Mint ahogyan az az ábrából is látható, az alkalmazásintegrációs köztesréteg egy egységes környezetet biztosít a fölötte elhelyezkedő alkalmazási réteg számára. Ebben az értelemben a különböző alkalmazások (fussanak azok eltérő platformon, hálózaton, más algoritmusokat és adatformátumokat használva) egyedi sajátosságai figyelmen kívül hagyhatók, és szabad út nyílik a kooperációra, az egymásnak küldött adatok/információk azonos módon történő értelmezésére. A középréteg azonban egy olyan infrastruktúraréteg, amely rugalmasságot ad az előre nem látható szituációk kezelésére is. A középrétegszoftverek piacán számos termék található. A szakirodalom általában négy kategóriát különböztet meg: (1) a *kommunikációs középrétegszoftverek* (MQSeries, RPC), (2) az *integrációs brókerek* (MQSeries Integrator), (3) az *objektumorientált környezetű integrációs brókerek* (CORBA, JMS, JCA), valamint (4) az *üzleti folyamatokat kezelő középrétegszoftverek* (BPM) csoportját.

19. ábra Az osztott környezetű rendszerek architektúramodellje

Mivel a CORBA a platform- és nyelvfüggetlensége, valamint a működési biztonsága miatt az egyik legalkalmasabb rendszer, és így kulcsszerepet tölt be az objektumszemléletű információfeldolgozási folyamatok működtetésében, ezért az OMG 1995-ben ezt a középrétegszoftvert fogadta el szabványként. A szabványosítás alapvető célja az alkalmazások vállalati szintű integrációjának, az objektumok hardverplatformtól, operációs rendszertől és programozási nyelvektől független együttműködésének a biztosítása volt. A COM/CORBA rendszer Javahoz és XML-specifikációkhoz történő illesztésére 1997-ben került sor, majd a CORBA lehetőségeit ezt követően kiterjesztették a valós idejű, a beágyazott és a biztonságkritikus rendszerek támogatására is. A CORBA-t, mint az egyetlen multiplatform, multilanguage megoldást, számos vállalat használja alkalmazásintegrációra.

MDA-szolgáltatások

Mivel az MDA alapvető célja az alkalmazások együttműködésének a biztosítása, ezért az OMG fontosnak tartja a vállalaton belüli és a vállalatok közötti Interneten keresztül történő együttműködést. Ehhez a *Pervasive Services*-nek nevezett szolgáltatások nyújtanak megoldást. A nagy könyvtári rendszereket szinkronizáló és karban-

tartó szolgáltatók különösen sokat profitálhatnak az MDA Pervasive Services által nyújtott lehetőségekből, mert azok hidat képeznek a különböző platformok között, biztosítják a futó alkalmazások biztonságos kooperációját, lehetővé teszik az osztott kezelést, a tranzakciókat, az igényelt állandó és egyéb szolgáltatásokat [20]. Az MDA-ban alapvetően négy alapszolgáltatás áll rendelkezésre: a *directory, a transaction*, a *security* és a *distributed event and notification services,* de van egy további szolgáltatás is, az alapmodellek által definiált, különböző platformokon implementálható interfészkészleteket kínáló *interface definition*.

A keretrendszer használata

Az MDA lehetőségeit és előnyeit az OMG Domain Task Forces még a szabványként történő elfogadás előtt megpróbálta kihasználni, és a fejlesztéseit MDA-elvek szerint hajtotta végre. Az eredmények kedvezőek voltak, az MDA-elvek tehát felelősséggel ajánlhatók.

Alkalmazásfejlesztés MDA szerint

Ha az MDA-t mint terméket tekintjük, akkor azt mondhatjuk, hogy egy olyan csomagról van szó, amely nemcsak követendő paradigmát fogalmaz meg a fejlesztők számára, nemcsak módszereket és eszközöket javasol az alkalmazásfejlesztés teljes életciklusára vonatkozóan, de egyértelműen definiálja, hogy *a modellezési folyamatban a PIM-modell kapja az elsődleges szerepet, és hogy a PSM-modell leképezése és az implementáció csak a PIM-modell alapján lehetséges.*

Az egyes modellnézetek, sőt a működtetés különböző rétegei közötti átjárhatóságot, az alkalmazások együttműködését a korábban már szabványként elfogadott MDA-elemek (UML, MOF, XMI, CWM, CORBA) garantálják. Bár az MDA-alapú fejlesztések eddigi tapasztalatai pozitívak, mégis általános az igény egy MDA-eszköz használatára. Jelenleg folyamatban van a teljes életciklust támogató MDA-alapú eszközök kifejlesztése, a piacon már találhatók EAI-koncepció szerinti fejlesztést részlegesen, vagy teljesen támogató megoldások. Ilyenek például:

- Az *ArcStyler* (Interactive Object Software) az EAI fejlesztőket a PIM-modellek kialakításában, és azoknak PSM-modellé történő átalakításában segíti [7].
- Az *ARI* (Adaptive Real-time Infrastructure; Kabira Technologies) a magas rendelkezésre állású, tranzakcióorientált rendszerek implementálásához és működtetéséhez nyújt támogatást.
- Az *iUML* és az *iCCG* (Kennedy Carter) PIM-modellek tervezését, létrehozását, tesztelését és már meglévő PIM-modellekbe történő integrálását teszi lehetővé, a végrehajtható UML *(iUML)* segítségével pedig biztosítja a PIM-modellek PSM-modellé történő leképezését és forráskóddá alakítását.
- A *ModelMethods* (Secant Technologies) a tranzakcióalapú és a tudásfelismerő rendszerek számára szolgáltat szabványos megoldást, amelyhez szabványos vizuális modellező eszközöket használ.

