Web alapú Információs Rendszerek modellezése

Modeling Web-based Information Systems

Doktori (PhD) értekezés tézisei

Adamkó Attila

Debreceni Egyetem
Természettudományi Doktori Tanács
Matematika és Számítástudományok Doktori Iskola

Debrecen, 2008.

Tartalom

1. Bevezetés

Az Internet rövid idő alatt nagyon széles körben terjedt el, bekapcsolódott az élet számos területére, egy új irányvonalat hozott létre az alkalmazásfejlesztésben. A webtechológiák fejlődésével a mindennapi életünkben egyre nagyobb szerepet tölt be a web, kezdve a személyes honlapoktól a webáruházakon és vállalati portálokon át a komplex üzleti folyamatokkal rendelkező webalkalmazásokig. A webalkalmazások sokszínűsége megnehezíti az együttműködésükhöz szükséges technológiák és platformok kiválasztását.

Az igen gyors ütemben fejlődő technológiák hamar elavulttá válhatnak. Megjósolhatatlan, melyik marad fenn vagy kerül előtérbe. Annak érdekében, hogy a technológiaváltás ne befolyásolja az üzleti szabályokat és folyamatokat fontos, hogy az üzleti logikát a lehető legnagyobb mértékben függetlenítsük a technológiától.

Ezen rendszerek *modellezése igen összetett feladat*, mert számításba kell vennünk a létező és a várható technológiákat, hogy egy rugalmas és bővíthető rendszert alakíthassunk ki. Annak ellenére, hogy a technológia változik, az alkalmazás szakterületi problémái változatlanok maradnak, mert ezek az üzleti modelleket és folyamatokat rögzítik. A gyakorlat azt igazolja, hogy az alkalmazás logikáját a technológiai részletektől függetlenül célszerű kialakítani.

1.1. Web alapú Információs Rendszerek

A kutatásaink során a fent említett webalkalmazásoknak egy kicsi, de fontos csoportját elemeztük: a Web alapú Információs Rendszereket (Web Information System – WIS). „*Egy WIS olyan információs rendszer, amely a weben keresztül biztosítja az interaktív szolgáltatásait és a komplex adatok elérhetőségét*” [1]. Ezen rendszerek „*az információs rendszerek egy olyan meghatározó alosztályát alkotják, amelyek a szolgáltatásaik révén tipikusan az online információelérést és a napi feladatok ellátását támogatják igen nagyszámú (ezres vagy milliós) felhasználói körnek, akik távoli helyeken vannak*” [2].

A következő táblázat ezen rendszerek egy lehetséges osztályozását mutatja be az információ jellegének és a kommunikáció irányának függvényében:

	Aszimmetrikus kommunikáció	Szimmetrikus kommunikáció
Objektív információ	Információszolgáltató	Információs rendszer
Befolyásoló információ	Hirdetési	Közösségi

1. táblázat: A WIS négy perspektívája

Az *információszolgáltatók* esetében a rendszertől a felhasználók felé történő egyirányú kommunikációról beszélhetünk. Tipikus példái ennek a hírportálok. A *hirdetési* rendszerek ezzel szemben tekintetőek az üzleti kommunikáció csatornájának is, ahol a cél a promóciós üzenetek eljuttatása a potenciális vásárlókhoz weboldalakon keresztül. Ezen felül szolgálhatnak egy virtuális közösség kialakítására is, amelyre a fórumok nyújtanak jó példát. A negyedik perspektíva azonban úgy is ismert, mint *adatorientált webalkalmazások*. Tipikusan az olyan nyilvántartási rendszerek tartoznak ide, mint például egy online repülőjegy foglalási vagy egy könyvtári nyilvántartási rendszer. E rendszereknél középpontjában maga az adatstruktúra, illetve a felhasználó és a rendszer közötti információáramlás áll. Összetett üzleti folyamatok segítségével támogatják a felhasználóik munkáját, és ebben az esetben a kommunikáció két irányú.

Ilyen rendszerek ***kidolgozása kimondottan összetett feladat***. A fejlesztők annak érdekében, hogy időt takarítsanak meg, többnyire mellőzik a szisztematikus módszerek alkalmazását. Ennek eredményeképpen általában hibás (helytelen) tervek és egyedi (nem újrafelhasználó) modellezési praktikák alakulnak ki. Ezen túlmenően a folyamatosan módosuló igényekre adott gyors reagálást szinte lehetetlen időben megvalósítani, mert az új igények megjelenése a rendszer egyes részeinek vagy akár magának a teljes rendszernek az újratervezését is megkövetelhetik. Az újraimplementálás azonban nagyon időigényes munka, amelynek az fő oka, hogy a rendszer modellje és annak implementálása közötti hiányzik a kapcsolat. A hagyományos szoftverfejlesztési módszerek alkalmazása esetében is gyakori, hogy miután a modell elkészült és első körben implementálásra került, a további lépésekben már nem használják, nem frissítik, egyszerűen „eldobják". Így nem alakul ki kapcsolat a modell és a kód között, ezért a modellben történő változás nem látszik a kódban, és ez fordítva is igaz, a kódban történő változások nem kerülnek átvezetésre a modellben. Emiatt a modell nem tudja ellátni azt a feladatát, amire elsődleges céllal létrejött, nevezetesen, hogy a rendszerről meghatározó információkat szolgáltasson.

2. Motiváció

2.1. A kiindulási rendszer

Pár évvel ezelőtt a Debreceni Egyetem Matematikai és Számítástudományok Doktori Iskolája elhatározta, hogy a nyilvános adatok weben keresztül történő elérhetőségét megvalósítja. A kialakítandó webalkalmazással szemben támasztott követelmények között szerepelt a Doktori Iskola oktatóinak és hallgatóinak nyilvántartási rendszerbe vétele, az adatváltozás nyomon követése, illetve a doktori programok és kurzusok menedzselhetősége. Mindamellett, hogy lehetővé vált az adatok rendszerezett formában történő elérése, különböző statisztikai kimutatások készítésére is lehetőség nyílt. A hallgatók és az oktatók számára rendelkezésre állt a publikációk egyszerűsített kezelése is. A fentebb szereplő táblázat alapján ezt a rendszert az információs rendszerek közé sorolhatjuk, hiszen interaktívan használható a különböző feladatok ellátására, melyek magukban foglalják az adatfeltöltést, a módosítást és a lekérdezéseket.

A rendszer alapját egy PostgreSQL adatbáziskezelő rendszer szolgáltatta, míg az elérhetőségét a HTML oldalakba ágyazott Perl nyelven íródott CGI szkriptek biztosították. Az alkalmazásréteg több rétegből állt, kihasználva a PL/pgSQL tárolt eljárásokat az adatbáziskezelőn keresztül. Mára ezen technológiák többé-kevésbé elavultak, mert nem támogatják az adat szerkezetének és megjelenítésének a magas szinten történő kettéválasztását.

2.2. Problémák

A rendszer üzemeltetése során új követelmények fogalmazódtak meg. Ezek egy része könnyen implementálható volt, viszont megjelent olyan követelmény is, amely a teljes adatstruktúra újratervezését igényelte volna. Ezek többsége a helyi szabályozások változására és a szervezeti átalakulásra vezethető vissza. Az adatmodell változásával azonban további problémák is megjelentek, amelyek a rendszer többi részére is kihatottak. Így világossá vált, hogy a rendszer karbantartása és további fejlesztése nagyon összetett és nehezen megvalósítható. Ezért megfogalmazódott a teljes rendszer újragondolása.

Gyakori eset, hogy a problémák jelentős része a követelmények fejlődésével (evolúciójával) jelenik meg. A nehézségek már a karbantartásnál jelentkezhetnek, mert a fiatal fejlesztők nem feltétlenül rendelkeznek a régebbi technológiákhoz szükséges ismeretekkel, de ha mégis, akkor a változtatások átvezetése könnyedén vezethet nem megfelelő tervezési és/vagy kódolási stratégiához. Mindezek alapján döntöttünk úgy, hogy egy modell alapú tervezési stratégiát alkalmazunk a fejlesztési idő és a karbantartási költségek csökkentése érdekében.

3. Elméleti alapok

3.1. Tartomány alapú tervezés

A tartomány alapú tervezés (*Domain Driven Design*) [3] a nagy összetettségű szakterületek vizsgálatát támogatja, magának a szakterületnek a projekt középpontjába állításával. Ez egy olyan szoftvermodell elkészítését és karbantartását igényli, amely a szakterületről kialakított átfogó ismereteinket tükrözi. Már kezdetben fontos átlátni annak jelentőségét, hogy a szoftver az adott területhez (az adott környezethez) nagyon szorosan kapcsolódik. A tartomány alapú tervezés segít a probléma megértésében és középpontba helyezésében, támogatja egy olyan szakterületi modell elkészítését, amely az alapvető koncepciókat tartalmazza. A tartományra vonatkozó ismereteink kialakítása során lehetővé válik a fontos információk kinyerése és általánosítása.

A tartomány alapú tervezés magának a szakterületnek a meghatározásához a következő építőelemeket használja: ***entitás, értékobjektum, szolgáltatás és modul***. A *szolgáltatások* biztosítják a felületet a tevékenységek végrehajtásához és céljuk a tartomány funkcionalitásának biztosítása. A *modulok* segítségével az összetartozó részeket foghatjuk egybe. Ezenfelül megjelennek még további jól ismert tervezési minták is, mint például az *aggregátor*, az *építő*, a *gyár* és a *tároló* annak érdekében, hogy a komplexitás csökkenjen. Az aggregátor egy olyan szerkezeti minta, amely az objektumok szoros összekapcsolódásának és határainak definiálását segíti. A gyár és a tároló minták pedig az objektumok létrehozását és elhelyezését támogatják.

3.2. Tartomány specifikus nyelvek

A tartomány specifikus nyelvek (*Domain Specific Languages – DSL*) [4] olyan nyelvek, amelyeket kimondottan egy adott problématerület leírására dolgoztak ki. Számos példát ismerünk tartomány specifikus nyelvekre a HTML-től kezdve az SQL-en át a UNIX belső programnyelveiig. Ami alapvetően új ötlet a tartomány specifikus nyelvek esetén, az az, hogy magunk készíthetjük el a saját speciális nyelvünket a saját projektünkhöz. A tartomány specifikus nyelvek számos alapvető különbséget mutatnak az általános célú nyelvekkel szemben:

- kevésbé átfogóak,
- sokkal nagyobb kifejező erővel rendelkeznek a saját problématerületükön,
- nagyon kevés felesleges elemet tartalmaznak.

A modellvezérelt fejlesztés során számos tartomány specifikus nyelvvel találkozhatunk. Ilyen többek között az OCL (*Object Constraint Language*), amely a modellek megszorításainak leírására szolgál vagy a QVT (*Query-View-Transform*), amely a modellek transzformációjához nyújt támogatást. Mindezekkel szemben, a modellek elkészítéséhez használt UML (*Unified Modeling Language*) nyelv tipikusan az általános célú nyelvek közé tartozik.

