Szakdolgozat

Dancsi József

Debrecen

2009

Debreceni Egyetem

Informatikai Kar

Webes felületek generálása UML modellek alapján

Témavezető:

Kollár Lajos

Egyetemi tanársegéd

Készítette:

Dancsi József

Programozó matematikus hallgató

Debrecen

2009

Tartalomjegyzék

1 Bevezetés

Mikor egyetemi éveim során elérkeztem a szakdolgozat témájának kiválasztásához, olyan témát szerettem volna választani, amit a való életben is fel tudok majd használni. Az utóbbi időben az internet rohamos terjedése egyértelmű irányt mutatott az új technológiák fejlődésének. Ezért választottam a webes technológiákhoz kapcsolódó témát.
Az internet korai éveiben a webes tervezés ad hoc jellegű volt, de a fejlődés következtében egyre komplexebb problémák megoldására szükségessé vált a feladatok modellezése. Így alakult ki napjainkra a Model Driven Developement (MDD), Modellvezérelt Fejlesztés. Az MDD alapját modellek, metamodellek és modelltranszformációk képezik. Mindhárom témáról lesz szó ezen dolgozat keretein belül, kiemelten kezelve a modelleket. Ezen dolgozatban egy Unified Modeling Language (röviden UML) alapú modellező rendszer az UWE (UML-based Web Engineering) lehetőségeit kívánom bemutatni egy saját projekt tervezése során.

2 Technológiai háttér bemutatása

2.1 Modellvezérelt fejlesztés

A modellvezérelt tervezés (Model-driven engineering - MDE) a szoftverfejlesztés egy viszonylag új, fejlődésben lévő ága. Legnagyobb jelentősége abban rejlik, hogy az absztrakciót kód szintről modell szintre emeli. Napjaink egyik legígéretesebb fejlesztési elvei közé tartozik. Fő célja, hogy szétválassza a rendszer funkcionalitását és az implementációs kérdéseket. A modellek ilyetén hangsúlyozásával a fejlesztők a problématérre koncentrálhatnak (azaz a modellekre) a konkrét megoldástér helyett.
A modellvezérelt fejlesztés első lépése egy számítás független modell (Computation Independent Model - CIM) megalkotása. Ez a modell írja le a rendszerrel szemben támasztott követelményeket. Ezen követelmények alapján készíthetőek el a platform független modellek

(Platform Independent Models - PIMs). Ezek a modellek a rendszer különböző aspektusait írják le az implementációs kérdések figyelmen kívül hagyásával. A következő lépés platform specifikus modellek (Platform Specific Models - PSMs) elkészítése. Ezek már egy konkrét platformra, implementációs kérdések figyelembevételével készülnek. Ez a lépés modell-transzformációval is történhet, de szükség lehet manuális beavatkozásra. A platform specifikus modellek a kód generálása kiindulópontjai.

A modellvezérelt fejlesztés legismertebb irányzata a Model Driven Architecture (MDA), amit az Object Management Group (OMG) hozott létre. [7] Az OMG nevéhez még sok egyéb jelentős szabvány megalkotása fűződik.

2.2 Az UWE alapjai

Az UML-based Web Engineering (UWE) egy webes alkalmazások fejlesztéséhez készített tervezési elv, amelyet a müncheni Ludwig-Maximilian egyetemen fejlesztenek napjainkban is. Milyen jellemzőkkel bír az UWE?

- Rendelkezik egy modellező nyelvvel webes alkalmazások grafikus reprezentálásához.
- Definiál egy metamodellt az UWE modellező elemeihez.
- Definiál egy fejlesztési eljárást.
- Több CASE tervezési eszközhöz támogatást biztosít. (MagicDraw, ArgoUML)

Az UWE fő jellemzője, hogy UML modelleket használ minden modellhez. Ahol lehetséges tiszta UML használatát javasolja. Miért az UML-t választották?

2.3 Az UWE és az UML kapcsolata

Az UML (Unified Modeling Language) célja, hogy megfelelő eszközöket biztosítson a rendszertervezők, programtervezők, programozók számára szoftver rendszerek tervezéséhez, elemzéséhez és implementálásához, üzleti folyamatok modellezéséhez. [5]

Az UML kezdeti verziója (UML 1) a 90-es években született meg. Alapjául főként az akkor vezető három objektumorientált módszertan (Booch, OMT és OOSE) szolgált. De az alkotói figyelembe vettek egyéb jónak számító gyakorlatokat is modellező nyelvekből,

objektumorientált programozásból, architektúraleíró nyelvekből. A ma frissnek számító 2.2-es verzió az UML 1-hez képest sokkal kifinomultabb. A szintaktikai szabályok és szemantika definiálása sokkal pontosabb, modulárisabb a nyelvi struktúrája, sokkal alkalmasabb nagyléptékű rendszerek modellezésére.
Az UML egyik fő célja, hogy megteremtse azt az álmot, hogy a különböző modellező eszközeink együtt tudjanak működni. Azonban, hogy ez valósággá váljon szemantikai és jelölésbeli megállapodások szükségesek. Az UML a következő követelményeknek tesz eleget:

- Egy MOF alapú metamodellt definiál, ami tartalmazza az UML absztrakt szintaktikáját.
- Minden UML modellezési elv szemantikájának részletes leírása technológia független módon.
- Jelölésre használatos elemek pontos specifikációja.
- Részletes definíciók, mik szükségesek egy UML tervezőeszköznek, hogy kompatibilis legyen a jelenlegi szabvánnyal.

