

Debreceni Egyetem

Informatikai Kar

Webes alkalmazásfejlesztés WebML segítségével

Témavezető:

Kollár Lajos

Egyetemi tanársegéd

Készítette:

Morvai Mihály

Programozó matematikus

Debrecen

2009

TARTALOM

BEVEZETÉS	5. oldal
1. Technológiai áttekintés	6. oldal
1.1. CASE eszközök.....	6. oldal
1.2. MDD	6. oldal
1.3. WebML	7. oldal
1.4. Fejlesztési folyamatok.....	.9. oldal
1.4.1. Inkrementális modell	.9. oldal
1.4.2. Boehm spirál.....	. 10. oldal
1.4.3. A WebML fejlesztési folyamata.....	. 11. oldal
1.4.4. Követelmény elemzés (requirement analysis).....	. 12. oldal
1.4.5. Fogalmi modell (conceptual modell).....	. 12. oldal
1.5. WebRatio.....	. 12. oldal
1.6. A használt technológiák rövid áttekintése	14. oldal
2. WebML a gyakorlatban.....	16. oldal
2.1. A WebRendelő projekt bemutatása.....	. 16. oldal
2.1.1. Általános leírás	. 16. oldal
2.1.2. Általános követelmények	. 16. oldal

2.1.3.	Szakterületi fogalmak.....	17. oldal
2.2.	WebRatio, az első lépések.....	18. oldal
2.3.	Adatmodellezés.....	20. oldal
2.3.1.	Az adatmodellezés elmélete	20. oldal
2.3.1.1.	Adatmodellezés WebRatióban	20. oldal
2.3.1.2.	Egyedek.....	21. oldal
2.3.1.3.	Kapcsolatok.....	21. oldal
2.3.1.4.	Egyedek és kapcsolatok leképezése az adatbázisba	22. oldal
2.4.	A hiperszöveg modell és részei (Hypertext modell)	23. oldal
2.4.1.	Oldalak	24. oldal
2.4.2.	Hivatkozások megjelenési formái	25. oldal
2.4.3.	Egyed – előfordulás szűrése	26. oldal
2.4.4.	Érvényesítési szabály (validation rule).....	27. oldal
2.4.5.	Egységek (Units)	28. oldal
2.4.6.	Műveletek (Operations).....	31. oldal
2.4.7.	Hasznos kiegészítő egységek.....	33. oldal
2.4.8.	Globális paraméterek és változók.....	34. oldal
2.5.	Modellezési példák	35. oldal
2.5.1.	Hitelesítés, a rendszer nézetei.....	35. oldal
2.5.2.	Területek, oldalak, navigáció, linkek.....	37. oldal
2.5.3.	Űrlapok készítése, mezők validálása	39. oldal

2.5.4. Lekérdezések, listák	41. oldal
2.5.5. Saját üzleti logika hozzáadása	42. oldal
2.5.6. Megjelenítés, formázás, rács nézet	42. oldal
3. ÖSSZEGZÉS	44. oldal
3.1. Az MDD/WebML hasznossága	44. oldal
3.2. Módszertana, tanulhatósága	44. oldal
3.3. A fejlesztés hatékonysága	44. oldal
3.4. A végeredményként előállt szoftver értékelése	45. oldal
IRODALOMJEGYZÉK	46. oldal
FÜGGELÉK	47. oldal

Bevezetés

„A szoftverfolyamat tevékenységek és kapcsolódó eredmények olyan halmaza, amely szoftverterméket állít elő” [1]. Ebből a folyamatból a szoftverfejlesztést, mint tevékenységet emelném ki, ugyanis a dolgozat témája nagyrészt e tevékenységet körül fog mozogni. A bemutatandó módszertan közel tíz éves múltra tekint vissza, de korai verziók, korlátaik miatt nem voltak alkalmasak az ipar és kereskedelem folyamatosan megújuló igényeinek kielégítésére. A dolgozatban szereplő verzió azonban több új bővítési lehetőséget tartalmaz. A WebML módszertan alapvetően modellezési feladatokra lett kitalálva, amely feladat végeztével a modellből forráskód generálható.

A dolgozat során igyekszem bemutatni a WebML modellező nyelvet, valamint a hozzá készített környezetet a WebRatio-t. A bemutatás három fő részre osztható, először magát a nyelvet és a kapcsolódó technológiákat mutatom be fogalmi szinten. Ezután elővezetek egy példaprojektet, definiálom a nyelv szükséges alkotóelemeit, működési mechanizmusát. A legutolsó részben pedig néhány megvalósított funkción keresztül mutatom be az összetevőket működés közben.

A dolgozat terjedelme és a rendelkezésre álló idő nem engedi, hogy teljes egészében végigvegyem a WebML és WebRatio témákat, nem is beszélve egy valós projekt teljes, minden részletekre kiterjedő megvalósításáról. Előre leszögezem, hogy ezért maradnak ki jelentős részek, mégis igyekeztem a legfontosabb és nem utolsósorban legalapvetőbb témaköröket kiválasztani. A projekt is sok helyen egyszerűsítve van, nyilván a valós életben egy üzleti környezetben nem biztos, hogy megállná a helyét, de prezentációs célokra még így is tökéletes.

1. Technológiai áttekintés

Ebben a fejezetben az [1],[3],[4],[6],[7],[8],[9],[10] támaszkodva megpróbálom elhelyezni a világban és a módszertanok között a WebML-t és a WebRatio-t, és definiálom az előzményként szolgáló módszertanokat.

1.1 CASE eszközök

A betűszó jelentése Computer-Aided Software Engineering, magyarul számítógéppel támogatott szoftvertervezés, ami olyan programokat takar, amely szoftverfejlesztés folyamatának egy-egy tevékenységét támogatja vagy információt szolgáltat a kifejlesztés alatt álló termékről. A CASE technológia napjainkban főleg rutintevékenységek kiváltására szolgál, és bár jelentős javulást sikerült elérni a használatával, ez azonban jelentősen elmarad a korábban hozzá fűzött reményektől.

CASE eszközöknek nevezhetjük a diagramszerkesztőket, kódszerkesztőket, kódgenerátorokat, nyomkövető és belövő eszközöket, verziókövetőket, hogy csak néhány gyakori előfordulást említsek, de a szoftverfejlesztés minden fázisához megtalálható a CASE eszközök megfelelő halmaza. A CASE eszközöket, eszközkészleteket többféle szempontból is lehet csoportosítani, a besorolás alapja lehet akár az adott folyamat támogatottságának mértéke vagy maga az, hogy melyik folyamatot milyen módszerrel támogatja. Az egyes technológiákhoz kapcsolódnak a módszer jelölésrendszeréhez tartozó szerkesztőprogramok, a rendszermodellt a módszer szabályai szerint ellenőrző és elemző modulok, és a rendszer dokumentáció elkészítésében segítő jelentésgenerátorok is. A CASE eszköz kódgenerátora a rendszermodellből és a folyamatirányításból automatikusan generálhat a forráskódot.

1.2 MDD

Az MDD (Model Driven Development), magyarul modell alapú fejlesztés, egy szoftverfejlesztési irányzatot képvisel, ahol egy kellően kifejező modell alapján forráskód generálható, motivációja a produktivitás növelése. A modell készítés folyamatát természetesen meg kell tanulni, de kellő tapasztalat birtokában és a megfelelő eszközszoftver

használatával a kódolás nagy része egy platform független modell elkészítéséből és annak a megfelelő platformra történő transzformációjából áll. A modellezés sok esetben könnyebben járható út, mint egy adott programnyelven történő kódolás, ugyanis a fejlesztő vagy tervező távolabbról, magasabb absztrakciós szintről tekinthet a rendszerre.

1.3 WebML

Kezdetben a webes alkalmazások fejlesztésére csupán mint egy összeépítési feladatra tekintettek, kevés hangsúlyt fektetve magára a szoftverfejlesztési folyamatra. Azonban az egyre növekvő méretű és bonyolultságú alkalmazások számos megoldandó problémát vetettek fel, nehézkessé vált a menedzselés és karbantartás, szükségessé vált az együttműködés más alkalmazásokkal valamint nem utolsósorban követni kellett a felhasználói igények folyamatos változását.

Több olyan megoldás született, amely az adat-intenzív webes alkalmazásokat támogatták leginkább, ilyenek az Microsoft által fejlesztett ASP (Active Server Pages) vagy a Java alapú JSP (Java Server Pages), melyek elsődleges funkciója a weblapok dinamikus generálása előre meghatározott séma alapján. Mindamellet, hogy nagyon eredményes implementációs eszközök, nem sok lehetőséget tartogattak a követelmény tervezés és a fejlesztés további fázisai közötti hézag áthidalására. A tapasztalatok alapján olyan módszertanokra, nyelvekre és eszközökre volt szükség, amelyek kiegészítve az aktuális webes technológiát lefedik a fejlesztési folyamat minden aspektusát.

Ezen igények hívták életre a W3I3 Projektet, középpontjában az információs rendszerek igényeit kielégítő Intelligens Információs Infrastruktúra kifejlesztésének gondolatával. A projekt két jelentős piaci résztvevő igényei formálták, végeredményként pedig előállt az újfajta modellező nyelv a WebML és az azt támogató CASE környezet a Toriisoft. Azóta a Toriisoft-ról csak néhány helyen tesznek említést, e dolgozat írásakor az egyetlen támogatott WebML fejlesztő környezet a WebRatio.

A **WebML** név a **Web Modeling Language** kifejezésből származik, jelentése Webes modellező nyelv, egy harmadik generációs webes tervezési módszertan, aminek eredeti célja az adat-intenzív web alkalmazások tervezésének és fejlesztésének támogatása. A korábbi

nyelvekhez képest egy újszerű szemléletet, a modell alapú fejlesztést helyezte előtérbe a webes alkalmazások esetén. A koncepció szerint ehhez a szemlélethez egy átlátható, intuitív grafikus megjelenés társul, ami lehetővé teszi az a különböző területek fejlesztő csoportjai egyszerű kommunikációját. A tervezőnek elegendő meghatározni az oldalak alapvető szerkezetét, bármiféle architektúrális szintű részletezés nélkül. A grafikus megjelenés mögött egy XML dokumentum írja le a modellt, ebből az XML-ből állítja elő a generátor program az oldal implementációját.

A WebML jelenleg a negyedik verziójánál tart. A legelső verzióban az interfészek moduláris felépítése, navigációja volt hangsúlyos. Olyan adat-intenzív weboldalak fejlesztéséhez szánták, ahol csak a tartalmak megjelenítése volt a cél, azok módosítása nem. A második verzióban lehetővé vált az üzleti logika korlátozott módon történő beágyazása, a felhasználók azonosítása, és ezekkel együtt a tartalom felhasználók általi menedzselése is. A harmadik verzióban bevezették a modell alapú bővítmények koncepcióját, amivel még szélesebb körben fedték le az új alkalmazások elvárásait. Ez a módosítás kiemelte a komponens-alapú modellezés szerepét és alapjául szolgált a későbbi kiterjesztéseknek. A jelenlegi, negyedik verzióban, a bővítmények új szemléletű megközelítése eredményeképpen, olyan területekről való komponensek adhatók, amiket korábban nem hoztak összefüggésbe a modell-alapú fejlesztéssel [4].

A WebML négy, egymásra merőleges nézőpontból szemléli egyszerre a rendszert és a hatékony munka érdekében az alábbi perspektívák mindegyike külön, az adott területen jártas szakembert igényel:

- Strukturális modell(Structural model) : Az oldal adat típusú összetevőit ábrázolja az egyedek és a közöttük lévő kapcsolatok segítségével. Az egyedek típusos attribútumokból állnak, és a kiterjesztést (generalization) használva hierarchiába szervezhetők.
- Hiperszöveg modell (Hypertext model): Egy vagy több hiperszöveget ír le, a különböző hiperszövegek a rendszer különböző nézeteit, szerepköreit jelenti. Az oldal nézet két részmodellre osztható:

- Kompozíciós modell (Composition model): azok a lapok és egységek összessége, amik egy hiperszöveghez sorolhatók. Az egységek hat típusát különböztetjük meg:
 - adat (data)
 - többes-adat (multi data)
 - index (index)
 - szűrő (filter)
 - görget (scroller)
 - direkt (direct)
- Navigációs modell (Navigation model): A lapok összeköttetését ábrázolja. Általában környezeti hivatkozásokról (contextual links) beszélünk, de némely esetben nem-környezeti hivatkozások (non-contextual links) is előfordulnak. Ezt a kérdést az összeköttetés módja illetve a forrás és cél határozza meg.
- Prezentációs modell (Presentation modell): a grafikus megjelenés és elrendezést leíró rész, XML szintaxist használ, a végső megjelenés független megjelenítő eszköztől és nyelvtől.
- „Személyre szabás” modell (Personalization model): A felhasználókat előre definiált csoportokba kell sorolni. A felhasználó által igénybe vehető szolgáltatások, az előtte megjelenő felület kiosztása és megjelenése attól függ, hogy épp melyik csoporthoz tartozik.