3.3. Modell vezérelt architektúra

Az OMG (*Object Management Group*) által kidolgozott modell vezérelt architektúra (*Model Driven Architecture – MDA*) [5] egy olyan keretrendszert biztosít, amelynek segítségével az alkalmazásainkat platform független formában adhatjuk meg. Az MDA-ban a termékek formális modellként jelennek meg, az adott rendszert különböző szemszögekből bemutatva. Az MDA az UML nyelvet, mint szabványos modellező nyelvet használja annak érdekében, hogy a résztvevők között egy mindenki által ismert felületet nyújtson.

A modell vezérelt fejlesztés során a platform független modellek (*Platform Independent Model – PIM*) készülnek el először. Ezen modellek magas fokú absztrakcióval rendelkeznek, miután az implementációs részletek – adatbáziskezelő rendszer, alkalmazás szerver platformja – rejtve maradnak. A PIM modellek mutatják be, hogy a rendszer miként fogja támogatni az „üzletet". Az elkészítésük után második lépésben a PIM modelleket transzformáljuk egy vagy több olyan modellé, amely már az egyes platformoktól függ. Ezeket nevezzük platform specifikus modelleknek (*Platform Specific Model – PSM*).

A transzformációs folyamatok feladata, hogy az egyes technológiai részleteket számításba vegye. Miután a mai rendszerek számos technológián alapulnak, minden egyes technológiához készül egy PSM. A fejlesztési folyamat utolsó lépése az egyes PSM modellek futtatható kód formájára történő transzformációja. A PSM modell nagyon szorosan kapcsolódik a technológiához, ezért viszonylag könnyen alakítható át kóddá.

Ezzel a módszerrel biztosítható, hogy egy technológiaváltáskor (vagy evolúciókor) a PIM modellekben ne kelljen változtatásokat végezni, hiszen az alkalmazást platform független módon jelenítjük meg. Mindössze annyit kell tennünk, hogy egy új platform specifikus modellt készítünk, amelyben leírjuk a platform független részek megjelenési formáját az új platformon, illetve az új technológiában. A platform specifikus modellek könnyebb kialakításának érdekében célszerű egy platform specifikus metamodellt meghatározni, amely az új platformot képes absztrakt módon leírni, ezáltal megkönnyíthetjük a platform specifikus modellek létrehozását. Természetesen ehhez még szükséges megadni azon transzformációs szabályokat is, amelyek a PIM metamodell elemeket leképzik a PSM metamodell elemeire. Ezen szabályoknak egy PIM modellre történő alkalmazásával előállíthatjuk az új platform PSM modelljét.

3.4. UML, mint modellező nyelv

Az MDA módszertan előírja egy olyan nyelv használatát, amely formális definíciókat használ annak érdekében, hogy az eszközök képesek legyenek a modellek automatikus transzformációjára. Az OMG az UML nyelvet ajánlja a platform független modellek létrehozásához. Esetünkben az UML ereje abban rejlik, hogy a rendszer strukturális szempontjait az alapeszközökkel könnyen modellezhetjük. Ehhez csupán az osztálydiagramokat használjuk, amelyek biztosítani tudják a platform specifikus modellek generálását a szerkezeti jellemzőkkel együtt.

3.5. UML profilok

A különböző szakterületekhez tartozó UML modellek létrehozásának a legáltalánosabb módja az UML szemantikájának a kibővítése. Ehhez használhatunk UML profilokat, amelyek sztereotípiákat és kulcsszavas értékeket adnak a modellekhez. Ezen profilokra úgy is tekinthetünk, mint amelyek a metamodellben helyezkednek el, különféle UML „*dialektusokat*" meghatározva, amelyeket a modelljeinkben felhasználhatunk. Ez a mechanizmus mind a platform független, mind a platform specifikus modellek elkészítéséhez hasznos segítséget nyújt.

Az egyes módszertanok esetében a PIM modellek létrehozásához új UML profilok kialakítása szükséges. Ilyen profilokat találunk a [6] [7] és [1] cikkekben, bár ezek alkalmazhatósága nem feltétlenül illeszkedik bármely tervezési stratégiához. Természetesen léteznek kidolgozott UML profilok a specifikus platformokhoz is, mint például az EJB (Enterprise Java Beans) vagy a CORBA, hogy néhányat említsünk. A nyelv bővíthetőségének köszönhetően új profilok is kialakíthatóak lehetővé téve az új platformok és technológiák alkalmazását a modellekben.

Ezek alapján a PIM (forrás)modell PSM(cél)modellé történő transzformációjához a metamodellben található koncepciók transzformálására van szükség.

4. Új eredmények

A dolgozatban a webalkalmazások fejlesztéséhez kínálunk ***egy új, szisztematikus tervezési módszertant***, amely ezen alkalmazásokat *adatorientált szempontból* vizsgálja. Bemutatunk irányelveket, amelyek az adott szakterülethez kapcsolódó tartomány alapú modellek fejlesztését segítik. A módszerünk az *UML metamodelljének kiterjesztésén* alapul, amelyhez felhasználtuk a tartomány alapú tervezés irányelveit.

A szakterület leírásához ***elkészítettünk egy új tartomány specifikus nyelvet*** (DSL), amely bevezeti a ***szerepkörök*** *fogalmát*. Fontosnak tartojuk már a szakterület feltérképezésekor kiemelni az egyes *entitások különböző szerepkörökben történő viselkedésének* vizsgálatát. A nyelvünk segítségével meghatározott modellelemek kerültek leképzésre a módszertanunk által kínált ***szerkezeti modell*** metamodelljére, majd ebből *származtatunk egy UML profilt*.

Az általunk ajánlott fejlesztési folyamat második lépése a ***navigációs modell*** elkészítése. Kidolgoztunk egy módszert, ami lehetővé teszi a webalkalmazás navigációs modelljének *származtatását a szerkezeti modellből*. Ezt követően, az UML2 újdonságait kihasználva, a navigációs elemekből ***komponenseket alakítunk ki*** annak érdekében, hogy koncepcionális információszolgáltató elemek jöjjenek létre. Ezen elemek interfészek segítségével kapcsolhatóak egymáshoz, amely kapcsolatok a szükséges információt tartalmazó *absztrakt oldalak* kialakítását teszik lehetővé. A prezentációs réteg ezen absztrakt oldalakat alakítja át a kliensek számára megfelelő formátumra, így biztosítva a különböző eszközökön át történő hozzáférést.

Az MDE fejlesztési irányzatban rejlő lehetőségek kiaknázásához ***UML és XML technológiákat alkalmazunk*** annak érdekében, hogy a modelljeinkből futtatható kódot állítsunk elő. A szabványok (*OMG XMI formátum*) és ajánlások (*W3C XML Schema, XForms*) használatával ezenfelül ***lehetőség nyílik az adatmanipulációs oldalak automatikus generálására*** is. Így a módszerünk hatékony támogatást biztosít az adatorientált webalkalmazások fejlesztéséhez.

4.1. A saját DSL létrehozása

A tartomány alapú tervezés irányelveit követve ***kialakítottuk a saját tartomány specifikus nyelvünket*** az *adatorientált* webalkalmazások számára. A nyelvünk előnye, hogy a szakterület modellezésénél figyelembe veszi a *szerepkörök modellezését*. A szakterületi modell elkészítésénél fontos, hogy az entitásoknál meg tudjuk jeleníteni a szerepekből adódó eltérő viselkedést. Ezen jellegzetességek figyelembevételével lehetőségünk nyílik arra, hogy a fejlesztési folyamat tervezési fázisát előkészítsük a modellezési elemek megfelelő használatára. A szerepkörök használatához azonban *az entitás definícióját ki kellett bővítenünk*, ami ezen felül megköveteli az egyes szerepköröknek megfelelő *szolgáltatások kialakítását* is.

A szolgáltatás és a modul definícióját változatlanul hagytuk, bár a modulokra úgy tekintünk, mint az összetartozó koncepciók határára. Ez a szerepkörök bevezetésénél azt eredményezi, hogy az ilyen entitásokat az egyes szerepkörökhöz szükséges szolgáltatásokkal együtt egy modulba célszerű összefogni.

4.2. Az UML metamodell kiterjesztése

A metamodell az egyes modellezési lépésekben használt koncepciók pontos definícióját adja meg elemek, kapcsolatok és szabályok formájában. A célunk az ***UML metamodelljének a kiterjesztése*** a szakterületi szempontok figyelembevételével. Ehhez az UML metamodell elemeiből *saját osztályokat és asszociációkat származtattunk*. Az egyes szerkezeti elemek közötti kapcsolat kialakításánál figyelembe vettük a „szabály objektum" (*Role Object*) tervezési mintát. A navigációs metamodell esetében pedig az UWE módszertan által bevezetett modellezési elemeket használtuk. Annak érdekében, hogy a modellezési eszközök támogatását ki tudjuk használni, ***elkészítettük a metamodelleknek megfelelő UML profilokat***. Ezen profilok tartalmazzák a megfelelő sztereotípiákat, kulcsszavas értékeket és szabályok definícióját, a modellező eszközök pedig ellenőrzik, hogy az elkészült modellek megfelelnek-e az adott profilnak.

4.3. Fejlesztési folyamat

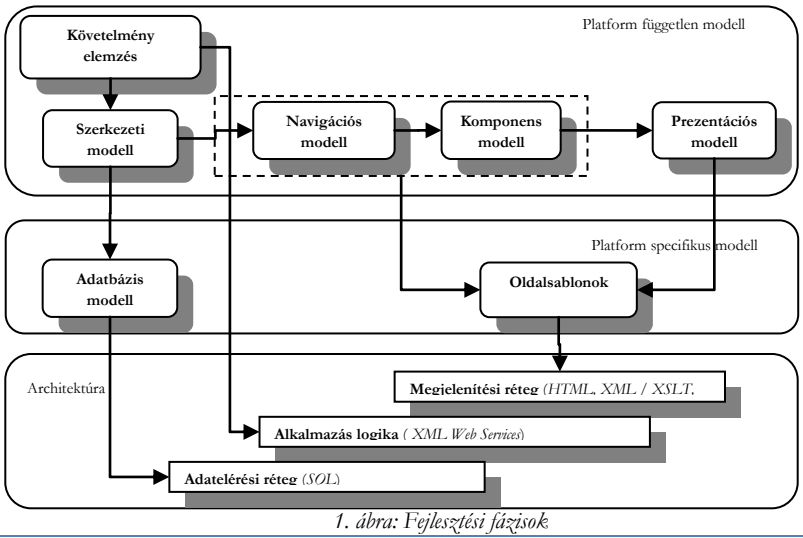
A rendszer megtervezésének első lépése a követelmények elemzése és a felhasználói kérések formalizálása. A használati esetek és aktivitás diagramok alkalmazásával meghatározhatjuk a rendszer körvonalát és a különböző felhasználói nézőpontokból fejezhetjük ki a rendszer alapvető funkcionalitását.

A webalkalmazás felhasználóinak eltérő szerepeit az aktorok reprezentálják, míg az általuk végezhető tevékenységeknek megfelelő műveletek leírásához *használati eseteket készítünk*. Jelentőségük abban rejlik, hogy felhasználhatóak az egyes felhasználói csoportokhoz kapcsolódó nyitóoldal kialakításához, amely az adott csoportba tartozó felhasználók esetén az általuk végezhető tevékenységek listáját tartalmazza.