Az utóbbi években a fejlesztőeszközöket kínáló cégek, mint például a Codagen Technologies, az IBM, az InferData, az Iona vagy a Hewlett-Packard, a termékeiket már az MDA-hoz igazították. Bár az eddigi tapasztalatok azt mutatják, hogy ezek az eszközök az esetek többségében jól használhatók modelltranszformációra és forráskódok automatikus előállítására, mégis azt kell mondanunk, hogy még sokszor van szükség arra, hogy az így generált forráskódokat a programozók korrigálják, módosítsák. A fejlődés iránya azonban mindenképpen az, hogy a fejlesztőeszközök érettségi szintjének növekedésével a szakterületi modellek automatikusan transzformálhatók implementációs modellekké, hogy a már specifikált modellelemek széleskörűen és általánosan újrafelhasználhatók és az új technológiákhoz illeszthetők, ami a fejlesztési idő drasztikus csökkenéséhez vezet.

Az MDA *legnagyobb előnyét* azonban éppen abban látom, hogy rámutat a *platformfüggetlen modell meghatározó szerepére, elsődlegességére,* hogy a PIM-modellekből automatikus vagy félig automatikus eljárásokkal lehetővé teszi forráskódok generálását az alábbi elemek vonatkozásában:

- interfészek tervezése OMG IDL vagy más interfészdefiniáló nyelvek segítségével,
- specifikációk szerinti funkcionalitás biztosítása például a CORBA Component Modellel vagy az EJB-vel,
- hozzáférés az MDA-szolgáltatásokhoz (Pervasive Services, Domain Facilities),
- egy automatikusan generált híd segítségével keresztplatformú hozzáférés az MDA-ban már szabványosított funkcionalitáshoz,
- kézi programozási munkák segítése tranzakciós, illetve biztonságkezelési csomagokkal, újrafelhasználható modellelemekre vonatkozóan előre definiált műveletekkel, változódeklarációkkal.

Az MDA-alkalmazás előnyei

Egy ipari szabvány előnyei csak akkor használhatók ki, ha azt az érdekeltek többsége követi. Egy technológia-alapú szabvány esetében azonban az ún. kritikus tömeg, éppen az alkalmazott megoldások kompatibilitásának a hiánya miatt, nehezen érhető el, és problémát jelent az is, hogy bizonyos szabványok a működtetésben csak formálisan érvényesülnek. Az OMG éppen ezeknek a problémáknak a megoldására dolgozta ki az MDA-keretrendszert, amely egyrészt a szakterület funkcionalitását és viselkedését leíró modellspecifikációk technológiától való függetlenségét deklarálja, másrészt pedig biztosítja a különböző platformokon futó implementációk együttműködését. Az MDA előnyeinek vizsgálatakor fontos hangsúlyozni,

- hogy az MDA-elveket követő architektúrákban megvalósul a múltbeli, a jelenlegi és a jövőbeli alkalmazások kooperációja,
- hogy az MDA középrétegszabványát alkalmazva lehetővé válik az alkalmazások integrálása, és
- hogy az MDA szerint definiált, szakterületspecifikus alkalmazások vállalati és vállalatközi platformokon egyaránt a korábbiaknál szélesebb körben képesek az együttműködésre.

MDA-kiterjesztés

Az MDA-val kapcsolatban említést kell tenni egyéb szabványosítási törekvésekről is, mint például a Microsoft .NET vagy a Sun ONE rendszere. Ezek a rendszerek azonban nem használhatóak általánosan, hiszen specifikus platformokra készültek (Microsoft és Sun). Mivel azonban az OMG célja általánosan érvényes szabvány elfogadása, ezért arra törekszik, hogy az MDA-ban jelenleg nem szereplő, de széleskörűen használt középrétegszoftvereket speciális elemként az MDA-ba integrálja. Bár a keretrendszer fókuszát a középrétegszabvány vonatkozásában továbbra is a CORBA képezi, az OMG fontosnak tartja, hogy a korrektül működő környezetek mindegyike (lásd Webszolgáltatások, COM, JAVA including EJB, C#, .NET, ONE vagy XML/SOAP) az MDA része legyen.