1.4 Fejlesztési folyamatok

1.4.1 Inkrementális modell

Az inkrementális modell a korábbi fejlesztési modellek hátrányait próbálja kiküszöbölni, ez a követelmények változásának kezelése és a szoftverstruktúra romlása. A követelmények változása akkor okoz problémát, ha szoftver fejlesztése a vízesés folyamatmodellel történik. Ekkor a követelmények meghatározása, a tervezés és az implementáció fázisai mind egymástól izolálva történnek, csak akkor lehet áttérni a következő fázisra, ha az előző teljesen

befejeződött. Gyakori eset azonban, hogy maga a megrendelő nem tudja pontosan mire van szüksége, és csak a kész terméket látva szembesülne a félreértésekkel. Természetesen az ilyen esetek elkerülésére van az evolúciós folyamatmodell, így a megrendelő még a fejlesztés korai szakaszában kap egy működőképes programot (prototípust), ami alapján finomítani lehet a követelményeket, és egy újabb prototípus áll elő. Azonban kellően határozatlan megrendelő esetén és a lépések sokszori ismételésével a program struktúrája leromlik.

Ezeket elkerülve az inkrementális modell azt mondja, hogy a rendszert kicsi, könnyen kezelhető részekre, úgynevezett inkremensekre kell bontani, amelyeknek jól definiálnak kell lenniük. A kifejlesztett inkremensek tesztelést és validációt követően integrálni kell a rendszerbe. Ezzel a fejlesztési folyamat egészében egy jól tesztelt rendszer áll rendelkezésünkre.

1.4.2 Boehm Spirál

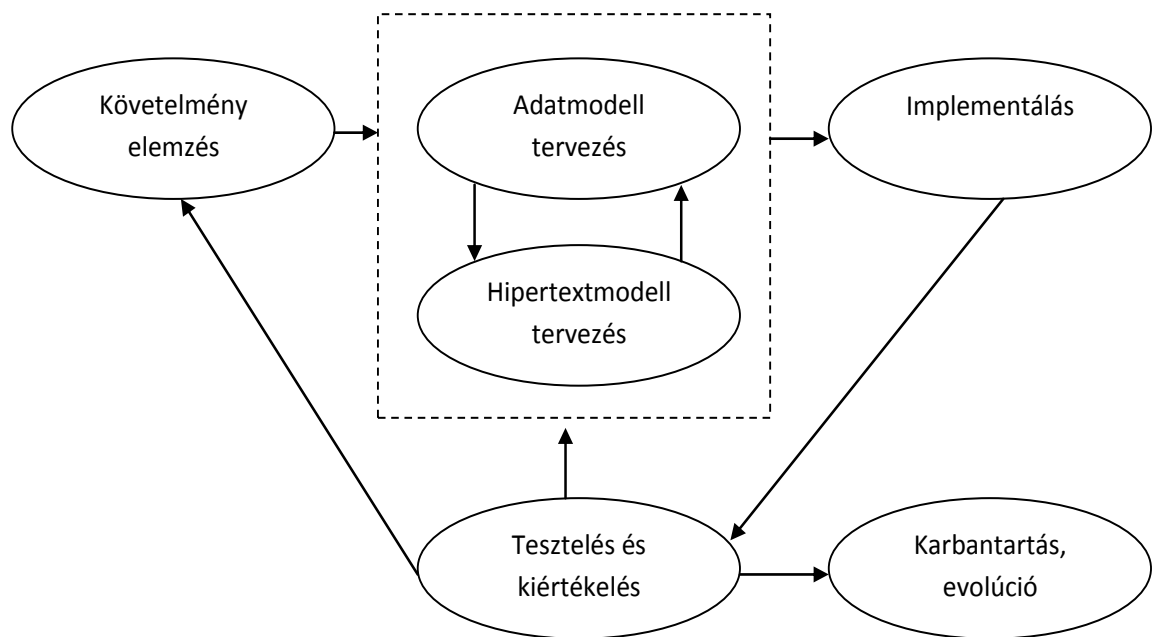
A Boehm spirál, avagy a spirális fejlesztési modell jelentősége, hogy nagy hangsúlyt fektet a kockázatok elemzésére és kezelésére. Ahogyan azt a neve is mutatja, úgy kell elképzelni a szoftverfejlesztési folyamatot, mint egy spirált, minden ciklus a fejlesztési folyamat egy fázisát írja le. A spirál legbelső ciklusa a megvalósíthatósági tervvel foglalkozik, a következő a rendszerkövetelményekkel és így tovább. Véges számú spirálciklus végrehajtása után vagy a befejezésről kell döntenie vagy eljutunk a működő rendszerhez. A spirál minden ciklusa tovább osztató négy szektorra:

- Cél kijelölése: meg kell határozni a követelményeket és célokat a spirálciklusra, fel kell ismerni a lehetséges kockázatokat.
- Kockázat elemzése, csökkentése: A felismert kockázatok részletes elemzése, ha lehet lépéseket kell tenni a kockázati tényezők minimalizálása érdekében.
- Fejlesztés és validálás: Az ismert kockázati tényezők tükrében választani kell egy fejlesztési modellt, erről szabadon dönthetünk, az azonban fontos tény, hogy egyes kockázati tényezők dominanciája a megfelelő fejlesztési modellel minimalizálható.

- Tervezés: Ki kell értékelni az addig elért eredményeket és dönteni a folytatásról. Ha úgy döntünk, hogy folytatódik a spirál a következő ciklussal, akkor meg kell tervezni azt, fel kell készíteni a projektet a következő fázisra.

1.4.3 A WebML fejlesztési folyamata

A WebML fejlesztési módszertan a spirális és az inkrementális modellen alapul összhangban a modern webes és szoftveres alkalmazásfejlesztési módszerekkel. A fejlesztés a fázisok iteratív ismétléséből áll, a későbbi fázisokban egyre pontosabbak a követelmények, a rendszer modellje egyre nagyobb mértékben és részletesebben kidolgozott. Éppen ezért a termék élettartama alatt számos fejlesztési cikluson megy keresztül, amelyek eredménye egy prototípus vagy egy hiányos funkcionalitású verzió. Minden iteráció teszteléssel és kiértékeléssel kezdődik, majd vagy a szoftver bővítése vagy a követelmények újrafogalmazása következik. Az **1. ábra** a fejlesztési folyamat egyes fázisait mutatja be.



1. ábra Fejlesztési fázisok WebML-ben

1.4.4 Követelmény elemzés (requirement analysis)

Mint a rendszerfejlesztésben általában, a követelmények elemzése ebben az esetben is egy speciális információhalmaz előállítását jelenti az üzleti követelmények alapján. A halmaznak tartalmaznia kell a szakterületről (application domain) információkat, a kifejlesztendő funkciókat és nem árt mindezt egy könnyen érthető formában leírni. Az elemzés végeztével ismertté válik a felhasználók lehetséges csoportosítása és a hozzájuk tartozó oldal nézetek (site view) is. Az üzleti folyamatokban azonos szerepet betöltő felhasználók egy csoportba kerülnek. Az egyes felhasználói csoportok más-más perspektívából szemlélik a rendszert, ezt az alkalmazás oldal nézetekre való felbontásával lehet elérni. Az oldal nézet egy adott felhasználó csoport által elérhető interfész halmaz. Az interfész halmazoknak és a felhasználó csoportoknak egyaránt létezhet metszete, azaz egy felhasználó egyszerre több csoportba is tartozhat, illetve lehetnek olyan interfészei, nézetei a rendszernek, amelyek több felhasználócsoport által látható vagy használható. Szintén a követelménytervezés fázis eredményének tekinthető a funkcionális követelmények specifikációjának pontosítása, valamint a rendszermag részét és az adatmanipuláció alapját képező információs objektumok meghatározása.

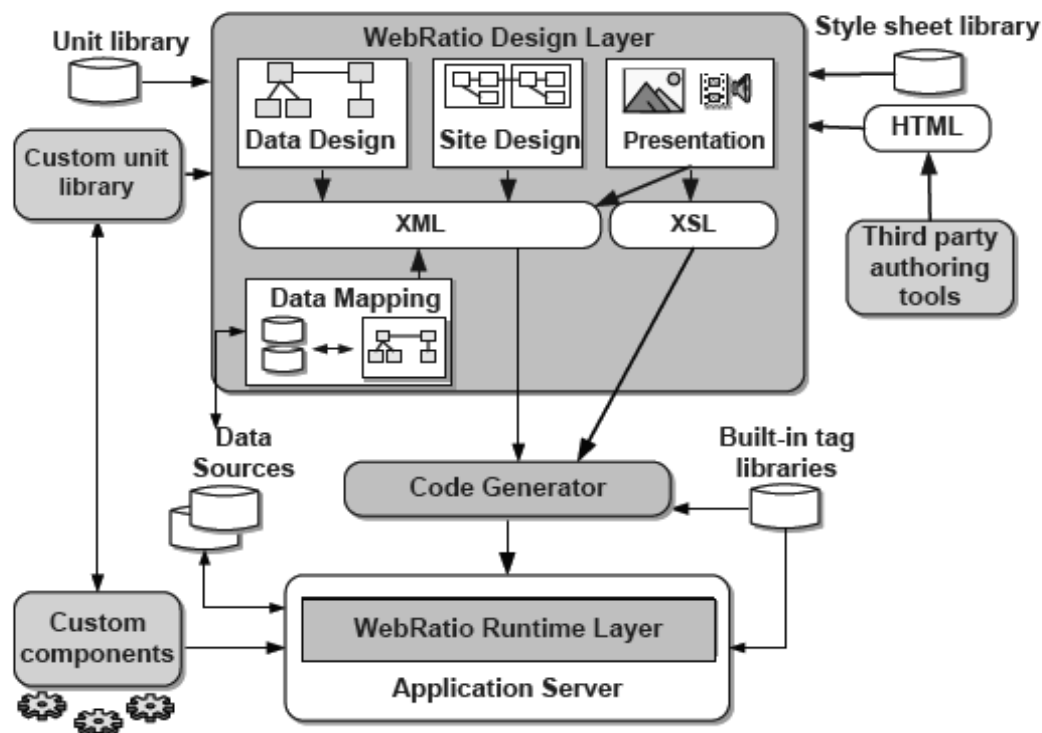
1.4.5 Fogalmi modell (conceptual modell)

A fogalmi modell elkészítése magas absztrakciós szinten történik, vázlat jellegű, konkrét implementációs kérdéseket nem tisztáz. A WebML szemléletben ez a fajta modellezés két fő részre bontható, mégpedig az adatszerkezet tervezésre és a hiperszöveg tervezésre. Ezekről bővebben a gyakorlati megvalósításról szóló fejezetben írok.

1.5 WebRatio

Az eddig tárgyalt dolgokon túl, a WebML abban is különleges, hogy az elméleti kutatásokkal párhuzamosan már folyt az ipari alkalmazása is. A Milánói Műszaki Egyetemen folyó WebML-el kapcsolatos kutatások egyik melléktermékeként létrejött 2001-ben a Web Models nevű vállalat, amelynek célja az MDD és a WebML eredményeinek versenyszférába történő

bevezetése. Legfőbb termékük a 2006-ban megjelenő WebRatio, ami egy WebML-t támogató üzleti felhasználásra szánt eszközrendszer. Két fő rétegből áll: tervezési réteg (design layer) és futtatási réteg (run-time layer).



2. ábra A WebRatio logikai felépítése

A tervezési réteg a legjobban szembeötlő, ez ugyanis az a grafikus interfész, amely a modellező nyelv vizualizációját jelenti, itt készíthetjük el az alkalmazás adat és hiperszöveg modelljének diagramját. A tervezési réteghez sorolható továbbá az adat feltérképező modul (data mapping modul) és az EasyStyler megjelenés tervező (EasyStyler Presentation designer).

Az adat feltérképező modul oda-vissza módú tervezést végez az adatmodell és a hozzá kapcsolt adatbázis séma táblái között. A meglévő adatmodellhez képes legenerálni az adatmodell realizációját előállító adatbázis (SQL) szkriptet. Ellentétes irányban, pedig a hozzárendelt sémában található táblákat és a közöttük lévő kapcsolatokat képezi le az adatmodellbe egyedekre és kapcsolatokra. A modult a kapcsolt adatbázis szinkronizálásával (Synchronize) indítjuk el, és a megjelenő párbeszédpanelen az import/export funkció választása határozza meg a konverzió irányát.

Az EasyStyler olyan szolgáltatást nyújt, ami lehetővé teszi a megjelenítés stílusának definiálását. Létrehozhatók XSL stíluslapok XHTML makettekkel, ezután az XSL stílusokat WebML lapokhoz kell rendelni, illetve meghatározni az egységek elrendezését és sorrendjét minden laphoz.

A futtatási réteg már nem ilyen látványos, előre implementált WebRatio összetevőkből és egy web alkalmazás keretrendszerből áll, amik a modellből generált implementációkkal együtt képezi a webes alkalmazást, ez a hármas kerül feltöltésre az alkalmazás szerverre, ez a hármas alkotja a modellből készült webes alkalmazást.

1.6 A használt technológiák rövid áttekintése

Tomcat, JSP

A Java Servlet és a JSP (Java Server Pages) technológiák implementációja, nagyteljesítményű, ipari felhasználásra szánják. A Java szervletek (Java Servlet) olyan Java nyelven íródott szerveroldali alkalmazások, amelyek képesek weboldalak dinamikus generálására. A JSP a dinamikus weboldalak generálásának egy egyszerűsített megközelítése, egy alapvetően HTML jellegű oldalba rövid java kódrészletek kerülnek, így az oldal egy sablonként működik, ami alapján generálódik a ténylegesen megjelenítendő oldal.