Az *alkalmazás logika leírásához tevékenységdiagramokat alkalmazunk*, melyek alapját a használati esetek nyújtják. Segítségükkel előállíthatunk kódrészleteket és programmodulokat. Természetesen a mi esetünkben ez valamennyit egyszerűsödik, hiszen ezek szorosan kötődnek az adatkezelő és manipuláló feladatokhoz.

Második lépésben az alkalmazás adatszerkezetének és hozzáférési útjainak a koncepcionális modellezése történik. Ehhez az UML2 osztálydiagramját használjuk fel, melyben alkalmazzuk a korábban előállított profiljainkat. A ***koncepcionális szinten szerkezeti, navigációs, komponens és megjelenítési modellek készülnek***. Az irodalomban található módszertanok többé-kevésbé ezzel megegyező utakat használnak, ilyen például az OO-HDM, WebML, UWE és a WAE.

Miután elkészültünk a platform független modellel egy modelltranszformáció segítségével előállíthatjuk a platform specifikus modelljeinket, amelyek már felhasználhatóak a kódgeneráláshoz. A ***módszerünk az alábbi ábrán látható***.



1. ábra: Fejlesztési fázisok

4.4. A tervezési fázis modelljei

4.4.1 Szerkezeti modell

A szerkezeti modell legfontosabb építőelemei az osztályok, az asszociációk és a csomagok. Webalkalmazások esetén a szakterület koncepcionális tervének kialakításához a használati esetek és az aktivitás diagramok szolgáltatják az alapot.

Az általunk vizsgált adatorientált webalkalmazásoknál a szakterület szerkezeti modellje azonban összetettebb, mint bármely eddigi diagram. Nagyon fontos ezen szerkezeti modell megfelelő „minőségű" kialakítása, mert számos további lépés (mint például az egyszerűbb navigációs elemek automatikus előállítása az asszociációk mentén) ezen alapul.

4.4.1.1 Szerepkörök modellezése

Az általunk bemutatott megközelítés további ***előnye, hogy a szerkezeti modellben figyelembe veszi a szerepkörök kialakításának lehetőségét***. Számos olyan helyzet alakulhat ki, amikor egy objektumnak attól függően kell különböző szolgáltatásokat nyújtania (adat, viselkedés), hogy épp milyen kontextusban történik az üzenetváltás. Gondoljunk például egy egyetemi nyilvántartó rendszerre, amelynek feladata az oktatók és hallgatók szervezeten belüli előrehaladásának a figyelmemmel kísérése. A rendszer működésében egy személy szerepelhet bizonyos esetekben oktatóként, míg más esetekben hallgatóként.

Ezek alapján a szerepkörök az egyes objektumok azon jellemzőinek a halmaza, amelyek ahhoz szükségesek, hogy egy adott kontextusban működni tudjon. A modellezésben ennek leírására a *Role Object* tervezési mintát alkalmaztuk.

4.4.2 Navigációs modell

A fejlesztési folyamat következő lépése ***a navigáció megtervezése***. A kialakítandó navigációs modell fogja meghatározni az alkalmazás egyes pontjainál a szerkezeti modell elérhető elemeit. A navigációs modell tervezése során a tervezőnek döntő fontosságú lépéseket kell tennie annak érdekében, hogy kialakítsa az alkalmazás működéséhez, funkcionalitásához szükséges navigációs utakat. A döntések a szerkezeti modellen, a használati eseteken és az alkalmazás által teljesítendő navigációs követelményeken alapulnak.

Esetünkben a navigációs diagram erősen kötődik a szerkezeti modellhez, miután a szerkezeti elemek közötti navigációs utakat határozza meg. Mindazonáltal új asszociációk is felvehetőek a közvetlen navigáció érdekében, ezzel biztosítva az egynél hosszabb navigációs utak elkerülését. Természetesen létezhetnek olyan osztályok is a szerkezeti modellben, amelyeket nem érintenek a navigációs utak. Ezeket nem ábrázoljuk a diagramon. Az alkalmazáson belüli navigáció többnyire az asszociációk mentén történik, amelyek a szerkezeti elemek közötti összefüggéseket írják le. Tipikusan ezek az asszociációk fognak megjelenni a felhasználói felületen hivatkozásként (link) vagy menüpontként.

A navigációs diagram a szerkezeti modell előbb említett bővítései mellett tartalmazni fog még további osztályokat és asszociációkat is, amelyek a különböző elérési szerkezeteket fogják reprezentálni, mint például *menü, lekérdezés és index*.

4.4.3 Komponens modell

A fejlesztési lépések közé ***bevezettük a komponensmodellt***, amely feladata ***a rendszer alapvető moduljainak a meghatározása***. A tartomány specifikus nyelvünk *modul definíciója* által meghatározott koncepciókat jelenítik meg ezen modulok. A modul definíció tartalmazhat szolgáltatás elérési pontokat is, amelyekhez szükséges a megfelelő interfészek kialakítása. Ezenfelül a modult használjuk az entitásokból származtatott navigációs osztályok és a hozzájuk kapcsolódó navigációs szerkezetek összefogására is. Ebből következik, hogy a modulok a navigációhoz szükséges felületet is biztosítani fogják. A megfelelő interfésze kiválasztása mindig a navigációs kontextus alapján történik. Természetesen a navigáció során is kialakulhatnak újabb szerepkörök, amelyeket a modulok az egyes viselkedésekhez szükséges újabb interfészek kialakításával biztosítanak.

A komponensek fogják szolgáltatni a prezentációs modell számára a szerkezeti modellben megjelenő koncepciók leképzését weboldalakra. Ezen komponenseket úgy kell elképzelni, mint egy objektumot, amely kapcsolatban áll a szerkezeti modell entitásaival. Ezáltal egy weboldalon található komponensek számos entitásból származó információt tudnak összefogni, ezenfelül még lehetőség nyílik a tartalom kibővítésére származtatott attribútumokkal vagy kapcsolatokkal is.

Egy komponens elemre (objektumra) tekinthetünk ezért úgy, mint egy oldal tartalmára, amelyet meg kell jeleníteni (a prezentációs modell által megadott módon) a szükséges navigációs elemekkel együtt, melyeket a navigációs modell szolgáltat. Ezzel nézeteket alakíthatunk ki a szerkezeti modell fölött, amely kimondottan hasznos olyan esetekben, amikor az egyes oldalak tartalmához különböző felhasználói csoportok férhetnek hozzá, hiszen ugyanazon szerkezeti elemekhez eltérő nézeteket definiálhatunk.

Mindezek mellett további (összetett) kompozíciók is létezhetnek az oldalaknál, mint például egy oktatói oldal esetén az általa irányított hallgatók száma esetleg kiegészítve statisztikai adatokkal (pl. az elmúlt öt év referált publikációinak a száma). Ilyen esetekben a természetes kapcsolatokból származó összefüggéseket szükséges kiegészíteni további asszociációkkal a szerkezeti modell megfelelő osztályaihoz.

4.4.4 Prezentációs modell

A prezentációs modell célja a komponens modell elmeinek a leképzése különböző jól ismert grafikus felülethez kötődő elemekre. Ennek *első lépését még koncepcionális szinten is elvégezhetjük*, ahol még nem vesszük figyelembe az implementációs részleteket. Ezt követően egy PIM-PSM transzformáció során leképezzük ezen elemeket egy adott platformra, bár előfordulhat, hogy nem minden koncepcionális elemhez létezik megfelelő grafikus elem. Ilyenkor helyettesíteni kell ezen elemeket például egy hierarchikus szerkezet esetén egy allistákból álló listával. Miután számos prezentációs felület létezik, a disszertációban ezen leképzési lehetőségeket nem ***tárgyaljuk***, kivéve ***az adatmanipulációért felelős oldalakat***, amelyet a következő részben ismertetett módon származtatunk.

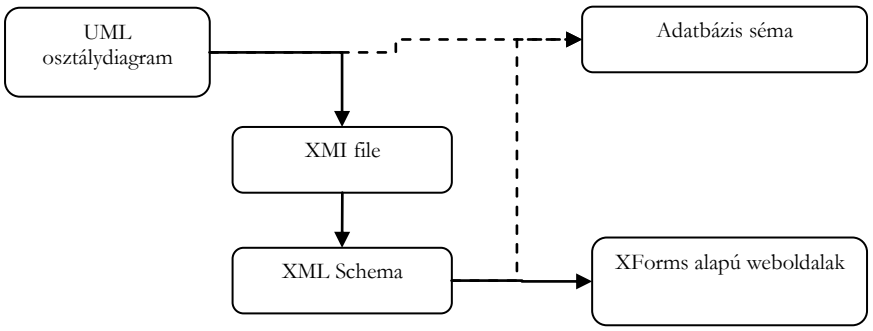
4.5. PIM-PSM transzformáció

Az MDA irányelveket követve számos ***transzformációt alkalmazunk***, hogy a platform független modellekből (*PIM*) platform specifikus modelleket (*PSM*) készítsünk. A szerkezeti modell esetén a Hibernate keretrendszert használjuk, amely egy implicit specifikus (relációs) modellt biztosít, így a transzformáció lépéseit nem részletezzük. A ***komponens és navigációs modelleket*** a prezentációs modellben megadott oldalsablonokkal együtt használjuk fel a felhasználótól érkező adatok kezeléséért felelő ***XForms alapú oldalak származtatásához***. Az egyes *platform specifikusmodelleket XML dokumentumok reprezentálják*, amelyeket az oldalsablonok felhasználásával XSLT transzformációkkal alakítunk át weboldalakká.

4.6. Kódgenerálás

A koncepcionális modellek elkészítéséhez az Eclipse Modeling Framework-öt használjuk. Ezek a modellek fogják szolgáltatni az ***általunk kidolgozott alkalmazáson belüli XML kommunikáció*** alapját. Már a felhasználó által kitöltött űrlap elküldésekor is egy XML dokumentum készül és ez kerül át a kiszolgáló oldalra. Természetesen ezen adatokat validálni kell a szerkezeti modellel szemben, melyben az XML dokumentumok érvényesítésénél elérhető sémanyelvek (DTD, XML Schema, RealaxNG, Schematron) segítenek. Egy modul segítségével (hyperModel) az UML2 ***osztálydiagramjainkat transzformáljuk egy XML Schema dokumentummá***, amelyet a folyamat további lépéseiben fel tudunk használni. A séma központi szerephez jut, mert egyrészt felhasználjuk az XForms alapú oldalak generálásához, illetve az alkalmazáson belüli kommunikáció során előálló XML dokumentumok validáláshoz. Az elkészült XForms oldalakat pedig összetettebb XHTML oldalakba ágyazhatjuk, amelyek megfelelnek a koncepcionális modellezés fázisában elkészült navigációs és komponens modelleknek.

A modellek létrehozása során az OMG XMI (*XML Metadata Interchange*) formátuma biztosítja a megfelelő eszközfüggetlenséget. Miután az XMI egy ipari szabvány az UML modellek metaadatainak XML dokumentum formájában történő cseréjéhez, a számításba kerülő modellező eszközök mindegyike támogatja, így a fejlesztés során az egyes csoportok akár különböző eszközöket is használhatnak. Elsőként az UML modelljeinket elmentjük XMI formátumban. Ezt követően ***automatikusan előállítjuk az XML Schema-t*** az Eclipse-be beépülő és az XMI-t feldolgozó hyperModel segítségével. Az adatok kezeléséért felelős XForms oldalakat pedig ezen sémából származtatjuk az ***általunk elkészített XSLT transzformáció segítségével***. A módszerünk lépéseit az alábbi ábra foglalja össze.