Következtetések, előretekintés

Az MDA valójában válasz azokra a kihívásokra, amelyet a különböző platformokon működő, eltérő architektúrájú rendszerekből álló hálózati környezetek, a folyamatosan változó körülmények jelentenek a számítógépes támogatásokat használók számára. A heterogén rendszerek együttműködési igénye számos elvárást támaszt az alkalmazott technológiával szemben, így biztosítani kell

- *a hordozhatóságot,* amely növeli az alkalmazások komplexitását, a többszörös felhasználás lehetőségét, és csökkenti a költségeket,

– a *platformfüggetlenséget, a különböző platformok közötti együttműködést,* és nem utolsósorban
– *a hatékonyságot.*

Az elvárások teljesítése azonban csak akkor lehetséges, ha a fejlesztők számára egyszerűen kezelhető, gyors megoldások állnak rendelkezésre, ha az alkalmazott megoldásoktól függetlenül lehetővé válik korábbi modellelemek felhasználása, valamint a fejlesztési repositoryk információinak projekten belüli és projektek közötti megosztása. Jelenleg már számos vállalat hajtott végre MDA-elvek szerinti fejlesztést, és széles kínálat van támogatóeszközökből is. Az MDA igazi sikerét azonban akkor könyvelhetjük el, ha a fejlesztők többsége a munkát a technológiafüggetlen PIM-modell megtervezésével kezdi ahelyett, hogy a forráskódok írásának állna neki. Fejlesztési tapasztalataimból kiindulva az MDA-nak ez a legfontosabb üzenete!

A cikkben alkalmazott rövidítések

API: Application Programming Interface
B2B: Business to Business
B2C: Business to Customer
BIS: Business Information System
CEO: Chief Executive Officer
CIO: Chief Information Officer
COM: Component Object Model
CORBA: Common Object Request Broker Architecture
CWM: Common Warehouse Metamodel
DTD: Document Type Definition
EAI: Enterprise Application Integration
EDI: Electronic Data Interchange
EDOC: Enterprise Distributed Object Computing
EJB: Enterprise Java Beans
IDL: Interface Definition Language
IS: Information System
IT: Information Technology
MDA: Model Driven Architecture
MDC: MetaData Coalition
MIP: Metadata Interchange Patterns
MOF: Meta Object Facility
OCL: Object Constraint Language
OLAP: On-line Analitical Processing
OMG: Object Management Group
PIM: Platform Independent Model
PSM: Platform Specific Model
RFP: Request for Proposal
ROI: Return of Investment
SPE: Software Process Engineering
UML: Unified Modeling Language
W3C: World Wide Web Consortium
XMI: XML Metadata Interchange
XML: eXtended Markup Language

Hivatkozások

[1] Bézivin, J.: *From Object Composition to Model Transformation with MDA* – Conference IEEE-Tools-39, Santa Barbara, USA, 2001
[2] Booch, G. – Rumbaugh, J. – Jacobson, I.: *The Unified Modeling Language* – Addison-Wesley Longman Inc., 1999.
[3] Chang, D.T.: *Common Warehouse Metamodel Specification* – http://www.omg.org/-cgi-bin/doc?ad/2001-02-01
[4] DSouza, D.: *An Architecture for Modeling* – Enabling Model Driven Integration – Kinetium, 2002
[5] Hazra. T.K.: *MDA brings standards-based developing Modeling to EAI Teams* – ADTmag, Application Development Trends, May, 2002.
[6] Heaton, L.: *OMG-XML Metadata Interchange (XMI) Specification, v1.2* – www.-omg.org/cgi-bin/doc?formal/2002-01-01
[7] Hubert, Richard: *Model Driven Architecture with ArcStyler* – The Business of IT Architecture, Interactive Objects 2002.
[8] Knapman, J.: *Business-Oriented Constraint Language* – 3rd International Conference on the Unified Modeling Language, University of York, UK, October, 2000.
[9] Korbyn, C.: *A Standardization Odyssey* – Communications of the ACM, 1999. Vol. 42. No. 10.
[10] *OMG Meta Object Facility V 1.4* – www.omg.org/; OMG April, 2002
[11] Raffai, M.: *Increasing IS Requirements and New Engineering Technologies* – 10th IDIMT Conference, Zadov, 2002.

[12] Raffai, M.: *The new Standard of UML 2.0 by the OMG TF Draft in September 2002.* – CIB-IqSoft Symposium, October, 2002
[13] Raffai, M.: *Egységesített megoldások a fejlesztésben – UML modellező nyelv és RUP-módszertan* – Novadat Kiadó, ISBN 963 9056 29 4
[14] Raffai, M.: *Objektumok az üzleti modellezésben – Az objektumorientált fejlesztés elvei és módszerei* – Publisher Novadat, ISBN 963 9056 28 6
[15] Raffai, M.: *Trends in Information Technology - Intelligent Solutions in BIS Development and Management* – 11th Interdisciplinary Information Management Talks Conference, Ceské Budejovice, 2003.
[16] Raim, M.: *Implementation Infrastructure: Enablers for Rapid Enterprise Integration* – OMG Information Day, 2002.
[17] Siegel, J.: *Developing in OMG's Model-Driven Architecture* – OMG, Developing in MDA, 2002
[18] Soley, R.: *Model Driven Architecture* – OMG Draft Paper, V 3.2, November, 2002.
[19] Sugár, P.: *Alkamazásintegráció és EAI* – infoByte, March, 2002
[20] *The Architecture of Choice for a Changing World* – www.omg.org, OMG 2002