Struts

Hasonlóan a Tomcat-hez a Struts is az Apache cég nevéhez kapcsolódó technológia, egy nyílt forráskódú, webes alkalmazás-keretrendszert takar. Java Servlet technológiával kiegészítve MVC (modell-view-controll) architektúraként használható.

Oracle RDBMS szerver

A jól ismert Oracle adatbázis-kezelő 10g XE verzióját választottam az alkalmazáshoz, mellette szólt az egyszerű telepíthetőség, menedzselhetőség, ingyenesség, és hogy hasonló tudással rendelkezik mint a nagy testvér, az Oracle Database 10g Release 2. Egyetlen apró

illesztés volt szükséges, a WebRatio driver mappájába be kellett másolni az ojdbc14-oracle10g.jar JDBC meghajtót.

Hibernate (SQL és HQL)

A Hibernate egy hatékony objektum-relációs tárolási és lekérdező szolgáltatás. Saját nyelvét használja, egy SQL kiterjesztést, amit HQL-nek neveznek.

2. WebML a gyakorlatban

A fejezet első felében a WebML gyakorlati alapjait szeretném bemutatni, a második felében pedig bemutatok néhány példát ezzel kapcsolatban

2.1 A WebRendelő projekt bemutatása

2.1.1 Általános leírás

Az alap probléma látszólag egyszerű, adott egy cég, amely szolgáltatásokat nyújt ügyfelei részére, legyenek ezek például takarítás, kertgondozás, költöztetés, rakodás és egyéb fizikai munkák. Az ügyfelek megrendelés útján veszik igénybe a szolgáltatásokat, ami történhet személyesen, telefonon, faxon illetve webes felületen. A dolgozat szempontjából az első három lehetőség nem fontos, azonban az utolsónál szükség van egy olyan webes alkalmazásra, ami:

- nyilvántartja a megrendeléseket és a velük kapcsolatos további információkat,
- lehetőséget nyújt a regisztrált ügyfeleknek rendelések leadására,
- kommunikációs biztosít a megrendelő és a megrendelést feldolgozó alkalmazott között.

Körvonalakban ez a WebRendelő projekt.

2.1.2 Általános követelmények

- A rendszer használata azonnal megtanulható legyen, a felhasználói felület jól átgondolt és könnyen használható legyen.
- A rendszer hiba esetén a felhasználónak értelmes értesítést adjon.
- A használatot, módosításokat naplózza a rendszer, hogy összegzéseket lehessen megtekinteni.

- Legyen megkülönböztetett rendszer adminisztrátor jog és felület, adminisztrátor jog és felület és legyen egy általános felület a regisztrált ügyfelek számára, ezek mindegyike felhasználónévvel és jelszóval legyen hozzáférhető.
- Ügyfelek csak regisztráció útján kerülhetnek a rendszerbe, amit az adminisztrátor végezhet el.
- A leadott megrendelések csak megerősítés után érvényesek, ekkor az ügyfél kapjon egy értesítést a rendelése megerősítéséről. A megerősített rendelés automatikusan rögzítődik.
- Nem teljesíthető megrendelés esetén az ügyféllel fel kell venni a kapcsolatot egyeztetés céljából. A személyes egyeztetés nem tartozik a szoftver hatáskörébe, viszont ha megegyezés születik, azt azonnal rögzíteni kell. Rögzítési joga csak az adminisztrátornak van.
- Minden rendezetlen számláról a rendszer küldjön bizonyos időközönként emlékeztetőt/fizetési felhívást az ügyfélnek, megadott számú vagy értékű rendezetlen számla után az ügyfél további rendeléseket nem adhat le, amíg azokat nem rendezi. Az adminisztrátor ezt felülbíráhatja. A számlázás és azok teljesítésének ellenőrzése nem tartozik a rendszer hatásköre, csupán annak megtörténtének jelzésére van lehetőség.
- A megrendelések nem módosíthatóak, de lehetséges az újrafelvétel és bármely megrendelés törölhető, ha az még nincs teljesítve.
- A megrendelések teljesítésének megszervezése és bérszámfejtés nem a szoftver feladata.
- A rendszer nyilvántartja a megrendeléseket teljesítő alkalmazottakat, de az alkalmazottaknak nincs hozzáférésük a rendszerhez.
- A rendszer a helytelen használatot próbálja megelőzni, a helyes használatot a rendszerben sugó segítse.

2.1.3 Szakterületi fogalmak

Cég struktúra:

A cég fogadja a hozzá érkező megrendeléseket, és ha lehetséges teljesíti azokat. Webes megrendelés minden esetben csak és kizárólag regisztrált és nem korlátozott ügyfél számára lehetséges. Egy ügyfél elvileg akármennyi megrendelést leadhat, de bizonyos számú rendezetlen számla után a rendszer automatikusan megvonja az adott ügyfél rendelési jogát. A rendszer másik oldalán az adminisztrátorok állnak, akik a cég alkalmazottai.

Adminisztrátorok feladatai:

Az adminisztrátorok feladata felvenni az új rendeléseket, rögzíteni a webes rendeléseket, kiírni a számlákat, jelezni a rendszer felé a megtörtént befizetéseket. További feladatai az új ügyfelek regisztrálása, a már meglévők adatainak módosítása, valamint jogában áll az ügyfél státuszának módosítása is, továbbá a már rögzített megrendelések törlése és újrafelvétele is.

A Rendszeradminisztrátor:

Felügyeli a rendszer működését, karbantartását. Betekintést nyerhet az egyes komponensek tartalmába. Ő felügyeli és menedzseli az adminisztrátorokat, üzleti folyamatokhoz nincs hozzáférése.

Az ügyfél:

Az ügyfelek regisztráció útján kerülnek a rendszerbe, adataikat a rendszer tartalmazza, így megrendeléskor csak a megrendelésre vonatkozó információkat kell megadnia. Joga van Webes megrendeléseket leadni a már említett szabályozott formában. Belépve a saját profiljába látja a legfrissebb neki szóló üzeneteket, aktuális számláit és megrendeléseit.

A megrendelés:

Az ügyfél jelzi az igényét valamilyen szolgáltatásra és amennyiben annak teljesítése lehetséges, rögzítődik a rendszerben. A teljesítést kizáró okok lehetnek:

- eszköz vagy emberi erőforrás hiánya illetve nem megfelelő időpont,
- az ügyfél státusza nem teszi lehetővé a megrendelést.

2.2 WebRatio, az első lépések

A WebRatio többféle beépített sémát tartalmaz. Ezen sémák valamelyikét használva kapunk egy kezdeti állapotban lévő munkaterületet és a fejlesztés a webes alkalmazás modelljének elkészítéséből áll. Mint azt már korábban írtam, a WebML XML formátumot használ, így az elénk táruló színes grafikus diagramrendszert a WebRatio folyamatosan XML fájlba képezi le. Jelen esetben ez a „Model.wr” fájl, ami a projekt könyvtárában található.

A rendelkezésre álló sémák közül én a Web Project-et fogom használni, egy párbeszéd panelen megadható a projekt neve, munkakönyvtára illetve az adatbázis kapcsolatra vonatkozó információk. Az így létrehozott modell három fő modulból áll: Adat modell (Data model), Web modell (Web Model), adatszolgáltatók és kiszolgálók (Service/Data Providers).

Mivel már az üres projekthez tartozó „Model.wr” fájl is elég terjedelmes, ezért csak kiemelnék belőle néhány fontosabb részt. Az első két sor nem szerves része a projekt leírásnak, csupán két XML deklaráció, az első magát az XML dokumentumot deklarálja, a második pedig a WebML verzióját. Az utána következő rész annál fontosabb, az XML dokumentum gyökere a <WebProject> elem, aminek attribútumai definiálják a db, gr, layout névtereket. Értelemszerűen a db az adatbázis-kapcsolathoz tartozik, gr a modell diagramjának grafikus megjelenéséhez, a layout pedig a generálandó oldalak megjelenéséhez.

```
<?xml version="1.0" encoding="UTF-8"?>
<?webml version="4.0.3"?>

<WebProject xmlns:db="http://www.webratio.com/2006/WebML/Database"
xmlns:gr="http://www.webratio.com/2006/WebML/Graph"
xmlns:layout="http://www.webratio.com/2006/WebML/Layout"
outputPath="{webapps_loc}/{project_name}" gr:showUnitContent="true"
gr:showTooltip="true" httpPort="8080" httpsPort="8443" control="">
.
.
.
</WebProject>
```

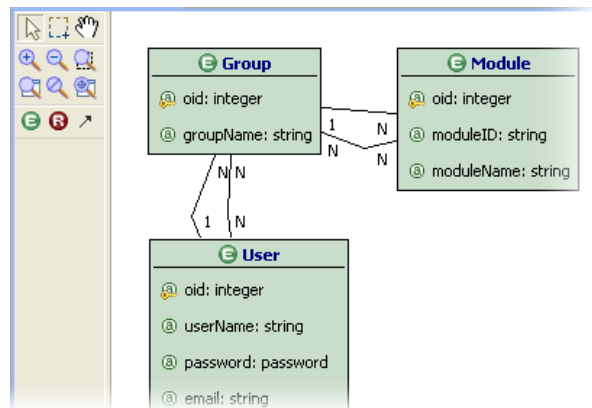
A <WebProject> elemnek három közvetlen gyermeke van: <DataModel>, <WebModel>, <ServiceDataProviders>, melyek a már említett három előre elkészülő modul adatait írják le.

2.3 Adatmodellezés

2.3.1 Az adatmodellezés elmélete

Az adatmodell leírása egy módosított ER-modell¹ segítségével történik. Az eszköztrendszer és a jelölési konvenciók egyszerűek. Az egyedeket téglalapok ábrázolják, minden téglalap tartalmazza az adott egyed nevét és megjelenítheti annak attribútumait is. Az egyedek közötti kapcsolatokat folytonos vonal jelöli a kapcsolat nevével címkézve, a kapcsolatok binárisak, számosságukra és szorosságukra vonatkozó információként minimum és maximum érték megadása szükséges. A minimum/maximum érték azt mutatja, hogy a kapcsolatban résztvevő egyedtípusok részéről legalább/legfeljebb hány egyedelőfordulás vesz részt a kapcsolatban. A minimum és maximum a 0, 1, N értékek valamelyikét vehetik fel.

2.3.1.1 Adatmodellezés WebRatio-ban



3. ábra Az adatmodellező felület

Az adatmodellezés a szerkesztőablak Data Modell fülének kiválasztásával történik. Mivel a WebRatio a WebML hivatalos fejlesztőkörnyezete (pontosabban CASE környezete), ezért az adatmodell tervező felület teljes egészében támogatja az előbb leírt módszertant, sőt kicsit többet is. Három elemből kell felépíteni az adatmodellt, egyed (, kapcsolat () és

¹ Egyed-kapcsolat modell (Entity-Relationship model)

hierarchikus kiterjesztés (generalization ). Az egyedeknek két típusa van, perzisztens (persistent), amit zöld négyzet jelöl és tranzien (volatile/session-scope), amit lila négyzet jelöl. A perzisztens egyedek mint táblák meg jelennek az adatbázisban, a tranzien egyedek viszont nem, azok csak egy munkamenet (session) erejéig léteznek. A táblák közötti vonalak a kapcsolatokat, a nyilak pedig a hierarchikus kiterjesztéseket jelölik. Ha a *Properties* ablak meg van nyitva (Window/Show view/Properties), akkor a modell bármely részére kattintva a *Properties* ablakban megjelennek a kiválasztott elem részletes beállítási lehetőségei.

2.3.1.2 Egyedek

Minden újonnan létrehozott egyednek csak egy *oid* nevű, *integer* típusú kulcs attribútuma van. További attribútumok hozzáadására két módszer van. Jobb gombbal az egyedre kattintva a helyi menüben megtalálható többek között az „*Add Attribute*” és az „*Edit Entity*” menüpontok. Az „*Add Attribute*” hatására megjelenik egy új attribútum az egyednél, aminek a neve és az adattípusának megadását kell csak elvégezni. Több attribútum hozzáadásához azonban ajánlott a másik menüpontot használni. Az „*Edit Entity*” egy párbeszédpanelt takar, amelyen megjelennek az egyed attribútumai, azokat lehet módosítani, törölni, újakat hozzáadni illetve sorrendjüket cserélgetni, majd az „*OK*” vagy „*Cancel*” gombokkal lehet menteni vagy elvetni a módosításokat. További beállításokra itt is a *Properties* ablakban van lehetőség.

2.3.1.3 Kapcsolatok

A **WebML**-ben és ebből adódóan a **WebRatio**-ban is az egyedek közötti kapcsolatok fokszáma pontosan kettő, tehát az egyes kapcsolatokban résztvevő egyedek száma kettő, bár megengedett az az eset, amikor az egyed rekurzívan önmagával áll kapcsolatban. Kapcsolat létrehozása a tervező felületen a  ikonra kattintással, majd a kapcsolatban résztvevő egyed(ek) összekötésével valósítható meg és mint a legtöbb összetevőnél jelen esetben is a *Properties* ablakban lehet a beállításokat megadni. Az újonnan létrehozott kapcsolatok alapértelmezetten *many-many* maximális egyedelőfordulás értékekkel rendelkeznek, ez módosítható *one-many* illetve *many-one*-ra. Az értékpáros jelöli az egyedek kapcsolatban

résztevő egyedelőfordulásainak mennyiségét. Értelmszerűen a *many* többet, a *one* pedig egyet jelent, ami a leképezésnél kap majd nagy hangsúlyt.