2. ábra: UML és XML technológiák használata a kódgeneráláshoz

Az ábrán folytonos vonallal jelöltük az általunk végrehajtott transzformációs lépéseket, amelynek eredményeképp a platform független modellünkből kialakult több platform specifikus modell is. *Ezen* ***modelljeink lehetővé teszik a webalkalmazásunk egy működő prototípusának gyors előállítását***. Az ábrán szaggatott nyíllal jelöltük azokat az opcionális utakat, amelyet a módszer még magában rejt. Ha az XML Schema-ból szeretnénk az adatok tárolásáért felelős réteget előállítani, akkor választhatjuk a séma relációs modellé történő transzformálást. Erre számos megoldással találkozhatunk. Viszont az XML sémánk magán hordozza annak a lehetőségét, hogy az adatbázisként valamilyen natív XML adatbáziskezelő rendszert alkalmazzunk. Ezen rendszerek jelenleg még nem terjedtek el meghatározóan, de a későbbiekben számolni kell ezzel is. A másik szaggatott nyíl pedig magából az osztálydiagramból indul, ami annak a lehetőségét jelenti, hogy a korábban már említett MVC modell részhez Java osztályok származtathatóak bármilyen nehézség nélkül, amelyek egy objektumrelációs fordító segítségével könnyedén tárolhatóak relációs adatbázisokban (ilyen például a Hibernate keretrendszer).

5. Konklúzió

Az általunk bemutatott módszertan az *adatorientált webalkalmazások tervezését és fejlesztését támogatja* az MDA irányelveinek felhasználásával. Megmutattuk, hogy az egyes fejlesztési fázisok mennyire összetett és épp ezért közel sem szisztematikus lépésekből állnak. ***Az ismertetett módszer a hatékony és gyors webalkalmazás fejlesztést és az adatorientált feladatok ellátását segíti az UML és XML technológiák felhasználásával*** kis- és közepes méretű projektek esetén. Az implementációs fázisnál megvizsgáltuk az XML technológiák szerepét a moduláris, méretezhető és bővíthető webalkalmazások esetén. Természetesen a további kutatások további irányba is folytathatóak, amelyekkel az UML alapú módszertanunkat bővíthetjük.

Mindazonáltal a kutatásaink eredménye egy *átfogó web alapú információs rendszer fejlesztését támogató keretrendszer alapját szolgáltatja*. A negyedik részben ismertetett modellezési lépéseknél érdekes lehetőséget kínálnak még a személyre szabhatóság szempontjai. Ennek megvalósításához a rendszer különböző felhasználóihoz a rendszer különböző nézetei szükségesek. A modellekbe történő (szabványos jelöléseken alapuló) beemelése előrelépést jelenthetne egy magas szinten testre szabható és rugalmas rendszer kialakításához. Másrészt a prezentációs szempontok is bővíthetőek az aktuális technológiákkal, mint például az AJAX és az OpenLaszlo. Csupán a PIM-PSM modell transzformációkra van szükség, hogy az absztrakt megjelenítési modelljeinket leképezzük az aktuális technológiára.

További kutatási irányként *a modellek minőségi vizsgálata kínálkozik*. A rendszer minőségének javításához számos metrika definiálása és mérése szükséges. Az irodalomban többféle metrika is létezik, azonban csak néhány alkalmazható a webalkalmazások esetén. Ezen túlmenően azt is figyelembe kell vennünk, hogy az MDA megközelítés alkalmazásával az előállított kód túlnyomó része automatikusan generálódik. Miután az MDA alapú fejlesztések a modellek létrehozásán és azok kóddá történő transzformálásán alapulnak, célszerűbb lenne a *modellek minőségét mérni* a kód minősége helyett. Ehhez a modellek különböző jellemzőit leíró modellmetrikákra van szükség, amelyek meghatározásához további kutatások szükségesek. Ezt követően a minőség javításának érdekében már statisztikai módszerek is felhasználhatóak. A megfelelő mérési módszerekhez meg kell határozni a működési profilokat és az alkalmazható statisztikai modelleket annak érdekében, hogy a döntő fontosságú faktorokat és metódusokat megtaláljuk.

A bemutatott módszer a szakmai konferenciákon és a közvetlen környezetemben is komoly szakmai visszhangot váltott ki. Több fiatal kolléga érdeklődését is sikerült felkeltenem, közöttük többen vannak olyanok, akik a tudományos munkájukat ebben az irányban szándékoznak folytatni [8] [9] [10].

Irodalomjegyzék

[1] Gnaho, C., "Web-based Information Systems Development - A User Centered Engineering Approach." Lecture Notes in Computer Science, 2001., old.: 105-118.

[2] Scharl, A. és Gebauer J. and Bauer, C., "Matching Process Requirements with Information Technology to Access the Efficienty of Web Information Systems." Information Techology and Management 2(2), 2001., old.: 192-210.

[3] Evans, Eric., *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Addison Wesley, 2003.

[4] Mernik, M., Heering J. and Sloane A. M., "When and how to develop domain-specific languages." ACM Computing Surveys, hely nélk. : ACM, 2005., 37(4). kötet, old.: 316–344.

[5] Object Management Group., Model Driven Architecture (MDA). [Online] 2001. július. http://www.omg.org/mda/.

[6] Ceri, S., Fraternali, P. and Matera, M., "Conceptual Modeling of Data-Intensive Web Applications.": IEEE Internet Computing, 2002., 6(4). kötet.

[7] Hennicker, R., Koch, N., "A UML-based Methodology for Hypermedia Design." UML '2000 (LNCS 1939), 2000., old.: 410-424.

[8] Arató M., Fazekas G.,Kollár L., "Statistical measurement of software quality." Proc. of the 7th International Conference on Applied Informatics, 2007.

[9] Adamkó A., Arató M., Fazekas G., Juhász I., "Performance Evaluation of Large-scale Data Processing Systems." Proceedings of the 7th International Conference on Applied Informatics, 2007.

[10] Adamkó A., Bornemissza Cs., "Combining the Benefits of MVC Design Pattern and UML Based Modeling for Different Software Platforms." : Proceedings of the 7th International Conference on Applied Informatics, 2007.

[11] Holck, J., "4 Perspectives on WebInformation Systems." Proc. of the 36th Hawaii International Conference on System Science, 2003., old.: 265-275.

[12] Koch, N. and Kraus, A., "Towards a Common Metamodel for the Development of Web Applications." Proc. of the 3rd International Conference on WebEngineering (LNCS 2722), 2003., old.: 497-506.

[13] Conallen, J., *Building Web Applications with UML.* 2nd Edition. Boston : Addision Wesley, 2002.

[14] Mernik, M., Heering,J. and Sloane, A. M., "When and how to develop domain-specific languages." ACM Computing Surveys, 2005., old.: 37(4):316–344.

[15] Schmidt, D.C., "Model-Driven Engineering." IEEE Computer 39 (2), 2006., old.: 25-31.

[16] Schwabe, D., Rosi, G., "An Object-Oriented Approach to Web-Based Application Design. Theory and Practice." TAPOS Vol. 4, 1998., old.: 207-225.

[17] Ceri, S., Faternali, P., Bongio, A., "Web Modeling Language (WebML)." In Proc. of 9th World Wide Web Conference, 2000.

[18] Koch, N. and Kraus, A., "The expressive Power of UML based Web Engineering." Proceedings of the 2nd International Workshop on Web Oriented Software Technology, 2002.

[19] Conallen, J., *Building Web Applications with UML.* Boston : Addision Wesley, 1999.

[20] D Bäumer, D Riehle, W Siberski, M Wulf., "The Role Object Pattern." Proceedings of PLoP, 1997.

[21] Distante, D., Rossi, G., Canfora, G., and Tilley, S., "A Comprehensive Design Model for Integrating Business Processes in Web Applications." In International Journal of Web Engineering and Technology (IJWET) : Inderscience Publishers, 2007., Vol. 2, Issue 1. kötet, old.: pp 43-72.

[22] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, T. J. Grose., *Eclipse Modeling Framework.* Boston : Addison-Wesley, 2003.

[23] Fowler, M., *UML Distilled, Third Edition.* s.l. : Addision-Wesley, 2003.

[24] —. Dealing with Roles. [Online] 1997. július 20. http://martinfowler.com/apsupp/roles.pdf.

[25] Gamma, E., Helm, R., Johnson J. and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software.* Boston : Addision Wesley, 1995.

[26] Gingeri A., Murgesan S., "The Essence of WebEngineering." IEEE Multimedia Vol. 8., 2004.

[27] Gomez, J., Cachero, C. and Pastor, O., "Extending a Conceptual Modeling Approach to Web Application Design." Proceedings of The 12th International Conference on Advanced Information Systems Engineering : Sprinter Verlag, 2000., LNCS 1789. kötet.

[28] Jacobson I., Booch G., Rumbaugh J., *The Unified Software Development Process.* s.l. : Addision Wesley, 1999.

[29] Kleppe, A, Warmer, J., Bast, W., *MDA Explained.* Boston : Addision Wesley, 2003.

[30] Koch, N., Zhang, G., Escalona, M., j., "Model Transformations from Requirements to Web System Design." New York : ACM Press, 2006.

[31] Kruchten, P., *The Rational Unified Process: An Introduction.* USA : Addision Wesley, 1999.

[32] Langham, M., Ziegeler, C., *Building XML Applications.* USA : Sams Publishing, 2002.

[33] L. Baresi, F. Garzotto, P. Paolini., "Meta-modeling Techniques meets Web Application Design Tools." Proc. of FASE 2002 : Springer Verlag, 2002., LNCS 2306. kötet, old.: 294-307.

[34] Mellor, S. J., Balcer, M. J., *Executable UML.* Indianapolis : Addision Wesley, 2002.

[35] Meliá, S., Gómez, J., Koch, N., "Improving Web Design Methods with Architecture Modeling." hely nélk. : 6th International Conference on E-Commerce and Web Technologies, 2005.

[36] Mendes, E., Mosley, N., "Web Engineering." Germany : Springer, 2006.

[37] Object Management Group., *QVT MOF 2.0 Query/Views/Transformations RFP.* 2002. October.

[38] —. "Meta Object Facility 2.0 Core Specification." [Online] 2003. http://www.omg.org/docs/ptc/03-10-04.pdf.

[39] —. XML Metadata Interchange (XMI). *XML Metadata Interchange (XMI).* [Online] Object Management Group, 2003. http://www.omg.org/.

[40] —. "Object Constraint Language." [Online] május 2006. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#OCL.

[41] —. "Unified Modeling Language." [Online] november 2007. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML.

[42] Leune, O. M. F. De Troyer and C. J., "WSDM: a user centered design method for Web sites." Proceedings of the Seventh International World Wide Web Conference, 1998.

[43] Reenskaug, T. M. H., *Models - Views – Controllers.* USA : Xerox PARC, 1979.

[44] Riehle, D., "Describing and Composing Patterns using Role Diagrams." In Proc. Ubilab Conference, 1996., old.: 137-152.

[45] Rossi, G., Gordillo, S., and Distante, D., "Improving Web Applications Evolution by Separating Design Concerns." hely nélk. : IEEE Software Technology and Engineering Practice, 2005.

[46] R. Soley et. al., Object Management Architecture Guide. [Online] 1997. http://doc.omg.org/ab/97-05-05.

[47] S. Ceri, P. Fraternali, M. Brambilla, A. Bongio, S. Comai, M. Matera., *Designing Data-Intensive Web Applications.* USA : Morgan Kaufmann, 2002.

[48] Schmid, H.A., Donnerhak, O., "OOHDMDA - An MDA Approach for OOHDM." In Proceedings of the 5th International Conference on Web Engineering : LNCS, 2005., 3579. kötet, old.: 569-574.

[49] Schwabe, D., Rosi, G., "Web Applications Models are more than Conceptual Models." Proc. of the International Workshop on Conceptual Modeling and the Web, 1999.

[50] Sun Microsystems, Inc., Sun ONE Architecture Guide. [Online] 2002. http://www.sun.com/software/sunone/docs/arch.

[51] T. Berners-Lee, J. Hendler, O. Lassila., "The Semantic Web." *Scientific American.* 2001.

[52] Portal, Web Engineering Community., *http://webengineering.org/*. [Online] http://webengineering.org/.

[53] World Wide Web Consortium., XML Schema. [Online] W3C, 2004. október. http://www.w3.org/TR/xmlschema-0/.

[54] —. Cascading Style Sheets. [Online] W3C, 2006. http://www.w3.org/Style/CSS/.

[55] —. Extensible Markup Language (XML) 1.1 (Second Edition). [Online] W3C, 2006. augusztus. http://www.w3.org/TR/2006/REC-xml11-20060816/.

[56] —. XForms 1.1. [Online] W3C, 2007. november. http://www.w3.org/TR/xforms11/.

[57] —. XSL Transformations (XSLT) Version 2.0. [Online] W3C, 2007. január. http://www.w3.org/TR/xslt20/.

[58] —. XHTML™ 1.1 - Module-based XHTML - Second Edition. [Online] W3C, 2007. február. http://www.w3.org/TR/xhtml1.

[59] —. Web Services Description Language (WSDL). [Online] W3C, 2007. június. http://www.w3.org/TR/wsdl20-primer.

[60] —. XQuery 1.0: An XML Query Language. [Online] W3C, 2007. január. http://www.w3.org/TR/xquery/.

[61] Zhao, W., Kearney, D. and Gioiosa G., "Architectures for Web based Appliactions." AWSA : ismeretlen szerző, 2002.

Publikációs lista

I. Nemzetközi konferenciák referált kiadványában megjelent dolgozatok

1. Adamkó A.: *Design concepts for data intensive Web applications*, Proceedings of the 6th International Conference on Applied Informatics, Eger, Hungary, 2004., Volume II, pp. 9
2. Adamkó A.: *New methods in Web application modeling with UML and XML,* IEEE EuroCon 2005, Belgrade, Serbia, 2005., Proceedings, **IEEE** Catalog Number: 05EX1255C, ISBN: 1 4244 0050
3. Adamkó A.: *Rapid Web Application Development and Modeling, based on XML and UML technologies*, Proceedings of the 7th International Conference on Applied Informatics, Eger, Hungary, 2007.
4. Adamkó A., Arató M., Fazekas G., Juhász I.: *Performance Evaluation of Large-scale Data Processing Systems*, Proceedings of the 7th International Conference on Applied Informatics, Eger, Hungary, 2007.
5. Adamkó A., Bornemissza Cs.: *Combining the Benefits of MVC Design Pattern and UML Based Modeling for Different Software Platforms*, Proceedings of the 7th International Conference on Applied Informatics, Eger, Hungary, 2007.
6. Adamkó A., Kollár L.: *MDA-based Development of Data-Driven Web Applications*, Web Information Systems and Technologies (**WEBIST** 2008), Funchal, Portugal, 2008.

II. Nemzetközi konferenciák kiadványaiban megjelent dolgozatok

7. Adamkó A.: *Web Information Systems Engineering: problems and solutions*, CSCS 2004, The Fourth Conference of PhD Students in Computer Science Szeged, Hungary, July1 - 4, 2004., Volume of extended abstracts, page 18. Full paper submitted
8. Adamkó A., Bornemissza Cs.: *Planning and Developing Dynamic Web Sites in different platforms,* CSCS 2004, The Fourth Conference of PhD Students in Computer Science Szeged, Hungary, July1 - 4, 2004., Volume of extended abstracts, page 19. Full paper submitted

III. Nemzetközi folyóirat cikkek

9. *UML-based Modeling of data-oriented Web Applications*, Journal of Universal Computer Science, **Vol. 12** / **Issue 9** page 1104-1117, 2006

IV. Hazai konferenciák kiadványaiban megjelent dolgozatok

10. Adamkó A.: *E-business támogató CA rendszerek*, Informatika a felsőoktatásban 2002, Debrecen,Hungary, 2002. aug. 28-30., 7 oldal, http://www.date.hu/if2002, cd kiadvány

11. Adamkó A.: *Elektronikus információs és nyilvántartási rendszer a doktori iskolák fiatal kutatói részére*, Networkshop 2004, Győr, Hungary, 2004. április 5-7., http://nws.iif.hu, cd kiadvány

12. Adamkó A.: *Doktori Iskolák egy Információs Rendszere*, AgriaMédia 2004, Eger, Hungary, 2004. október 18-19., cd kiadvány

13. Adamkó A.: *Webalkalmazások modellezése új tervezési stratégiákkal,* DOSZ – Tavaszi Szél 2005, Debrecen,Hungary, 2005. május 5-8., konferencia kiadvány, 4-7

14. Adamkó A.: *Uml alapú adatcentrikus Webalkalmazások modellezése,* Informatika a felsőoktatásban 2005, Debrecen, Hungary, 2005. aug. 24-26., cd kiadvány

V. Lektorált oktatási segédanyag

15. Adamkó Attila: *Operációs rendszerek 1. Gyakorlati segédanyag*,
mobiDIÁK könyvtár, 2004

Frissítése és bővítése: 2008. tavasz

VI. Konferencia előadások:

magyar nyelven:

1. *E-business támogató CA rendszerek*, Informatika a felsőoktatásban 2002, 2002. augusztus 29.

2. *Elektronikus információs és nyilvántartási rendszer a doktori iskolák fiatal kutatói részére*, Networkshop 2004, 2004. április 6.

3. *Doktori Iskolák egy Információs Rendszere*, AgriaMédia 2004, Eger, 2004. október 19.

4. *Webalkalmazások modellezése új tervezési stratégiákkal*, Tavaszi Szél 2005, Debrecen, 2005. május 7.

5. *Uml alapú adatcentrikus Webalkalmazások modellezése*, Informatika a felsőoktatásban 2005, Debrecen, 2005. augusztus 25.

angol nyelven:

6. *Design concepts for data-intensive Web applications*, 6th International Conference on Applied Informatics, Eger, Hungary, 2004. január 27.

7. *Web Information Systems Engineering: problems and solutions*, CSCS 2004, The Fourth Conference of PhD Students in Computer Science, Szeged, 2004. július 2.

8. *Planning and Developing Dynamic Web Sites in different platforms*, CSCS 2004, The Fourth Conference of PhD Students in Computer Science, Szeged, 2004. július 2.

9. *New methods in Web application modeling with UML and XML*, IEEE EuroCon 2005, Belgrad, Serbia, 2005. november 22.

10. *Performance evaluation of large-scale data processing systems*, XXVI. Seminar on Stability Problems for Stochastic Models, Sovata, Romania, 2006. augusztus 28.

11. *Rapid Web Application Development and Modeling, based on XML and UML technologies*, 7th International Conference on Applied Informatics, Eger, Hungary, 2007. január 30.

12. *Performance Evaluation of Large-scale Data Processing Systems*, 7th International Conference on Applied Informatics, Eger, Hungary, 2007. január 31.

VII. Szakmai előadások

1. *Webalkalmazások modellezése*, Tanszéki szeminárium, Debrecen, 2005. április 19.

2. *Webes Információs Rendszerek fejlesztése*, V. Gyires Béla napok, Debrecen, 2005. november 18.

VIII. Szoftver termék

1. A Debreceni Egyetem Matematika és Számítástudományok Doktori Iskolájának Web alapú Információs Rendszere, 2004-2005, ~12.000 sor program kód + dokumentáció

2. Debreceni Egyetem Informatikai Kar, Kari honlap oktatás részének elkészítése, üzemeltetése és szerkesztése

1 Introduction

The evolution of Web technologies increased the presence of the Web in our everyday life starting from personal home pages through corporate portals and Web shops up to Web applications implementing complex business processes. This diversity of applications makes the selection of the appropriate technology and platform harder, in particular, when these applications should collaborate with each other.

Some of them might be suitable in a given case, which are sometimes alternatives to each other, while they should be combined in other cases. As these technologies evolve very fast, they might become obsolete soon. It is very hard to predict, which technologies will be out "tomorrow" so business logic should be as independent as possible from technologies to allow change in the technology without affecting business processes and rules.

The modeling of such systems is a very complex process since (in an ideal case) it should take both existing and future technologies into consideration in order to create a flexible system which allows further development. Despite changing of the technology, models of the application domain remain almost the same since they capture business model and processes. The best practice tells us to keep the modeling of the application domain and the technological details separated.

1.1 Web Information System Development

In our research, we are mainly considering a subset of the aforementioned applications of the Web: the Web Information Systems (WISs). "*A WIS is an Information System providing facilities to access complex data and interactive services via the Web*" [1]. These "*represent a sub-category of mass information systems that are typically support on-line information retrieval and routine task by way of self-service for a large number (thousands or millions) of occasional users who are spread over many locations*" [2].

Table 1 shows a categorization of these systems depending on the direction of communication and the characteristics of information [11].

	Asymmetrical communication	Symmetrical communication
Objective information	Information provider	Information system
Persuasive information	Advertisement	Community

Table 1: Four perspectives of WISs.

The focus of information providers is clearly on one-way distribution of information, from the system to its external users. A news portal is an example of such a system. A WIS can also be seen as a marketing channel, on purpose to distribute promotional messages to an external audience: a home page can be considered like an "advertisement" system. A WIS can be used to create and develop a virtual community whereof forums are examples. The fourth perspective is also known as the development of data-intensive Web applications. Typical examples of such WISs are corporate portals, on-line flight booking systems, etc. Their focus is on the data structure and the information flow between the system and the users to support their work by the help of complex business processes which both require and provide information from/to the users.

In order to save time, development groups often disregard methodological approaches during the modeling of such systems. In most of the cases, this leads to an improper design and individual (not reusable) modeling practices. These make dynamic responses to emerging needs just on time almost impossible since new requirements might cause complete redesign of some system parts or even the whole system. The re-implementation of the affected system parts could be a very time-consuming operation. Its reason is that there is a gap between the model and the implementation of a system. Even when applying traditional software design methodology, once a model is created, it is "thrown away" after the first implementation and is neither used again nor updated. There is no direct connection between the model and the code so changes in the model are not reflected in the code, and contrary, changes in the code are not reflected in the model. This means that after any of them are modified, they are not synchronized anymore; therefore, the model cannot fulfill one of its primary purposes, namely, to tell relevant information about the system.