Másik módja az egyedek összekapcsolásának a hierarchikus kiterjesztés (generalization). Kiterjesztés, mert egy E1 egyedhez egy E2 egyed kapcsolódik, és hierarchikus, mert az E1 egyed szuper egyed (super entity) lesz az E2 pedig az alárendelt egyed (sub-entity). Az összekapcsoláshoz ki kell jelölni az E2 minden kulcs attribútumához egy-egy E2-beli attribútumot és az E1-beli egyedelőforduláshoz akkor kapcsolható egy E2-beli egyedelőfordulás, ha ezek az attribútum párok megegyeznek. Gyakorlatban úgy fogható fel, mint egy előre definiált *join* művelet, ami a szuper egyedhez implicit módon mindig elérhető.

2.3.1.4 Egyedek és kapcsolatok leképezése az adatbázisba (Data Mapping)

A megtervezett strukturális modellt a WebRatio adat feltérképező modulja (data mapping modul) képezi le adatbázis objektumokra, a működését a hozzá kapcsolódó szinkronizáló eszközzel (Synchronizer tool) irányíthatjuk. A szinkronizáló eszköz egy varázsló jellegű programegység, amivel kiválaszthatjuk, azokat a tételeket, amiket exportálni vagy importálni szeretnénk. Tapasztalataim alapján, a WebRatio megjegyzi, hogy mely adatmodell elemekhez mely adatbázis objektumok tartoznak. Ennek eredményeképpen a szinkronizáló eszköz mindössze azokat az elemeket ajánlja fel importálásra vagy exportálásra, aminek nincs megfelelője a másik oldalon. Exportáláskor minden perzisztens egyedből adatbázis tábla lesz, az egyed attribútumai pedig a nekik megfelelő típusú mezőkként jelennek meg, és a kulcs attribútum(ok) alkotják a tábla elsődleges kulcsát. A tranziens egyedek nem jelennek meg az adatbázisban, mint ahogy a hierarchikus kiterjesztés sem.

A kapcsolatok leképezése az egyedenként maximálisan résztvevő egyedelőfordulások számától függ, a következőképpen:

- *many-many* típusú kapcsolat esetén a két összekapcsolt egyed E1 és E2, és a leképezésükből született táblák T1 illetve T2. Az E1 és E2 közötti kapcsolat egy újabb, úgynevezett kapcsolótáblaként jelenik meg, aminek a neve T1_T2. és ez a két mező együtt elsődleges kulcsot alkot, továbbá mint külső kulcsként működik a két összekapcsolt tábla elsődleges kulcsaira nézve.

- *one-many* vagy *many-one* típusú kapcsolat esetén nem jön létre további újabb tábla, hanem külső kulcsokkal és kapcsoló mezők használatával kerül be a kapcsolat az adatbázisba.

Jelen esetben a kapcsolótábla neve a megjelenítendő kapcsolat neve lesz, mezőit pedig az összekapcsolt táblák elsődleges kulcs mezőinek a „másolata”. Másolat alatt azt értem, hogy veszi az adat feltérképező modul az egyik összekapcsolt táblának az összes olyan mezőjét, ami elsődleges kulcsot alkot, majd hozzáveszi a másik tábla elsődleges kulcsának mezőit. Az új táblában a mezők neve [eredeti_tábla]_[eredeti_mezőnév] formában jelenik meg, és az összes mező együtt alkotja az elsődleges kulcsát a kapcsoló táblának, ebből adódóan minden sor a kapcsolótáblában egyedi, ráadásul, az összekapcsolt táblák elsődleges kulcsainak megfelelő két mezőcsoport külső kulcsként működik az összekapcsolt táblák elsődleges kulcsaira hivatkozva.

2.4 A hiperszöveg modell és részei (Hypertext modell)

A WebML hiperszöveg modelljének legfőbb összetevői az *oldal nézetek* (site views), *területek* (areas), *oldalak* (pages), *egységek* (units), *műveletek* (operations), *hivatkozások* (links), *munkamenet/alkalmazás változók* (session/application variables) [4].

A teljes hiperszöveg *oldal nézetekből* épül fel, ami egy diagram az *oldal nézethez* tartozó *oldalak* kapcsolatáról, és esetleges *területekre* osztásáról. Több szempontja lehet az *oldal nézetekre* bontásnak, az egyik a hiperszöveg funkcionálisan összetartozó nagyobb darabokra bontása, a másik szerint így lehet különválasztani az eltérő jogosultságokkal és szerepkörökkel rendelkező felhasználók vagy felhasználó csoportok által használható funkciókat, *oldalakat*. Egy harmadik szempont szerint figyelembe kell venni az eltérő platformot használó felhasználókat is, így az ugyanabba a csoportba tartozó felhasználók különböző platformról megközelítve a rendszert más *oldal nézetet* használnak. Manapság, amikor a mobiltelefonok és kézi számítógépek jelentős része rendelkezik internetes kapcsolattal, nem törvényszerű, hogy a felhasználó PC-ről próbálja elérni az adott rendszert.

Az *oldal nézeteket* tovább lehet bontani *területekre*, azaz összetevők valamilyen szisztéma szerinti csoportjára, ahol az összetevők lehetnek *oldalak*, *műveletek*, *egységek*, *hivatkozások*.

2.4.1 Oldalak

A szerkezeti egységek következő szintjén az *oldal* áll, ami a modell legkisebb önállóan is megjeleníthető és működő eleme. A gyakorlatban inkább nevezhető konténer elemnek, ami közrefogja, és működési környezetet biztosít az *egységek*, *műveletek* és *hivatkozások* számára. A lapokat szerepük szerint jelzőkkel lehet illetni. Minden oldal nézetnek vagy egy egyedi, alapértelmezett oldala, a kezdőoldal (home page), amit a WebML szerint kis „h” betűvel kell jelölni, a WebRatio viszont egy kis házikót tesz az oldal neve mellé. Ezzel az oldallal találkozik először a felhasználó, ha belép egy oldal nézetbe, ilyen például a felhasználó személyes kezdőlapja, amikor belép a WebRendelő rendszerbe.

Létezik az oldalaknak egy harmadik megjelölése, ez a területjelzés (landmark), WebML szerint „l” jelölné, a WebRatioban pedig „L” jelöli az oldal/terület bal alsó sarkában. A területjelzés eredménye, hogy a területjeles összetevőre az oldalnézetben belül bármely oldalról közvetlen hivatkozással érhető el. Mivel a hivatkozások célja területek is lehetnek, a területeknek egy alapértelmezett oldal beállításával lehet megadni, hogy melyik oldal legyen a hivatkozás célja. Az alapértelmezett oldal (default page) jelölésére a WebML kis „d” betűt javasol a jobb alsó sarokba, a WebRatio pedig nagy „D” betűvel jelzi ezt.

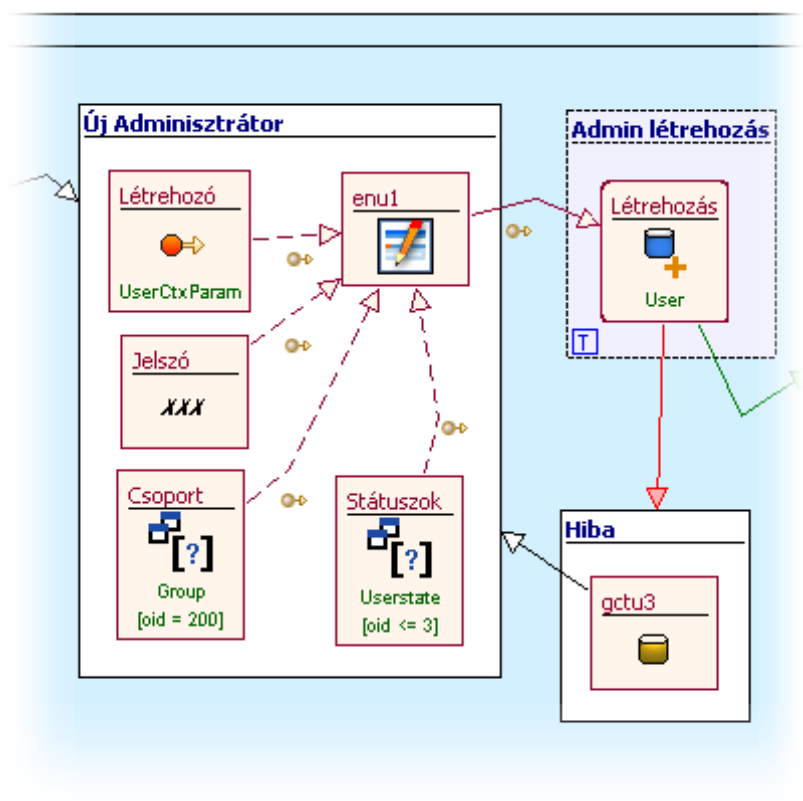
Az oldalak közötti navigáció és az adatáramlás hivatkozásokkal van biztosítva, amik az oldalakat és egységeket kötik össze. Többféle működési szempont alapján több csoportra lehet a hivatkozásokat bontani. A modell elméleti szempontjából érdekes csoportosítás szerint vannak környezetfüggő hivatkozások (contextual links) és nem-környezetfüggő hivatkozások (non-contextual links). A nem környezetfüggő hivatkozások a tartalmak közötti navigálást teszik lehetővé. Az környezetfüggő hivatkozások az összekötött egységek közötti adatátvitelt biztosítják a navigáció mellett.

Egy másik hasonló, de a hivatkozásokat inkább gyakorlati oldalról megközelítő felosztás szerint közönséges (normal link) és szállító (transport link) módban működő hivatkozások vannak. A szállító hivatkozás tisztán csak az adatok átviteléért felelős, a működő alkalmazásban nem látható, és csak környezetfüggő típusa létezik. Ezzel szemben a közönséges hivatkozásnak van környezetfüggő és nem környezetfüggő változata is. A nem környezetfüggő változat lehet egy egyszerű, például egy „vissza” horgony (anchor). A környezetfüggő változat a navigálás mellett az egységek közötti egyirányú

információáramlásról is gondoskodik. Ennek egy megjelenési formája például egy űrlap elküldés gombja.

2.4.2 Hivatkozások megjelenési formái

A hivatkozások jelölése a modellben nyilakkal történik, a közönséges hivatkozást folytonos, a szállító hivatkozást pedig szaggatott szárú nyilak jelzik. A közönséges hivatkozásokat további



4. ábra Az összes hivatkozási típus egy hiperszöveg modell darabban

három típusra lehet osztani, ezek a hivatkozás (link), OK hivatkozás (OK link), KO hivatkozás (KO link). A hivatkozás a közönséges hivatkozás alapesete, erre az imént leírtak érvényesek, folytonos fekete nyíl jelöli. Az OK hivatkozás jelölése folytonos zöld nyíl, a KO hivatkozás jelölése folytonos piros nyíl (4. ábra). A különlegességük az, hogy némely egységnek és műveletnek van két olyan határozott kimenete, amik közül az egyikre az OK hivatkozást lehet kapcsolni a másikra pedig KO hivatkozást lehet kapcsolni. Ha ez a művelet vagy egység sikeresen fejezte be a működését, akkor az OK hivatkozás mentén navigál

tovább az alkalmazás, viszont ha valami miatt sikertelen volt, a KO hivatkozást választja. Ez a két hivatkozástípus kölcsönösen és egyértelműen kizárja egymást, és minden olyan összetevőnél, ahol a használatuk támogatott legfeljebb egy OK és legfeljebb egy KO hivatkozás lehet.

Mind a környezetfüggő közönséges, mind pedig a szállító hivatkozás esetében meg kell adni, hogy a forrás (source) egység/művelet mely paramétereinek értéke a cél (target) egység/művelet mely paramétereivel kapcsolódnak össze. Az összekapcsolás (coupling) két módon történhet, automatikus (automatic/default coupling) vagy kézzel (manual coupling), ezt részletesen az összekapcsolási párbeszédpanelen lehet beállítani (coupling dialog). Automatikus esetben a paraméternevek egyezés szerinti összekapcsolása történik, kézi esetben pedig egyedileg lehet a cél paramétereknek értéket adni, ami lehet egy forrás paraméter, egy konstans, NULL vagy akár az is előfordulhat, hogy nincs hozzárendelt érték/paraméter a hivatkozásban.

2.4.3 Egyedelőfordulások szűrése

A WebML és ezzel összhangban a WebRatio is számos egységet alkalmaz, de mielőtt ismertetném ezeket nem árt pár szót ejteni a forrásról (source), a válogatóról (selector) és a feltételekről (conditions).

A forrás egy egyed, amelynek az egyedelőfordulásaival dolgozik az adott egység. A WebRatio-ban forrást használó egységnek a *Properties* ablakában kell megadni az egyedet. Az egyedelőfordulások megjelenítését végző egységek esetében további opcióként megadható, hogy az összes helyett, az attribútumoknak csak egy részhalmazát jelenítse meg, esetleg egy előre megadott sorrendben.

 A válogató egy olyan predikátum, ami eldönti egy egyedelőfordulásról, hogy megfelel-e a feltételeknek, ezáltal szűri az egyedelőfordulásokat, és csak az összesnek eleget tevők mennek át a válogatón. Általánosságban elmondható, hogy logikai feltételek együtteséről van szó, amelyek mindegyikének teljesülése szükséges egyszerre egyedenként vagy kapcsolati szerepkörönként.