2 Motivation

2.1 The legacy system

Several years ago the author was involved in a project whose aim was to develop a Web-based application for the Doctoral School of Mathematics and Computer Science at the University of Debrecen. The requirements against the system were to keep track of students and teachers of the Doctoral School and, manage data of doctoral programs and courses as well. Besides the usefulness of having these data collected, it also allows the creation of some statistical reports. Moreover, teachers and students are able to have their list of publications collected. According to Table 1, this system can be considered as an information system since it is interactively used to accomplish their tasks including querying, modifying and inserting data.

The back-end of the system was a PostgreSQL DBMS while the front-end consisted of HTML pages combined with CGI scripts written in Perl. The application logic is multi-layered: utilizing PL/pgSQL stored procedures (executed inside the DBMS) and Perl.

Nowadays, almost all of these technologies are relatively rarely used as they do not support a high level of separation between the structure and the presentation of data.

2.2 Problems

After deploying the system new requirements arose. Some of them were easy to implement but a few of them required the redesign of the whole data model. This was partly needed due to the changes of the legal regulation and organizational restructuralization. On the other hand, most of the problems arose when the underlying data model had to be changed. This caused several other changes since it influenced other system parts as well.

These issues outlined that the maintenance and further development of the system is complicated therefore hard to accomplish. As a consequence the importance of the reconsideration of the whole system became apparent.

In general, problems are arising when requirements are evolving (which is more often the case), since maintenance might be very problematic as new developers are probably not so familiar with the older technologies and even if they are, this may lead to an improper design and/or coding strategy. For that reason, we considered to apply a model-based solution to reduce development time and maintenance costs.

3 Theoretical background

3.1 Domain Driven Design

Domain Driven Design is an approach for dealing with highly complex domains which is based on making the domain itself the main focus of the project, and maintaining a software model that reflects a deep understanding of the domain. We need to understand from the beginning where the software is originated from and which domain (environment) is related to. Domain Driven Design helps us to understand and to focus on the problem domain and to build a domain model which captures the basic concepts. During the construction of the domain knowledge we have the possibility to extract the essential information and to generalize it.

Domain Driven Design uses basic building blocks to define the domain itself. These blocks are entities, value objects, services and modules. Services act as interfaces which provide operations. The purpose of service is to simply provide functionality for the domain. Using modules in design is a way to increase cohesion and decrease coupling. Moreover, it uses well known design patterns like Aggregate, Builder, Factory and Repository to decrease complexity. Aggregate is a domain pattern used to define object ownership and boundaries. Factories and Repositories are two design patterns which help us deal with object creation and storage.

3.2 Domain Specific Languages

A DSL is a language specially geared to working within a particular area of interest: it might be a vertical domain such as telephone design, or a horizontal one like workflow. Some well-known examples of DSLs are HTML, SQL and UNIX mini languages. What is radically new is the idea of creating your own DSL for your own project. Domain specific languages have important design goals that contrast with those of general-purpose languages:

- domain specific languages are less comprehensive.
- domain specific languages are much more expressive in their domain.
- domain specific languages should exhibit minimum redundancy.

In Model Driven Engineering many examples of domain-specific languages may be found: like OCL, a language for decorating models with assertions or QVT, a domain specific transformation language. However, languages like UML are typically general purpose modeling languages.

3.3 Model Driven Architecture

OMG's Model Driven Architecture (MDA) [5] provides a framework which allows applications to be described in a platform-independent manner. In MDA, artifacts are formal models describing a given aspect of the system. MDA uses UML as a standard modeling language in order to achieve a common understanding among stakeholders.

During model-driven development, a platform-independent model (PIM) is created first. This model should have a high level of abstraction hence implementation details—database system, application server platform, etc.—should be hidden. PIM models give viewpoints on how the system will support the business. The next step is a transformation of a PIM into one or more platform-dependent models called Platform Specific Models (PSMs).

The transformation process will deal with a specific set of technologies. For each specific technology a separate PSM is generated because most of today's systems span several technologies. The final step in the development process is the transformation of each PSM to code. Because a PSM fits its technology rather closely this transformation could be done relatively easy.

In case of a technological change (or evolution), a PIM is not subject to change as it models an application in a platform-independent manner. All what we need is to create a PSM in order to show how platform-independent components manifest on the new platform or technology. A PSM metamodel should be defined (or refined) which can describe the new platform in an abstract way to make the creation of platform-specific models easier. Transformation rules are also needed for mapping PIM metamodel elements to PSM metamodel elements. Applying them to an existing PIM will result PSMs for the new platform.

3.4 UML as Modeling Language

The MDA approach requires a language which uses formal definitions so the tools will be able to transform these models automatically. OMG recommends the use of UML to construct platform-independent models. The power of UML in our case is the modeling of the structural aspects of a system. This is done through the use of class diagrams which enables the generation of PSMs with all structural features.

3.5 UML Profiles

The semantics of UML models can be extended by applying UML profiles by adding stereotypes and tagged values which is a common way for building UML models for particular domains. These profiles are considered to reside in the metamodel layer in order to define UML "dialects" that can be used by models in the model layer. This mechanism is useful for both PIMs and PSMs.

For PIMs, a new profile should be defined to support a given design methodology. There are several profiles for that purpose [12] [7] [13], but they might not be appropriate for a different design strategy.

Many popular UML Profiles for specific platforms (e.g., UML Profile for EJB, UML Profile for CORBA, and so on) also exist but due to the extensibility mechanism new ones may also be created. This is how new platforms and technologies should be handled.

Therefore, UML Profiles might serve as a basis for defining a PIM–PSM transformation. This should be defined using the concepts of the metamodels used for describing PIM (source) and PSM (target).

4 New Results

We proposed a systematic design method for Web applications which takes into account the data-oriented aspects of these applications. We presented guidelines for the development of domain models adopted for this problem domain. Our method is based on an extension of the UML metamodel which is influenced by the paradigm of Domain Driven Design [3]. For the problem domain a custom Domain Specific Language is created [14] utilizing the concepts of roles. These domain model elements are mapped to a metamodel for our methodology`s structural model. An UML profile is derived by means of stereotypes and tagged values. The next step in our suggested development process is the construction of the navigational model. We provide a method to derive the navigation model from the structural model of a Web application based on the UWE [7] method. Using UML2 features we construct components from navigational entities in order to realize conceptual information provider elements. These elements can be connected with each other through interfaces to form an abstract page containing the necessary information. These pages could be transformed by the presentational layer to the desired format achieving universal client access.

To exploit the MDE development approaches [15] we utilize UML and XML technologies to realize code template generation from our models. Using standards (like OMG's XMI format) and recommendations from W3C (like XML Schema, XSLT and XForms) we are able to generate the data manipulation pages as well. The created artifacts make our method valuable for data-oriented Web application development.

4.1 Definition of a custom DSL

Following the guiding principle of the Domain Driven Design methodology we have created our custom Doman Specific Language for data-oriented Web Information Systems. The main advantage of our specific language is the ability to capture the concept of different roles in the problem domain. Using roles in the domain model is essential to outline the different behavior of the entities. If we deal with this characteristic we have the opportunity to precondition our design phase of the development process for the proper handling of modeling artifacts. We have realized this behavior by extending the definition of the entity term. However, this extension requires the definition of the appropriate services in the entity as well.

The service and module definition is not modified, but one should think about the module as a boundary of coherent concepts. If we introduce roles in entities it is recommended to create a separate module for them with the required services for each aspect.

4.2 UML metamodel extension and Profile derivation

A metamodel provides a precise definition of concepts used in the modeling steps, including modeling elements, their relationships and well-formedness rules. Our goal is to extend the UML metamodel with the concepts of the problem domain. This is achieved by deriving custom classes and associations from the UML metamodel elements. We have established the relationship between the structural elements using the Role Object pattern. Moreover, a navigational metamodel is created by adopting the modeling elements of the UWE method.

In order to keep CASE tool support we derive the proper UML Profiles from these metamodels. The profiles are comprises the definition of stereotypes, tagged values and constraint about their usage. The modeling tools can check that a model conforms to a given profile.

4.3 Development Process

The first step during system design is the analysis of requirements to gather and formalize user requests. Using use cases and activity diagrams, we could determine the outline of the system and describe its fundamental functional aspects from the different users' viewpoints.

Actors represent different roles of users of the Web application. "Use cases" are used to describe available operations which are used for determining the users' activities. This is important because they serve as a bird's eye view of the possible actions appearing on the start page of the related user groups which might be performed by users belonging to the group.

Activity diagrams are based on use cases and they are used to describe business logic. Moreover, they could be used to derive program modules and code skeletons, as well. Of course, our case is much easier as the structural model is deeply bound to the data structure, and our focus is on the data management and manipulation tasks which should be performed.

The second step is the conceptual modeling of data structure and access paths of the application. This is achieved by UML2 class diagrams which apply our UML profile. At conceptual level, structural, navigational, component and presentational models are created. This approach is well- known in the literature. Several design methods (OO-HDM [16], WebML [17], UWE [18], Jim Conallen's WAE [19], etc.) follow more or less the same way.

After a platform-independent model has been developed, it is subject to a model transformation which results in a platform-specific model which serves as a starting point for code generation. Our method is illustrated in the next Figure.

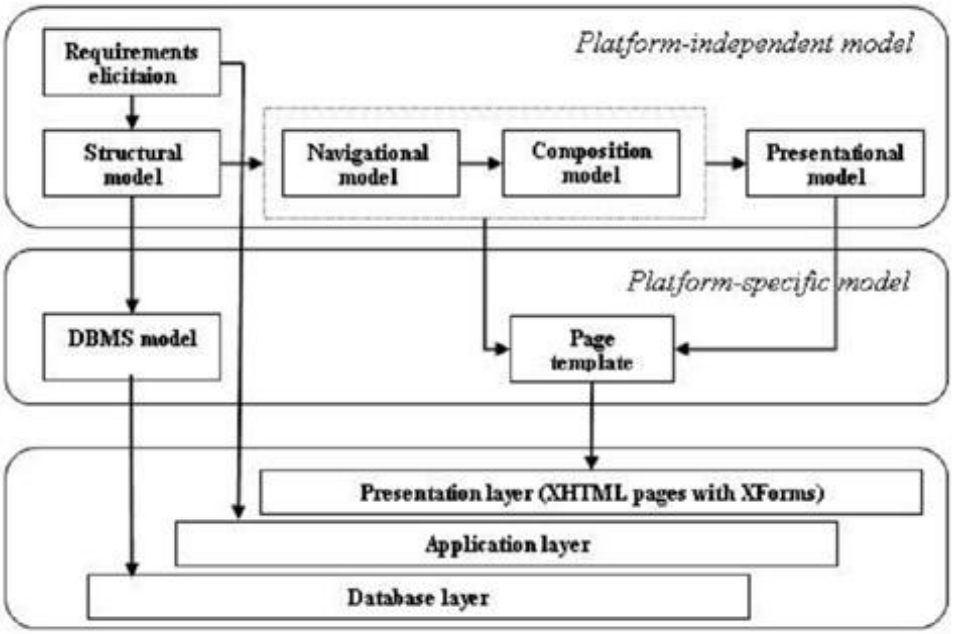


Figure 1: Phases of development

4.4 Models of the Design Phase

4.4.1 Structural Model

Main modeling elements of the structural model are classes, associations and packages. For a common Web application, use cases and activity diagrams are the base of the conceptual design of the domain.

However, in data-centric applications (which we consider), the structural model of the application domain is typically much more complex than any other models that have been mentioned previously. It is very important to create a class diagram which is of a good quality since many other activities (such as generating some primitive navigational elements like information access along associations) are relying on it.

4.4.2 The concept of roles

One of the advantages of our approach is the introduction of the roles in the structural model. There are situations when an object must provide different services (data, behavior) according to the context in which they are accessed. Like an academic accounting system which needs to track students and academics but a particular person may act sometimes as a student or sometimes as an academic. We use the role concept as a set of properties which are important for an object to behave in a particular context. In order to describe the different roles we have made use of the Role Object design pattern.

4.4.3 Navigational Model

The next stage in the development process is the navigational design. The navigational model specifies the elements of the structural model can be accessed from other parts of the application. In the navigation model's building process, the developer takes crucial design decisions such as what navigation paths are required to ensure the application's functionality. These decisions are based on the component model, use case diagrams and navigational requirements that the application needs to satisfy.

The navigational diagram is strongly connected to the structural one since it defines the navigational paths among structural model elements. In general, new associations might be added for direct navigation to avoid navigation paths of length greater than one. However, there may be some classes in the structural model that are not subject to be a visiting target. Therefore, they should be omitted from the navigational diagram. The navigation inside the application mostly occurs along associations which are used to describe the relation between structural elements. Typically, these associations appear as either hyperlinks or menus in the user interface.

The navigational diagram supplements the structural model with some additional associations and classes representing different access structures such as menus, queries and indices.

4.4.4 Component model

The Component model is intended to define the basic modules of the system. These modules are the concepts that are described our domain specific language with the module definition. Modules may have services defined in the DSL, so they will expose the necessary interfaces to provide their functionality. Moreover of the entity definition, we use modules to join together the navigational classes with their navigation structures (like menus and indexes). This implies modules will expose the required interfaces for the navigation. These interfaces are determined by the navigational context. Naturally, the behaviors for the introduced navigational roles are also expressed by the components.

These components will serve the basis for the presentational model mapping between concepts of the structural model and Web pages. One should think of a component as an object which is associated with the entities of the structural model. Hence, a page can arbitrarily intermix information coming from multiple entities. Moreover, it is possible to extend the content with derived attributes or relationships.

An element (object) of the component model could be considered as the content of a page which should be rendered somehow (described by the presentational model) along with several navigational elements (given by using the navigational model). This actually provides a view over the structural model elements. It can be extremely useful when different user groups are subject to access different page contents since several views might be defined on the same structural elements.

However, there could be more complex compositions such as when a supervisor's page should hold some pieces of information about his/her supervisees along with some data having statistical character (e.g., the number of the supervisor's refereed publications in the past 5 years). In such cases, the generated skeleton must be complemented with additional associations (to the appropriate classes of the structural model).

4.4.5 Presentational Model

The main goal of presentational modeling is to map the elements of the component model to some well-known GUI primitives. This task can also be applied at a conceptual level since it does not hold information about particular implementation. After the PIM–PSM transformation, some of those primitives might be replaced by others if the target platform does not support the given one (e.g., tree view of some hierarchical data might be replaced by a list with indented sublists). However, there are several presentational frameworks available we are not treated presentational aspects in this thesis except the generation of the data manipulation pages.

4.5 PIM–PSM transformations

Following the MDA principles we use several transformations to create PSMs from PIMs. For instance, at the structural model we use the "Hibernate" framework which provides an implicit PSM (relational model) so we does not deal with issues of this kind of model transformation. Composition and navigational models are used to formalize page templates along with presentational model. These templates are used to generate concrete pages using W3C's XForms standard for handling user inputs. The PSMs are represented with XML documents which allow XSLT to be used for the page generation from templates.

4.6 Code Generation

We use the Eclipse Modeling Framework to create the conceptual models. These ***models will serve the basis for our proposed XML-based communication*** in the deployed application. When a user interacts with the system by filling and submitting forms, an XML document is created and passed to the server side. However, user-supplied data should be validated against the data model. Since XML documents' validation are based on any of the well-known XML schema languages (e.g., DTD, XML Schema, RelaxNG, Schematron), a schema was needed. A freely available Eclipse plug-in called hyperModel (http://www.xmlmodeling.com/hypermodel) is able to ***transform a UML2 class diagram into an XML Schema***, therefore we applied it in our process. The generated XML Schema is one of the most ***important parts of our architecture*** since it is used as the ***base of the generation of XForms pages***. On the other hand, it is applied for validating all the XML documents which are used for intra-system communication, as well. XForms pages might be embedded in more complex XHTML pages which conform to the component and navigational model defined at the conceptual modeling phase.

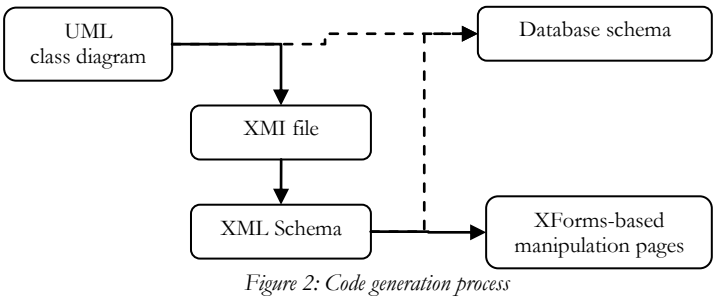


Figure 2: Code generation process

For creating models, OMG's XML Metadata Interchange (XMI) gives the possibility of a kind of tool-independence. As XMI is being an industrial standard for exchanging metadata of UML models in XML format, it is supported by all the major modeling tools so – in a collaborative environment – teams might use different tools. First of all, UML diagrams should be saved in an XMI-compliant format. Afterwards, XMI should be imported to Eclipse and let hyperModel do the transformation to XML Schema (.xsd). XForms pages are generated directly from the .xsd using an XSLT.

5 Conclusion

The presented approach describes a method to design and develop data-driven Web applications using MDA. We could see how complex it is therefore, these development steps are far from being systematic. We have elaborated a new methodology to help the development of Web applications rapidly and effectively. Our approach is based on UML and XML technologies supporting data management task in small- and medium-sized projects. We have added some important remarks concerning the implementation phase utilizing XML technologies to develop modular, scalable and expandable Web based systems. Ongoing researches can go in several interesting directions in the design and development phase. We are going to study the additional expandability of our UML-based methodology.

Obviously, the current state of our work could only be a part of a more comprehensive Web Information System development framework. Besides the modeling steps described in Section 4, it is also common to include modeling of personalization and presentation (Fraternali, 1999). In order to attain personalization, different participants of the system may need different viewpoints of the system's structure, navigation or presentation. These can be modeled using standard modeling notations which are big steps towards developing highly customizable and flexible systems. On the other hand, presentational aspects could be extended with current technologies like AJAX, OpenLaszlo, etc., only PIM–PSM transformations are required to define the mappings between abstract presentational models and concrete technologies.

Another possible research area is the measurement of models' quality. In order to improve system quality, several metrics should be defined and measured. There are several code metrics in the literature but only few of them are really useful. When applying model-driven development, even these code metrics might not be appropriate since the high majority of code is generated. As MDA development is based on creating models and transforming them into code, model quality should be measured instead of code quality. Therefore, model metrics are needed which are able to describe different properties of models. To obtain which possible metrics are worth measuring, further research is needed. Later on, statistical models should be applied to achieve higher quality. For the correct measurement we need to identify operational profiles and applicable statistical models to find the relevant factors and methods [9] [8].

References

[1] Gnaho, C., "Web-based Information Systems Development - A User Centered Engineering Approach." Lecture Notes in Computer Science, 2001, pp. 105-118.

[2] Scharl, A. and Gebauer J. and Bauer, C., "Matching Process Requirements with Information Technology to Access the Efficienty of Web Information Systems." Information Techology and Management 2(2), 2001, pp. 192-210.

[3] Evans, Eric., *Domain-Driven Design: Tackling Complexity in the Heart of Software.* s.l. : Addison Wesley, 2003.

[4] Mernik, M., Heering J. and Sloane A. M., "When and how to develop domain-specific languages." ACM Computing Surveys, s.l. : ACM, 2005, Vol. 37(4), pp. 316–344.

[5] Object Management Group., Model Driven Architecture (MDA). [Online] július 2001. http://www.omg.org/mda/.

[6] Ceri, S., Fraternali, P. and Matera, M., "Conceptual Modeling of Data-Intensive Web Applications." s.l. : IEEE Internet Computing, 2002, Vol. 6(4).

[7] Hennicker, R., Koch, N., "A UML-based Methodology for Hypermedia Design." UML '2000 (LNCS 1939), 2000, pp. 410-424.

[8] Arató M., Fazekas G.,Kollár L., "Statistical measurement of software quality." Proc. of the 7th International Conference on Applied Informatics, 2007.

[9] Adamkó A., Arató M., Fazekas G., Juhász I., "Performance Evaluation of Large-scale Data Processing Systems." Proceedings of the 7th International Conference on Applied Informatics, 2007.

[10] Adamkó A., Bornemissza Cs., "Combining the Benefits of MVC Design Pattern and UML Based Modeling for Different Software Platforms." : Proceedings of the 7th International Conference on Applied Informatics, 2007.

[11] Holck, J., "4 Perspectives on WebInformation Systems." Proc. of the 36th Hawaii International Conference on System Science, 2003, pp. 265-275.

[12] Koch, N. and Kraus, A., "Towards a Common Metamodel for the Development of Web Applications." Proc. of the 3rd International Conference on WebEngineering (LNCS 2722), 2003, pp. 497-506.

[13] Conallen, J., *Building Web Applications with UML.* 2nd Edition. Boston : Addision Wesley, 2002.

[14] Mernik, M., Heering,J. and Sloane, A. M., "When and how to develop domain-specific languages." ACM Computing Surveys, 2005, pp. 37(4):316–344.

[15] Schmidt, D.C., "Model-Driven Engineering." IEEE Computer 39 (2), 2006, pp. 25-31.

[16] Schwabe, D., Rosi, G., "An Object-Oriented Approach to Web-Based Application Design. Theory and Practice." TAPOS Vol. 4, 1998, pp. 207-225.

[17] Ceri, S., Faternali, P., Bongio, A., "Web Modeling Language (WebML)." In Proc. of 9th World Wide Web Conference, 2000.

[18] Koch, N. and Kraus, A., "The expressive Power of UML based Web Engineering." Proceedings of the 2nd International Workshop on Web Oriented Software Technology, 2002.

[19] Conallen, J., *Building Web Applications with UML*. Boston : Addision Wesley, 1999.

[20] D Bäumer, D Riehle, W Siberski, M Wulf., "The Role Object Pattern." Proceedings of PLoP, 1997.

[21] Distante, D., Rossi, G., Canfora, G., and Tilley, S., "A Comprehensive Design Model for Integrating Business Processes in Web Applications." In International Journal of Web Engineering and Technology (IJWET) : Inderscience Publishers, 2007, Vols. Vol. 2, Issue 1, pp. pp 43-72.

[22] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, T. J. Grose., *Eclipse Modeling Framework*. Boston : Addison-Wesley, 2003.