❏ A válogatót irányító logikai kifejezések a feltételek használatával adhatók meg. Minden válogatóhoz legalább egy feltételt meg kell adni, különben az működésképtelen, hiszen nem tudja, hogy mi alapján kell szűrnie. A WebRatio három féle feltételt alkalmaz:

❏ A kulcs-alapú feltétel (key condition): egy egyedelőfordulás akkor tesz eleget a feltételnek, ha kulcs attribútumainak valamelyike megegyezik a keresett kulcsértékkel. Természetesen ha csak egy kulcs attribútuma van, akkor minden esetben ennek a kulcs attribútumnak kell megfelelnie.

❏ Az attribútum-alapú feltételt (attributes condition): egy egyedelőfordulás akkor tesz eleget a feltételnek, ha az előre kiválasztott attribútumára a logikai kifejezés igaz. A logikai kifejezések sokszínűek, többféle adattípus kezelésére is fel lettek készítve, így a null értékre vonatkozó predikátumtól kezdve a karakteres és numerikus értékeken értelmezett predikátumok is megtalálhatók. Csak ízelítőképpen, a teljesség igénye nélkül: „Is Null”, „Is Not Null”, ami null értéket vizsgál, a „Less Than”, „Equal” és „Greater Than” numerikus értéket hasonlít össze egy előre beállított konstanssal. Továbbá a karakteres típusú értékek sztringliterállal történő összehasonlítását a „Begins With”, „Contains”, „Ends With” predikátumokkal, illetve ezek kis/nagybetű érzéketlen változataival lehet elvégezni.

❏ A kapcsolati szerepkör alapú feltételt (relationship-role condition): a kapcsolati szerepkör feltételnek azon egyedelőfordulások felelnek meg, melyek az előre beállított kapcsolati szerepkör alapján kapcsolatban állnak egy másik egyedelőfordulással.

2.4.4 Érvényesítési szabály (validation rule)

Az érvényesítési szabályok logikai kifejezések, azt lehet megszabni, hogy milyen értékeket kaphat egy beviteli mező. Ha egy adatbeviteli mezőhöz érvényesítési szabály van hozzárendelve, akkor amíg a tartalma ki nem elégíti a logikai kifejezést, addig érvénytelen. Kiváló példa erre egy elektronikus űrlap. Minden beviteli mezőhöz meghatározható hogy milyen értékintervallumokat kezeljen, ne legyen üres, vagy adott formátumban legyen. Néhány beépített szabályt a WebRatio is tartalmaz, de ha nincs megfelelő, akkor magunknak kell definiálni minden mezőhöz, predikátum/összehasonlítási érték/hibaüzenet.

2.4.5 Egységek (Units)

Először is, csak ismétlésképpen: az egységek (unit) és műveletek (operation) azok az összetevők, amikből az oldalak felépülnek.

Az egységek tartalmi elemeket jelenítenek meg, a művelek pedig a tartalmi elemek közötti interakciókat valósítják meg. Az egységek strukturális séma (structural shema) egy vagy több egyedelőfordulását reprezentálják, kiválasztásuk tipikusan az egyedek attribútumai és kapcsolatai felhasználásával működő lekérdező eszközökkel történik.



Adat egység (data unit):

Az egyedek megjelenítésére használható, az adat egység helyén a képernyőn egy egyedelőfordulás kiválasztott attribútumai jelennek meg, meghatározott sorrendben. Kötelező egy forrás és egy válogató beállítása. A forrás a *Properties* ablakban állítható be, az „*entity*” értéknél, egy új válogató pedig automatikusan tartozik hozzá. A válogató kulcs-alapú feltételt használ, kulcs értékét bemenő paraméterként várja az egység. A kötelező feltételen túl további feltételek is megadhatók.



Többes-adat (multidata unit):

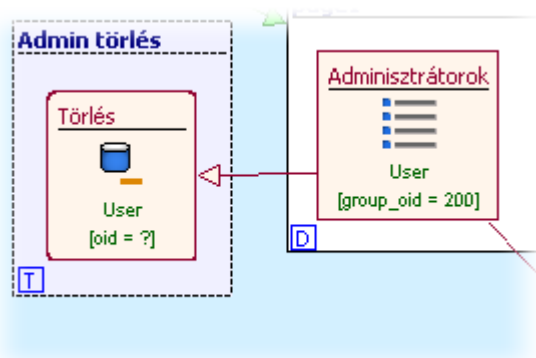
Az adat egység kiterjesztett változata, annyiban különbözik tőle, hogy egyazon egyed több előfordulását is képes egyszerre kezelni, megjeleníteni. A forrás megadása ennél az egyednél is kötelező, de mivel egyszerre több egyedelőfordulást is képes megjeleníteni, a válogató opcionális, ezáltal a típusa is választható, illetve több eltérő típusú is megadható.



Index egység (index unit):

Az index egység a forrásként megadott egyedelőfordulásai attribútumainak egy részhalmazát listázza, megadva a felhasználónak a lehetőséget arra, hogy kiválasszon egyet. A kiválasztott egyedelőfordulásra a kulcsával hivatkozik a rendszer. A forrás megadása kötelező, ez a jól ismert *Properties* ablakban lehetséges, illetve válogatók megadásával lehet szűkíteni az egyed előfordulásait. Például ilyen eset látható az 5. ábrán. Az adminisztrátorok nevű index egység listázza az össze Adminisztrátort, azaz az olyan „*User*” egyedet, amelynek a „*group_id*” attribútumának értéke 200. A balra

mutató nyíl, egy hivatkozás - a képernyőn egy „Törlés” feliratként jelenik meg minden – ami továbbadja az egyed kulcsát a törlést elvégző műveletnek.



5. ábra Egyed törlése index egységen keresztül

Többválasztásos index egység (multichoice index unit):

Az index egység kiterjesztett változata, jelölőnégyzetek használatával lehetővé teszi egyszerre több egyedelőfordulás kiválasztását. A forrás megadása kötelező, a válogatók megadása opcionális, és lehetőség van egy érvényesítési szabály megadására is.

Hierarchikus index egység (Hierarchical Index Unit):

Az index egység egy másik szemléletű kiterjesztése, egyszerre több egyed előfordulásainak listázását biztosítja, feltéve, hogy azonos kapcsolati szerepkörben vannak. Egy egyed mint forrás megadása kötelező, ez lesz az alapértelmezett, legfelső szinten, illetve további hierarchikus index szintek (hierarchical index level) adhatók meg, ami egy azonos kapcsolati szerepkörrel rendelkező újabb egyedet jelent. Mindezen túl további első szintű egyedek adhatók hozzá, illetve az egyes szintekhez további alsóbb szintek adhatók meg, azonban minden alsóbb szintnek azonos kapcsolati szerepkörben kell lennie a felette lévő szinttel. Végül a szintek definiálása után előáll egy megjelenítési hierarchia, amiben a legfelső szinten lévő egyed minden – vagy az esetleges szűrések után megmaradó – előfordulása és egyedelőfordulásonként a kapcsolatban álló alsóbb szintek megfelelő egyedelőfordulásai. Mint az előző egységekben itt is lehetőség van válogató és feltételek hozzáadására, ami minden szinthez külön is lehetséges, ezzel további szűréseket végezve.

Görgető egység (Scroller Unit):

A görgető egység listába szervezi az egyed előfordulásai, és egyszerre több is elérhető el rajta keresztül, de nem jeleníti meg, hanem a kulcsértéke(i) mint az egység kimeneti paramétere áll rendelkezésre. A listában lehet mozogni, minden listaelemről közvetlenül négy másik érhető el az első, az előző, a következő és az utolsó, kivéve, ha az első vagy az utolsó elemen állunk, vagy éppen egy vagy nulla elemű a lista. Hasonlóan a többi egységhez a forrás megadása itt is kötelező, és az egyed szűkíthető válogatókkal és feltételekkel. Összekapcsolva egy adat egységgel, egy olyan oldalrészletet kapunk, amivel egyesével meg lehet tekinteni például a felhasználók adatait.

Adatbeviteli egység (Entry Unit)

Az űrlap alapú adatbevitelt biztosít, lényegében egy konténer elem. Az adatbevitel mezőkön keresztül történik, ezek az alapvető űrlapelemek adatbeviteli mező (field), legördülő lista (selection field), jelölőnégyzet (multi selection field).

Ezeket felül használ még a WebRatio néhány ezekből származtatott egységet:

Többes adatbeviteli egység (Multi Entry Unit):

Lényegi részét tekintve megegyezik az adatbeviteli egységgel, az eltérés mindössze annyiból áll, hogy ennél a típusú egységnél lehetőség van egyszerre több mező értékének a módosítására.

Rekurzív hierarchikus index egység (Recursive Hierarchical Index Unit)

A hierarchikus index egység, ahol a kapcsolt egyedek hierarchiája a mérvadó, az egyedek közötti gyerek-szülő kapcsolatok határozzák meg a megjelenő egyedelőfordulásokat és azok megjelenési szerkezetét.

Erős index egység (Power Index Unit)

A sima index egység továbbfejlesztett változata, belekerült az egyedelőfordulások görgetésének és dinamikus rendezésének lehetősége.



Többes üzenet egység (Multi Message Unit)

A bemenetére kapott karakterlánc tömböket (string array) jelenít meg.



Üres tartalom egység (No Op Content Unit)

Ez egy olyan tartalmi egység, ami legfeljebb a nevét jeleníti meg, de például előszeretettel használtam, mint fix pontok, ahová a hivatkozásokat lehet kapcsolni, főleg, mivel megjelenik a rács nézeten is.

2.4.6 Műveletek (Operations)

Az egyre újabb WebML verziókban specifikálva lettek az információk megjelenítésén túli felhasználói lehetőségek is, értem ez alatt az alkalmazás alapjául szolgáló adatok módosítását, törlését vagy éppen újak hozzáadását, és ezek megvalósítására születtek meg az műveleteknek nevezett összetevők. Mivel a WebRatio a műveletet is egységeknek nevezi, így minden művelethez létezik egy egység, ami azt megvalósítja, illetve magában foglal számos további műveletet és egységet, valamint hangolható mechanizmusokat a futási hibák kezelésére, kiterjesztési lehetőséget a felhasználóknak, hogy egyéni egységeiket hozzáadhassák a rendszerhez. Az „alapműveletek”:



Létrehozás (Create Unit)

Új egyedelőfordulás létrehozása, az egyed definiálásakor be kell állítani kezelendő egyedet, létrehozáskor pedig az új előfordulásnak az attribútumait pedig bemenő paraméterként kell megadni



Módosítás (Modify Unit)

Egy vagy több egyedelőfordulás attribútumait módosítja. Az előfordulásokat válogatóval lehet szűrni és új értékeit bemenő paraméterként kapja meg az egység. A válogatók mindhárom feltétel típust támogatják.



Törlés (Delete Unit)

Az előre beállított egyed kulccsal hivatkozott előfordulását vagy előfordulás halmazát törli. Ennél valamint két előző műveletnél is létezik két specifikált kimeneti pont,

amihez az OK és a KO hivatkozásokat lehet kapcsolni. Kapcsolható továbbá az egységhez egy szállító hivatkozás, ez a hibakezelésnél vagy az egyéb visszajelzéseknél lehet hasznos.

Összekapcsolás (Connect Unit)

Két egyedelőfordulás halmaz között definiál kapcsolati szerepkört. Mindkét halmazhoz létre kell hozni egy-egy válogatót, ami alapján az összekapcsolás történik. Az egyik egyed neve forrás válogatónak (source selector) nevezi, a másikat pedig cél válogatónak (target selector). Mind a forráshoz mind a célhoz kapcsolt válogatók feltételei lehetnek kulcs, attribútum, valamint kapcsolati szerepkör alapúak.

Szétkapcsolás (Disconnect Unit)

Az előző egység által létrehozott kapcsolatokat törli. Az egyedelőfordulás halmazokat a kapcsolódáshoz hasonlóan kell megadni, csak itt a válogatóknak megfelelő két előfordulás halmaz közötti kapcsolat törlődik.

Újrakapcsolás (Reconnect Unit)

Az újrakapcsolás egység az előző két egység kombinációja, természetesen csak már összekapcsolt előfordulás halmazokkal végezhető. A két egyedelőfordulás halmaz azonosítása ugyanúgy működik. Első lépésben az egység a már meglévő kapcsolati szerepkört felbontja, és egy újat hoz létre, ami különbözni szokott az előzőtől, de ez nem kikötés. Lehetőség van azonban arra is, hogy az előfordulás halmazoknak csak egy részhalmazát válasszuk le, erre való a szétkapcsolási cél válogató (disconnect target selector).

Elágaztatás művelet (Switch Unit)

A programozásban jól ismert switch vagy más néven case szerkezet megvalósítása. Az egység létrehozásakor meg kell adni egy egymást kölcsönösen kizáró feltételhalmazt. A feltételek mindegyikéhez rendelhető egy-egy hivatkozást, ami azt mutatja, hogy az adott feltétel teljesülése esetén hova navigáljon a rendszer. Tehát sematikusan úgy működik, hogy kap egy értéket a bemenetre, azt összeveti a feltételhalmazzal, és az egyező feltételhez rendelt irányba navigál tovább a rendszer.