[23] Fowler, M., *UML Distilled, Third Edition.* s.l. : Addision-Wesley, 2003.

[24] —. Dealing with Roles. [Online] július 20, 1997. http://martinfowler.com/apsupp/roles.pdf.

[25] Gamma, E., Helm, R., Johnson J. and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston : Addision Wesley, 1995.

[26] Gingeri A., Murgesan S., "The Essence of WebEngineering." IEEE Multimedia Vol. 8., 2004.

[27] Gomez, J., Cachero, C. and Pastor, O., "Extending a Conceptual Modeling Approach to Web Application Design." Proceedings of The 12th International Conference on Advanced Information Systems Engineering : Sprinter Verlag, 2000, Vol. LNCS 1789.

[28] Jacobson I., Booch G., Rumbaugh J., *The Unified Software Development Process.* s.l. : Addision Wesley, 1999.

[29] Kleppe, A, Warmer, J., Bast, W., *MDA Explained.* Boston : Addision Wesley, 2003.

[30] Koch, N., Zhang, G., Escalona, M., j., "Model Transformations from Requirements to Web System Design." New York : ACM Press, 2006.

[31] Kruchten, P., *The Rational Unified Process: An Introduction.* USA : Addision Wesley, 1999.

[32] Langham, M., Ziegeler, C., *Building XML Applications.* USA : Sams Publishing, 2002.

[33] L. Baresi, F. Garzotto, P. Paolini., "Meta-modeling Techniques meets Web Application Design Tools." Proc. of FASE 2002 : Springer Verlag, 2002, Vol. LNCS 2306, pp. 294-307.

[34] Mellor, S. J., Balcer, M. J., *Executable UML*. Indianapolis : Addision Wesley, 2002.

[35] Meliá, S., Gómez, J., Koch, N., "Improving Web Design Methods with Architecture Modeling." s.l. : 6th International Conference on E-Commerce and Web Technologies, 2005.

[36] Mendes, E., Mosley, N., "Web Engineering." Germany : Springer, 2006.

[37] Object Management Group., *QVT MOF 2.0 Query/Views/Transformations RFP.* October 2002.

[38] —. "Meta Object Facility 2.0 Core Specification." [Online] 2003. http://www.omg.org/docs/ptc/03-10-04.pdf.

[39] —. XML Metadata Interchange (XMI). *XML Metadata Interchange (XMI).* [Online] Object Management Group, 2003. http://www.omg.org/.

[40] —. "Object Constraint Language." [Online] május 2006. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#OCL.

[41] —. "Unified Modeling Language." [Online] november 2007. http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML.

[42] Leune, O. M. F. De Troyer and C. J., "WSDM: a user centered design method for Web sites." Proceedings of the Seventh International World Wide Web Conference, 1998.

[43] Reenskaug, T. M. H., *Models - Views – Controllers.* USA : Xerox PARC, 1979.

[44] Riehle, D., "Describing and Composing Patterns using Role Diagrams." In Proc. Ubilab Conference, 1996, pp. 137-152.

[45] Rossi, G., Gordillo, S., and Distante, D., "Improving Web Applications Evolution by Separating Design Concerns." s.l. : IEEE Software Technology and Engineering Practice, 2005.

[46] R. Soley et. al., Object Management Architecture Guide. [Online] 1997. http://doc.omg.org/ab/97-05-05.

[47] S. Ceri, P. Fraternali, M. Brambilla, A. Bongio, S. Comai, M. Matera., *Designing Data-Intensive Web Applications.* USA : Morgan Kaufmann, 2002.

[48] Schmid, H.A., Donnerhak, O., "OOHDMDA - An MDA Approach for OOHDM." In Proceedings of the 5th International Conference on Web Engineering : LNCS, 2005, Vol. 3579, pp. 569-574.

[49] Schwabe, D., Rosi, G., "Web Applications Models are more than Conceptual Models." Proc. of the International Workshop on Conceptual Modeling and the Web, 1999.

[50] Sun Microsystems, Inc., Sun ONE Architecture Guide. [Online] 2002. http://www.sun.com/software/sunone/docs/arch.

[51] T. Berners-Lee, J. Hendler, O. Lassila., "The Semantic Web." *Scientific American.* 2001.

[52] Portal, Web Engineering Community., *http://webengineering.org/*. [Online] http://webengineering.org/.

[53] World Wide Web Consortium., XML Schema. [Online] W3C, október 2004. http://www.w3.org/TR/xmlschema-0/.

[54] —. Cascading Style Sheets. [Online] W3C, 2006. http://www.w3.org/Style/CSS/.

[55] —. Extensible Markup Language (XML) 1.1 (Second Edition). [Online] W3C, augusztus 2006. http://www.w3.org/TR/2006/REC-xml11-20060816/.

[56] —. XForms 1.1. [Online] W3C, november 2007. http://www.w3.org/TR/xforms11/.

[57] —. XSL Transformations (XSLT) Version 2.0. [Online] W3C, január 2007. http://www.w3.org/TR/xslt20/.

[58] —. XHTML™ 1.1 - Module-based XHTML - Second Edition. [Online] W3C, február 2007. http://www.w3.org/TR/xhtml1.

[59] —. Web Services Description Language (WSDL). [Online] W3C, június 2007. http://www.w3.org/TR/wsdl20-primer.

[60] —. XQuery 1.0: An XML Query Language. [Online] W3C, január 2007. http://www.w3.org/TR/xquery/.

[61] Zhao, W., Kearney, D. and Gioiosa G., "Architectures for Web based Appliactions." AWSA : s.n., 2002.

Publications

I. International conferences – reviewed papers

1. Adamkó A.: *Design concepts for data intensive Web applications*, Proceedings of the 6th International Conference on Applied Informatics, Eger, Hungary, 2004., Volume II, pp. 9
2. Adamkó A.: *New methods in Web application modeling with UML and XML,* IEEE EuroCon 2005, Belgrade, Serbia, 2005., Proceedings, **IEEE** Catalog Number: 05EX1255C, ISBN: 1 4244 0050
3. Adamkó A.: *Rapid Web Application Development and Modeling, based on XML and UML technologies*, Proceedings of the 7th International Conference on Applied Informatics, Eger, Hungary, 2007.
4. Adamkó A., Arató M., Fazekas G., Juhász I.: *Performance Evaluation of Large-scale Data Processing Systems*, Proceedings of the 7th International Conference on Applied Informatics, Eger, Hungary, 2007.
5. Adamkó A., Bornemissza Cs.: *Combining the Benefits of MVC Design Pattern and UML Based Modeling for Different Software Platforms*, Proceedings of the 7th International Conference on Applied Informatics, Eger, Hungary, 2007.
6. Adamkó A., Kollár L.: *MDA-based Development of Data-Driven Web Applications*, Web Information Systems and Technologies (**WEBIST** 2008), Funchal, Portugal, 2008.

II. International conferences – Proceedings

7. Adamkó A.: *Web Information Systems Engineering: problems and solutions*, CSCS 2004, The Fourth Conference of PhD Students in Computer Science Szeged, Hungary, July1 - 4, 2004., Volume of extended abstracts, page 18. Full paper submitted
8. Adamkó A., Bornemissza Cs.: *Planning and Developing Dynamic Web Sites in different platforms,* CSCS 2004, The Fourth Conference of PhD Students in Computer Science Szeged, Hungary, July1 - 4, 2004., Volume of extended abstracts, page 19. Full paper submitted

III. International Journal papers

9. *UML-based Modeling of data-oriented Web Applications*, Journal of Universal Computer Science, **Vol. 12** / **Issue 9** page 1104-1117, 2006

IV. National conferences – Proceedings

10. Adamkó A.: *E-business támogató CA rendszerek*, Informatika a felsőoktatásban 2002, Debrecen,Hungary, 2002. aug. 28-30., 7 oldal, http://www.date.hu/if2002, cd kiadvány

11. Adamkó A.: *Elektronikus információs és nyilvántartási rendszer a doktori iskolák fiatal kutatói részére*, Networkshop 2004, Győr, Hungary, 2004. április 5-7., http://nws.iif.hu, cd kiadvány

12. Adamkó A.: *Doktori Iskolák egy Információs Rendszere*, AgriaMédia 2004, Eger, Hungary, 2004. október 18-19., cd kiadvány

13. Adamkó A.: *Webalkalmazások modellezése új tervezési stratégiákkal,* DOSZ – Tavaszi Szél 2005, Debrecen,Hungary, 2005. május 5-8., konferencia kiadvány, 4-7

14. Adamkó A.: *Uml alapú adatcentrikus Webalkalmazások modellezése,* Informatika a felsőoktatásban 2005, Debrecen, Hungary, 2005. aug. 24-26., cd kiadvány

V. Educational Materials

1. Adamkó Attila: *Operációs rendszerek 1. Gyakorlati segédanyag*,
mobiDIÁK könyvtár, 2004

   Update: 2008. spring

VI. Conference talks:

Hungarian:

1. *E-business támogató CA rendszerek*, Informatika a felsőoktatásban 2002, 2002. augusztus 29.

2. *Elektronikus információs és nyilvántartási rendszer a doktori iskolák fiatal kutatói részére*, Networkshop 2004, 2004. április 6.

3. *Doktori Iskolák egy Információs Rendszere*, AgriaMédia 2004, Eger, 2004. október 19.

4. *Webalkalmazások modellezése új tervezési stratégiákkal*, Tavaszi Szél 2005, Debrecen, 2005. május 7.

5. *Uml alapú adatcentrikus Webalkalmazások modellezése*, Informatika a felsőoktatásban 2005, Debrecen, 2005. augusztus 25.

English:

6. *Design concepts for data-intensive Web applications*, 6th International Conference on Applied Informatics, Eger, Hungary, 2004. január 27.

7. *Web Information Systems Engineering: problems and solutions*, CSCS 2004, The Fourth Conference of PhD Students in Computer Science, Szeged, 2004. július 2.

8. *Planning and Developing Dynamic Web Sites in different platforms*, CSCS 2004, The Fourth Conference of PhD Students in Computer Science, Szeged, 2004. július 2.

9. *New methods in Web application modeling with UML and XML*, IEEE EuroCon 2005, Belgrad, Serbia, 2005. november 22.

10. *Performance evaluation of large-scale data processing systems*, XXVI. Seminar on Stability Problems for Stochastic Models, Sovata, Romania, 2006. augusztus 28.

11. *Rapid Web Application Development and Modeling, based on XML and UML technologies*, 7th International Conference on Applied Informatics, Eger, Hungary, 2007. január 30.

12. *Performance Evaluation of Large-scale Data Processing Systems*, 7th International Conference on Applied Informatics, Eger, Hungary, 2007. január 31.

VII. Special talks

1. *Webalkalmazások modellezése*, Tanszéki szeminárium, Debrecen, 2005. április 19.
2. *Webes Információs Rendszerek fejlesztése*, V. Gyires Béla napok, Debrecen, 2005. november 18.

VIII. Software products

1. A Debreceni Egyetem Matematika és Számítástudományok Doktori Iskolájának Webes Információs rendszere, 2004-2005, ~12.000 sor program kód + dokumentáció

2. Debreceni Egyetem Informatikai Kar, Kari honlap oktatás részének elkészítése, üzemeltetése és szerkesztése