Null érték vizsgálat (*Is Not Null Unit*)

A null érték vizsgálata a bemenő paraméteren kapott értékre történik. Az egység kimenete az OK és KO hivatkozások kapcsolódási pontjai. Ha a bemenő paraméterként kapott érték null érték, akkor a KO hivatkozáson, egyébként az OK hivatkozáson navigál tovább az alkalmazás.

Üres művelet (*No Op Operation*)

Nem alapértelmezett WebML művelet, úgynevezett helykitöltő szerepet szolgál, ugyanis nem végez semmilyen műveletet. Olyan szituációkban hasznos, amikor a modell megköveteli, hogy legyen valami az adott helyen, de az üzleti logika szempontjából ott semmilyen teendő nincs.

Bejelentkezés (*Login Unit*)

Az alkalmazás felhasználói számára szolgáltat bejelentkezés funkciót név/jelszó páros megadásával.

Kijelentkezés

A bejelentkezett aktuális felhasználó számára kijelentkezés funkciót szolgáltat. A bejelentkezésről és kijelentkezésről bővebben a hitelesítésnél írok.

2.4.7 Hasznos kiegészítő egységek (*Utility Unit*)

^{xxx} Jelszó egység (*Password Unit*)

Megadott karakter hosszúságú jelszót generál, amit kimenetéről szállító hivatkozással lehet kinyerni. A generált véletlenszerű jelszó tartalmazhatja az angol abc kis és nagy betűit valamint decimális számjegyeket.

Idő egység (*Time Unit*)

Az aktuális rendszeridő lekérdezésére illetve idő és dátumbélyegzők előállítására szolgál. Elérhető külön az aktuális dátum és idő bármely komponense (óra, perc, stb.) vagy komplett időbélyegzők.

[?] Válogató egység (Selector Unit)

Mint azt korábban tisztáztam az egyedek szűrése válogatók segítségével lehetséges. Van azonban egy olyan lehetőség, amikor nem egységekhez kapcsolok válogatókat, hanem a válogató egység segítségével a már szűrt egyedelőfordulás halmazt kapja meg az egység. Hasznos lehet például, ha ugyanazon előfordulás halmazt egy oldalon több egységnek is szűrnie kellene, azonos feltételek szerint.

2.4.8 Globális paraméterek és változók

A globális paraméter (global parameter) adatok átmeneti tárolására szolgáló változó, élettartama egy munkamenet, hatásköre pedig a munkamenethez tartozó oldalnézet. A WebRatio a globális paramétereket környezeti paramétereknek (context parameter) nevezik, és alapértelmezetten van belőle négy előre definiált: CountryISOCtxParam a lokalitás, GroupCtxParam a felhasználó csoport, LanguageISOCtxParam a nyelvi területre, UserCtxParam a felhasználóra vonatkozó beállítást tartalmazza. Továbbiakat is lehet hozzáadni a WebRatio web modelljéhez (web model), jobb gombbal a web modellre kattintva, majd a helyi menüből az „Add context parameter”-t kell kiválasztani, végezetül a már jól megszokott *Properties* ablakban állítható be a típusa és változtatható meg a neve.

A környezeti paraméterek értékeinek manipulálására beépített egységek vannak, ezeknek a létrehozáskor (vagy akár később is) a *Properties* ablakban meg kell adni, hogy melyik környezeti paraméter értékét módosítják. A egységek a következők:

beállító (set unit)

A bemenetére kapott értékre állítja be a *Properties*-ben hozzárendelt környezeti paraméter értékét.

lekérdező (get unit)

Kimenete az előre beállított környezeti paraméter aktuális értéke.

visszaállító (reset unit):

Alaphelyzetbe állítja, eltávolítja a beállított környezeti paraméter értékét.

2.5 Modellezési példák

A rendelkezésre álló sémák közül én a Web Project-et fogom használni. A projekt létrehozása egyszerű. A WebRatio elindítása után *File/New/Web Project*. A megjelenő párbeszédablakban megadom a projekt nevét, majd a „Next” gombra kattintva megadhatom az adatbázis kapcsolatra vonatkozó információkat, végül a „Finish” gombra kattintok.

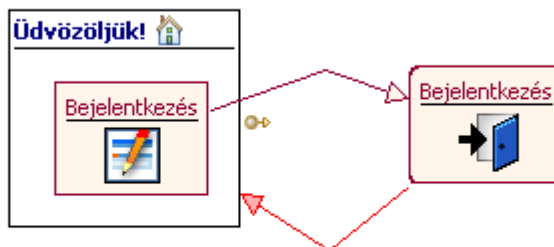
Az így létrejött projekt azonnal tartalmazza Data Model-t, amivel a perzisztens objektumok leírását, létrehozását végezhetjük. A munkaterület alján kiválasztva a „Data Model” fület megkapjuk a „Data Model” szerkeszthető diagramját, ami már tartalmaz néhány létfontosságú elemet, ezek a „User”, „Group”, „Module” egyedek és a közöttük lévő kapcsolatok

2.5.1 Hitelesítés (authentication), az alkalmazás nézetei

Az előbb említett három egyed a hitelesítés alapját képezi, ezen egyedek leképezéséből készült adatbázis táblák tartalmazzák a WebRatio által alkalmazott beléptetés helyes működéséhez szükséges adatok. Csak érdekességgéppen jegyzem meg, hogy ez a három egyed a később létrehozottakkal nem egy helyen van tárolva. A „User”, „Group” és „Module” egyedek a <DataModel> szekció első részében, az adatbázis kapcsolatot leíró elem előtt, míg a többi egyed utána helyezkedik el, ráadásul az első részben leírt egyedek és kapcsolatok nem törölhetők, de részben módosíthatók. Fontos még kiemelni, hogy a modell minden összetevőjének projekt szinten egyéni azonosítója van. Ez az összetevő nevének rövidítéséből és egy számlálóból áll, például „ln10” egy hivatkozást jelent. Az azonosítót a WebRatio automatikus rendeli hozzá minden összetevőhöz, és nem módosítható.

A hitelesítés a bejelentkezés egységen keresztül kezdődik. Az egység feladata, hogy a paraméterként megkapott felhasználónév és jelszó páros alapján keresse ki az adatbázisból a felhasználót, az alapértelmezett csoportját és a csoporthoz tartozó modul azonosítóját. Ezután – ha nem történt hiba – a felhasználót egy munkamenethez hozzárendelve átirányítja a neki megfelelő nézetre. A beléptetést egy nem védett oldal nézetén illik létrehozni, vagy az alapértelmezett beléptető oldalt használni, és minden oldal nézetet védetté nyilvánítani. A

védett oldalnézet jelentése, hogy csak olyan hitelesített felhasználó férhet hozzá, akinek a csoportjához hozzá van rendelve az adott oldal nézet.



6. ábra Beléptetés oldal nézet

Ahogy a 6. ábrán látszik egy beléptető oldal nézet modellje viszonylag egyszerű. Először is van maga az oldal, ami az oldal nézet kezdőlapja (ezt a házikó jelöli az „Üdvözljük!” felirat mellett). Az oldalon ezen kívül csak egy adatbeviteli egység van, ami a felhasználónév/jelszó páros bevitelére szolgál. A hivatkozás, ami az adatbeviteli egységből indul, egy gomb formájában jelenik meg az oldalon, arra kattintva a begépelt felhasználónév és jelszó páros átkerül a bejelentkezés egységhez feldolgozásra. Ha nem hitelesíthető, akkor visszatér bejelentkező oldalra. A bejelentkezés egységhez nem kell OK linket adni, hiszen sikeres bejelentkezés esetén azonnal a felhasználó csoportjához rendelt alapértelmezett oldal nézetre irányít át. Minden felhasználót csoportba kell rendezni, illetve minden csoporthoz egy modult kell hozzárendelni. A „*Module*” táblában adható meg a csoport kezdő nézete, sőt lehet még több is. A „*Module*” táblának három oszlopa van. Az első az *id*, ez az elsődleges kulcs. A második a *modulid*, itt kell megadni, a modul a projekt mely elemét képviseli, illetve a *modulname* a modul nevét jelenti (bár az azonosításban nem vesz részt).

Ahhoz, hogy különböző felhasználócsoporthoz különbözőképpen lássák ugyanazt az információs rendszert, a nézetek szétválasztására van szükség. Az egyes nézeteket felvesszük az adatbázisba mint modult, így módon a felhasználócsoporthoz mindegyikéhez egy modul/nézetet adunk meg, amin keresztül a rendszert látják. Az nézetek függenek a csoportok igényeitől, illetve lehetőségeitől. Például a konkrét esetben: egy ügyfélnek nem engedünk

teljes körű betekintést a rendszerbe, csak a saját adataival dolgozhat, ezzel szemben az adminisztrátornak létszükséglet, hogy több megrendelés illetve megrendelő adatát láthassa.

2.5.2 Területek, oldalak, navigáció, hivatkozások

A rendszer konkrét bemutatásán keresztül, néhány megvalósított funkció segítségével könnyen megérthető a modellből generált alkalmazás alapvető szerkezete és az összetevők közötti navigálás. Mint azt már korábban írtam, a teljes alkalmazást oldal nézetekre kell bontani, az eltérő felhasználói igények és jogosultságok egyszerű kezeléséhez.

A megvalósított projekt négy oldal nézetre lett bontva:

- „*login*” nézet:

A felhasználók bejelentkezését kezelő oldalt egy külön nézetre tettem. Az többitől eltérően ez az egy nézet nem védett, azaz hitelesítés nélkül bárki megtekintheti, sőt ez az alapértelmezett nézet, tehát ha a felhasználó megkísérli elérni az alkalmazást, ez az oldal tárul elő. A hitelesítésnél tárgyalt modell (részlet) ebből a nézetből való.

- „*system_op*” nézet:

A rendszeradminisztrációs teendők ellátásához lett kialakítva. Főbb funkciói az adminisztrátorok kezelése, rendszernapló megtekintése, személyes és körüzetek kezelése. Ez a rendszer nézetének a legfontosabb keresztmetszete. Ebben a nézetben szinte a teljes rendszerbe betekintést nyerünk, láthatjuk a tranzakciókat, jelentéseket nézhetünk, módosíthatjuk az adminisztrátorok adatait. A kezdőlap a rendszeradminisztrátor legfrissebb üzeneteit tartalmazza, a legutolsó naplóbejegyzéseket valamint néhány, a profiljára vonatkozó adatot.

- „*Admin*” nézet:

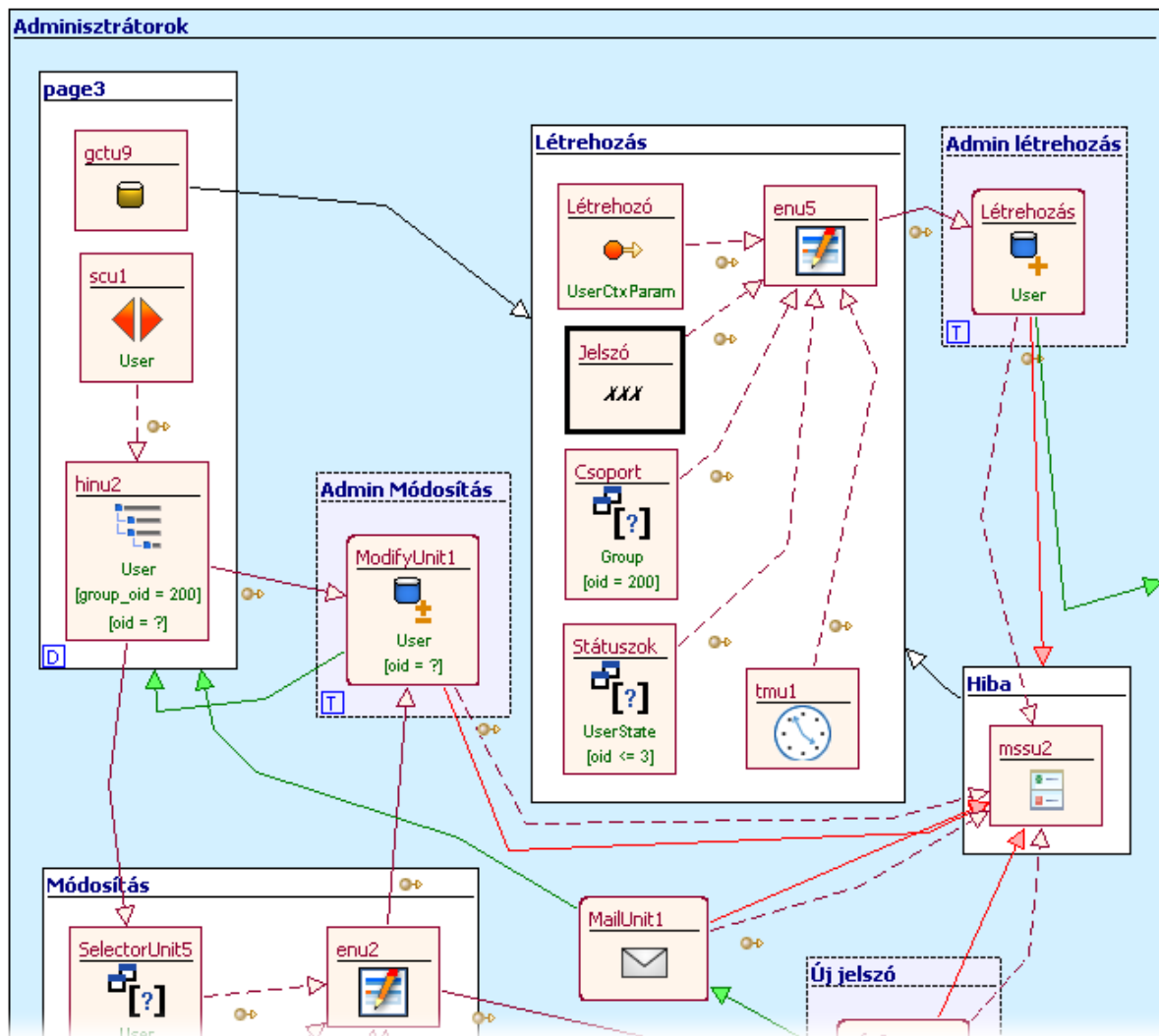
Ebben a nézetben az adminisztrátorok tevékenykedhetnek, a felhasználókat és a megrendelések adatait menedzselhetik, valamint itt is lehetőség van a rendszer többi felhasználójával vagy felhasználó csoportjával kommunikálni üzenetek formájában.

- „*Customer*” nézet:

Az ügyfelek számára kialakított nézet, ez a legszűkebb és legjobban korlátozott keresztmetszete a rendszernek. Nyilvánvaló okokból az ügyfél csak a saját

megrendeléseit látja, üzeneteket nem küldözgethet bárkinek, csak az adminisztrátoroknak illetve a rendszer adminisztrátoroknak, és nincs lehetősége körüzenetek küldésére sem. Új megrendeléseket megkötés indíthat, amíg az adminisztrátorok nem korlátozzák, például sok rendezetlen számla miatt.

És akkor most következzen néhány konkrét eset. A 7. ábrán egy kivonat látható az Adminisztrátorok menedzselését végző területből. A „page3” nevezetű oldal, amint az látható is, a területet kezdőlapja. Található rajta üres tartalmi egység, amiből a létrehozás hivatkozás indul, továbbá egy hierarchikus index és görgető egység kombináció, ami az adminisztrátorokat jeleníti meg a státuszukkal, egyszerre legfeljebb tízet.



7. ábra Adminisztrátorok terület

Ami nem látszik az ábrán, hogy a görgető egység két példányban jelenik meg a generált oldalon, egyszer az index egység felett és egyszer pedig alatta, megkönnyítve ezzel a keresést.

A létrehozás oldal is némi magyarázatra szolgál. Attól függetlenül, hogy olyan sok egység van az oldalon, az „enu5” azonosítóval rendelkező adatbeviteli egységen kívül semmi sem látható. A többi egység ugyanis csak adatokat generál az űrlaphoz.

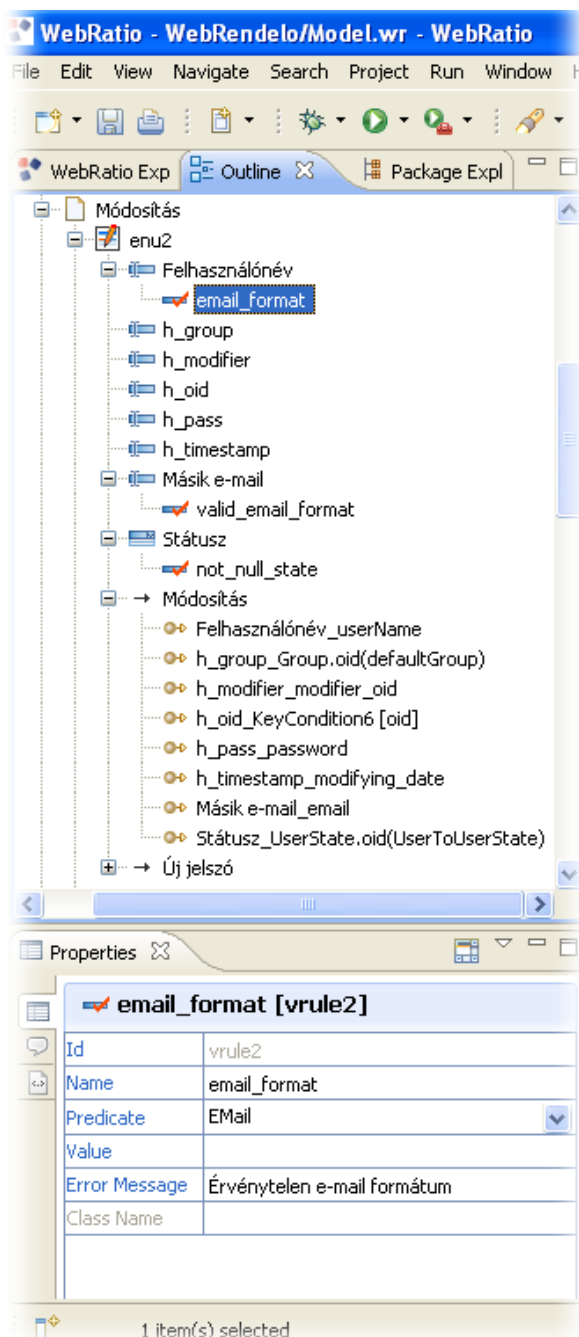
Látható még két, úgynevezett műveleti csoport (operation group), amire mondjuk a csoport kifejezés túlzás, mert csak egy-egy művelet van mindegyikben. Ezek az „Admin módosítás” és az „Admin létrehozás”, és szaggatott vonallal határolt lilás négyzet jelöli. A műveleti csoport lehetővé teszi több művelet egy csoportba foglalását, így például az egy csoportban lévő válogató egységek logikailag egyszerre végzik el a szelektálást.

Megfigyelhető még az OK és KO hivatkozások rengetege is. Minden műveletből indul egy zöld és egy piros nyíl. A piros (KO) hivatkozások egy hibalapra mennek, ahol egy többes üzenet megjelenítő egység írja ki a hibaüzeneteket, amiket a velük párhuzamos, szaggatott nyíllal jelölt szállító hivatkozások visznek át. A zöld (OK) hivatkozások mindig az kezdő oldalra kerülnek vissza, ennek két módja látható, az egyik szerint direkt az oldal legyen a célpont, ekkor magával az oldallal kell összekötni a műveletet. A másik módszerrel a terület legközelebbi széléhez kötöm a hivatkozást, ekkor a művelet helyes elvégzése után a területre navigál ezzel együtt pedig a kezdőoldalra vissza.

2.5.3 Űrlapok készítése, mezők érvényesítése (validation)

Űrlapokat adatbeviteli egységek segítségével lehet létrehozni. Ez önmagában csak egy konténer egység, ehhez létre kell hozni a mezőket és az előre feltöltő és érvényesítő eszközöket. Az adatbeviteli mező (field) egy egyszerű karakteres mező, a tartalom típusa kiválasztható (integer, string, password, stb.), illetve további opciók adhatók meg, miszerint legyen előre beállított (preloaded), módosítható (modifiable), rejtett (hidden). Előre beállított esetben az űrlap betöltődésekor a mező már rendelkezik az előre beállított értékkel. Ha a módosíthatóság nincs bepipálva, akkor a mező értékét nem lehet módosítani, értéke vagy az előre beállított marad vagy null lesz. A rejtett opció kiválasztásakor a mező bár létezik az űrlapon, de nem jelenik meg, ezáltal nem is lehet közvetlenül módosítani az értékét.

A beviteli mezők másik bemutatásra kerülő fajtája a legördülő lista, vagy ahogy a WebRatio hívja selection field. A legördülő listához egy érték/kimenet (value/output) halmazt lehet hozzárendelni, és az űrlapon ezek közül lehet majd választani. A hozzárendelés történhet reteszek (slot) megadásával, de én az alkalmazásban bemenő paraméterként adtam meg egy válogató egység segítségével. A legördülő lista három értéket kér, címke (label), ami



megjelenik a legördülő listában, kimenet (output), ami a kiválasztott címkéhez tartozó érték, és előválasztás (preselection), ekkor betöltődéskor már ki van választva az a listaelem, feltéve, ha létezik.

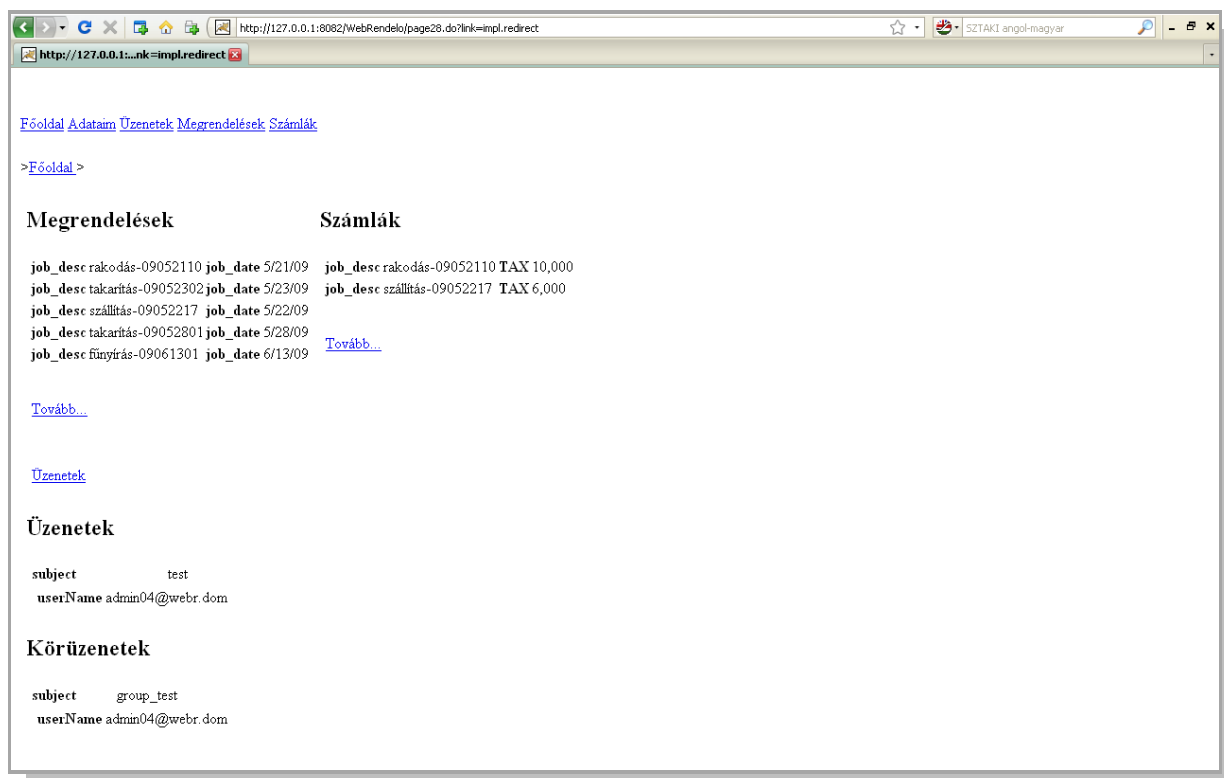
Minden mezőhöz lehet érvényesítési szabályt rendelni, az ábra erre is mutat néhány példát. Rögtön ott van az „email_format” nevezetű érvényesítési szabály, aminek látszanak a beállításai is a properties ablakban. Meg kell adni egy predikátumot (predicate), egy összehasonlítási értéket (value), illetve lehet saját hibaüzenetet is megadni. A bemutatott esetben pont egy olyan predikátum van kiválasztva, amelyhez nem tartozik összehasonlítási érték. Egyébként ez a predikátum igazat ad, ha a mezőben szereplő érték szabályos e-mail formátumban van (user@provider.domain).

Szintén a 8. ábrán láthatók olyan mezők, amiknek a neve „h_”-al kezdődik. Ezek lesznek a rejtett mezők, amik nem jelennek meg az űrlapon. Gyakorlatilag változók kiváltására használható.

8. ábra Egy űrlap körvonala (Outline)

2.5.4 Lekérdezések, listák

Alapvetően minden, index egység, válogatók, feltételek, szóval az egyedekkel dolgozó egységek háttérében egyszerű SQL-szerű (HQL vagy OQL) lekérdezések működnek. A 9. ábrán egy a megrendelők nézetéből kiragadott képernyőkép látható. Ez a főoldala az oldal nézetnek, ahogy belép a felhasználó rögtön az aktuális megrendeléseit, teljesítetlen számláit és neki illetve csoportjának érkezett üzeneteket.



9. ábra a Megrendelő kezdőlapja

Az ábrán látható listák mindegyike hierarchikus index egység, azért ez, mert nem egy táblába kerültek az egyazon adathalmazhoz tartozó részek. Például az üzenetek táblában csak a küldő azonosítója van tárolva, viszont itt megjelenítem a küldő nevét is, ami a hierarchikus egységen definiált szinttel (level) lehetséges, természetesen a kettőt összekötő kapcsolat létezik, mind adatmodell mind pedig adatbázis szinten.

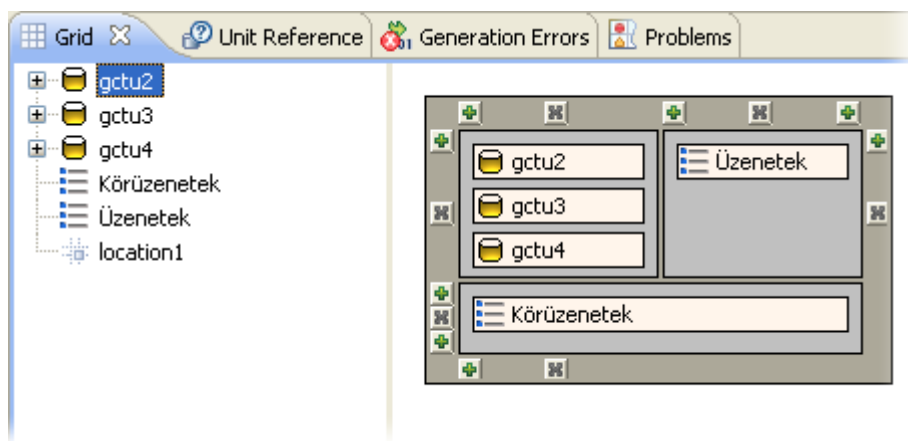
2.5.5 Saját üzleti logika hozzáadása

A lekérdezés alapú unitoknál lehetőség van úgynevezett Custom Descriptor definiálására, ami a „kattintgató” felületen túli lehetőségeket teszi elérhető, többek között összetett OQL lekérdezések írására is van lehetőségünk.

Másik módszer, amikor egy teljesen új egységet hozunk létre. Ilyenkor *Web Project* helyett egy új *Units Project*et kell létrehozni, majd a *Units Project*-en belül létre lehet hozni a saját egységet. Ezután az azonosító rövidítés megadása, amihez majd társul a számláló, valamint az alapértelmezett elrendezés. Ezután Java-ban kell megírni pontosabban kiegészíteni az új egységet, a „setter” metódusok a bemenő, a „getter” pedig a kimenő paraméterek lesznek. A dolgozat korlátai miatt ezt a témát részletesebben nem érintem, további információ itt található: http://wiki.webratio.com/index.php/Developing_Custom_Units.

2.5.6 Megjelenítés és formázás, rács nézet (GRID view)

Található a WebRatio-ban egy olyan rács (grid) nevezetű szerkesztőnézet, ami egy a „*Window/Show View/Others.../WebRatio/Grid*” menüpontból elérhető ablak. Itt a prezentációs modell felvázolására vagy akár teljes megvalósítására is lehetőség van. A Rács ablak bal oldalán egy lista, jobb oldalán pedig egy táblázat jelenik meg.



10. ábra Rács nézet

A lista tartalmazza az egy oldalhoz tartozó megjeleníthető vagy megjelenítést végző egységeket. A jobb oldalon lévő táblázatban pedig ezen egységek elrendezése szabható meg. Ahogyan a 10. ábrán is látszik, a táblázat cellái egyesíthetők illetve egy cellába több egység is rakható. Bár a 10. ábrán erre nincs példa, a cellákba újabb táblázatok szúrhatók

be, továbbá egy egység akár többször is megjelenhet. Az egységeket „fogd és vidd” (drag and drop) módszerrel lehet pakolgatni, áthúzni egyik cellából a másikba vagy a listából behúzni, ha automatikusan nem került bele.

Amikor ezt a nézetet használjuk a *Properties* ablak átváltozik, pontosabban ugyanahhoz az egységhez más, a megjelenítéssel és elrendezéssel kapcsolatos beállítások válnak elérhetővé. A WebRatio tartalmaz néhány beépített stílust, valamint továbbiak is definiálhatók mint Stílus Projekt (Style project). A Stílus Projekt sablonokat és erőforrás fájlokat tartalmaz, ami az összetevők megjelenését szabályozza. A dolgozat időbeli és terjedelmi korlátjai miatt ebbe a témába részletesebben nem mennék bele. További információk a <http://wiki.webratio.com/index.php/Presentation> lehet találni.

3. Összegzés

3.1 Az MDD/WebML hasznossága

A dolgozat elején ugyebár a modell alapú fejlesztőeszközökről azt írtam, hogy gyorsabb, produktívabb fejlesztés valósítható meg a használatukkal. Az igaz, hogy a módszertan alapos megismerése után nagyságrendekkel gyorsabban meg a fejlesztési folyamat, mégis igazi innováció szoftveresen nem megvalósítható, az emberi tényező kiküszöbölhetetlen, konstruktív gondolkodásra kétségtelenül szükség van. Mindazonáltal a jó problémamegoldó képességen túl nem árt a kapcsolódó technológiák több mint felületes ismerete, érteni kell, hogy mi folyhat éppen a háttérben és miért, mert különben az egyszerűnek indult fejlesztés végeláthatatlan hibakeresésbe csaphat át.

3.2 A módszertan tanulhatósága

Alapvetően maga a modellezési módszertan szinte autodidakta módon tanulható, a kevés, de jól körülhatárolt és sok problémakörre alkalmazható eszközrendszernek köszönhetően. Úgy érzem a módszertan még nem egészen kiforrott, sok helyen jól paraméterezhető egységek vannak, de néhány helyen még, ahol igazán szüksége lenne a rugalmasságra, csak a rendszert/módszertant alaposan ismerő fejlesztő rúghat labdába. Nem igazán tudom, mit ajánlanék jobban egy a módszerrel most ismerkedőnek, hogy először kezdjen el a gyakorló (tutorial) feladatokkal foglalkozni, vagy olvassassa kicsit a referenciákat. Saját szemszögemből nézve, gyakorlati oldalról megközelítve szerintem hamarabb megérthető maga a fejlesztés menete és pár alapelem, de nem árt minden elem felhasználása előtt a referenciába vagy valamelyik leírásba is beleolvasni.

3.3 A fejlesztés hatékonysága

Amint azt korábban írtam a fejlesztés hatékonyságával nincs semmi problémám, kellő tanulás és gyakorlás után órák alatt összeépíthető egy-egy méretben hasonló modell mint ami a dolgozathoz készült. Nem utolsó szempont a fejlesztőeszköz megjelenése és ergonómiája. A megjelenés modern, az szokásos menüpontokat a szokásos helyen kell keresni, semmilyen kirívó menüpont elhelyezést sem találtam. Maga a környezet kicsit csúfolt, 1200x800-as

felbontás mellett nem sok hely maradt, ha minden éppen szükséges ablakot megjelenítettem. Talán ami még hiányzott, hogy az egységek tulajdonságait akár egy felugró ablakban is lehetett volna kezelni.

Többször is meggyúlt a bajom az adatbázis szinkronizálással, akár néhány apró módosítás is tönkretelheti az adatmodellt. Van viszont egy hasznos funkciója a szinkronizálásnak, ugyanis az ilyenkor előálló SQL szkriptet képes automatikusan menteni.

A beágyazott Tomcat alkalmazáserver sem bírta annyira a strapát, mint ahogy a fejlesztő oldalán olvasható, néhány tíz projekt újragenerálás után hajlamos kivételt dobni, és ilyenkor már csak az újraindítás segít rajta. Szintén ide kapcsolható, hogy nem minden esetben írónak felül korrektül a szükséges részek, az egyéb fejlesztő környezetekben megszokott „Clean” menüpontot nem találtam, ami törölné az alkalmazás fájljait a szerverről, majd újra létrehozná őket. Szintén fontos szempont az óriási memóriaigény, gyakran előfordult ugyanis, hogy egy 8-10 oldalból álló, több órája fejlesztett modellnél simán elfogyott 1GB ram.

3.4 A végeredményként előállt szoftver értékelése

A végeredményként kapott szoftver megfelel a céloknak, azaz prezentációs célokra tökéletesen alkalmas. Mindezekén túl, egy „éles” projekt fejlesztése esetén, úgy látom kiváló alapot nyújt, a kifejlesztendő szoftver vázának vagy első részleges prototípusának kifejlesztésére. Végül is arra jutottam, hogy bárkinek ajánlható ez a fejlesztőeszköz, aki jártas az eddig említett technológiákban, illetve gyorsan működőképes, bemutatható prototípushoz akar jutni, amire lehet építkezni.

IRODALOMJEGYZÉK

Könyv:

[1] *I. Sommerville*, Szoftverrendszerek fejlesztése, Második, bővített, átdolgozott kiadás, Panem, 2007.

PhD-dolgozat:

[2] *Adamkó A.*, Web alapú Információs Rendszerek modellezése, PhD-dolgozat, Debreceni Egyetem, 2008.

Elektronikus cikkek:

[3] *S. Ceri, P. Fraternali, A. Bongio*, Web Modeling Language (WebML): a modeling language for designing Web sites, Proc. of the 9th Int. World Wide Web Conference, 137-157, 2000.

[4] *M. Brambilla, S. Comai, P. Fraternali, M. Matera*, Designing Web Applications With WebML And WebRatio, in book: G. Rossi, O. Pastor, D. Schwabe, L. Olsina (eds), Web Engineering: Modelling and Implementing Web Applications, Human-Computer Interaction Series, Springer, 2008.

[5] *M. Brambilla, S. Ceri, P. Fraternali, I. Manolescu*, Process Modeling in Web Applications, ACM Trans. Softw. Eng. Methodol. 15 (4), 360-409, 2006.

Internetes kutatás:

[6] *WebRatio Wiki*,
http://wiki.webratio.com/index.php/Main_Page

[7] *5 type of Modell Driven Software Development*

<http://www.theenterpriseearchitect.eu/archive/2009/03/31/5-types-of-model-driven-software-development>

[8] *Apache Tomcat – Overview*

<http://tomcat.apache.org/>

[9] *Apache Struts*

http://en.wikipedia.org/wiki/Apache_Struts

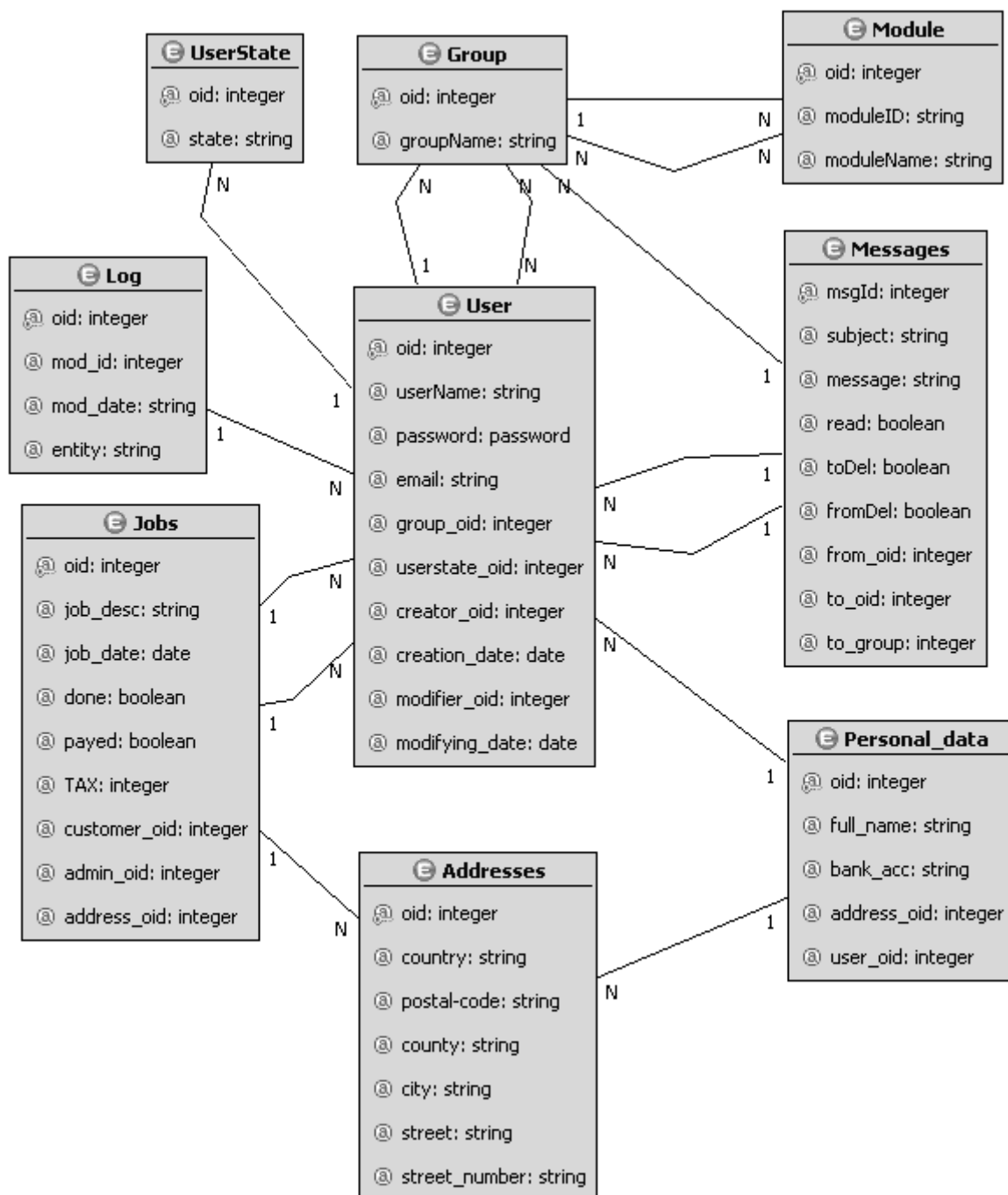
[10] *Hibernate*

<http://www.hibernate.org>

[11] *Modell Driven engineering*

http://www.theenterpriseearchitect.eu/archive/2008/03/14/model_driven_engineering

FÜGGELÉK



I.ábra A teljes adatmodell