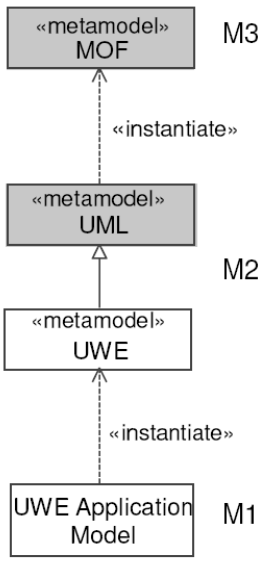
Az UML tervezőrendszerek és az UML alapú szoftverfejlesztő eszközök alapvonása, hogy megfelelő grafikus eszközökkel hatékony segítséget nyújtanak a szoftverfejlesztési folyamat során létrehozandó modellek megalkotásában, majd a modellek alapján képesek különböző célnyelvekre automatikusan forráskódot generálni, ezzel az implementációt is jelentősen gyorsítják és megkönnyítik. Az UML alapjaiban véve egy grafikus eszköz, azaz a modelleket különböző diagramok segítségével jeleníti meg. Az UML azonban több mint egy egyszerű rajzeszköz, egy nyelv is, azaz szintaktikai és szemantikai szabályok összessége. A szintaktikai szabályok a szimbólumrendszert, azok megjelenését, és kapcsolódási módjaikat, a szemantikai szabályok az egyes szimbólumok, és a szimbólumok kapcsolatainak értelmezését definiálják.

Mennyire használható az UML a webes alkalmazások esetében? Nem tartalmaz külön webes elemeket, de lehetővé teszi a nyelv kibővítését az UML profil nevű eszköz segítségével. Az UML profil egy kiterjesztési mechanizmus, ami lehetőséget biztosít új sztereotípiák, címke értékek (tagged values), és OCL feltételek bevezetésére. Létezik olyan profil, ami eszközöket biztosít az UWE mellé.
([4] és [5] források alapján)

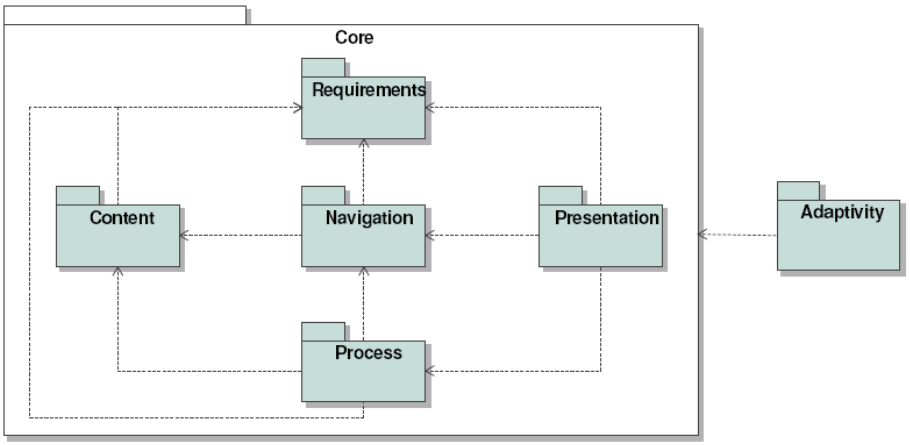
2.4 A Meta Object Facility (MOF)

Az MOF egy OMG szabvány, ami a modellvezérelt tervezéshez kapcsolódik. Az MOF gyökerei visszavezethetőek az UML-hez. Az OMG-nek szüksége volt egy metamodellező architektúrára az UML pontos definiálásához. Az MOF-et egy négyrétegű architektúrának tervezték. Ennek a legfelső szintjét metametamodellnek, vagy M3-nak is nevezik. Ez az M3-modell az a nyelv, amit a metamodellek (M2-modellek) építéséhez használ az MOF. Ezen kettes szintű modellek között kiemelt helyet foglal el az UML metamodellje, ami magát az UML-t definiálja. Ezek a kettes szintű modellek írják le az M1 szintű modellek elemeit, azaz magukat az M1 modelleket. Ilyen M1 modellekre jó példák maguk az UML-ben írt modellek. Az utolsó szintet M0 szintnek, vagy adat szintnek nevezzük. Itt való világbeli objektumokat írunk le. [6]

3 Az UWE metamodellje



A webes modellező rendszerek célja, hogy különválassza a webes rendszereket leíró különböző szerepköröket, mint a tartalom, hipertext, struktúra, prezentáció, folyamatok. Az UWE webspecifikus modellelemeket biztosít ezen szerepkörök modellezéséhez. Ezen modellelemek és a közöttük lévő kapcsolatok leírását tartalmazza a metamodell. Az UWE metamodelljét az UML 2.0 metamodell konzervatív kiterjesztéseként definiálják. A konzervatív ebben az esetben azt jelenti, hogy az UML metamodell elemeket nem módosítják, hanem minden új elem egy eredeti elem leszármazottjaként jön létre. Az új elemekhez további jellemzőket adnak, és további kapcsolatokat definiálnak közöttük. Az új elemek szemantikai egységességét OCL megszorítások használatával biztosítják, az UML jólformázottsági szabályának megfelelően. Az UWE két csúcsszintű csomaggal bővíti ki az UML metamodellt, ezek a Core és Adaptivity. A webalkalmazások szerepköreinek szétválasztását jól tükrözi a Core csomag szerkezete.



1. ábra : Az UWE metamodell áttekintése

3.1 Requirements csomag

A Requirements csomag tartalmazza az UWE kiterjesztését a használati eset modellhez, amivel különválasztja a navigációt az üzleti logikától. További részleteket egyelőre nem közöltek az UWE készítői. A teljesség kedvéért írtam le ezt a pár gondolatot. ([1] 2. fejezet alapján)

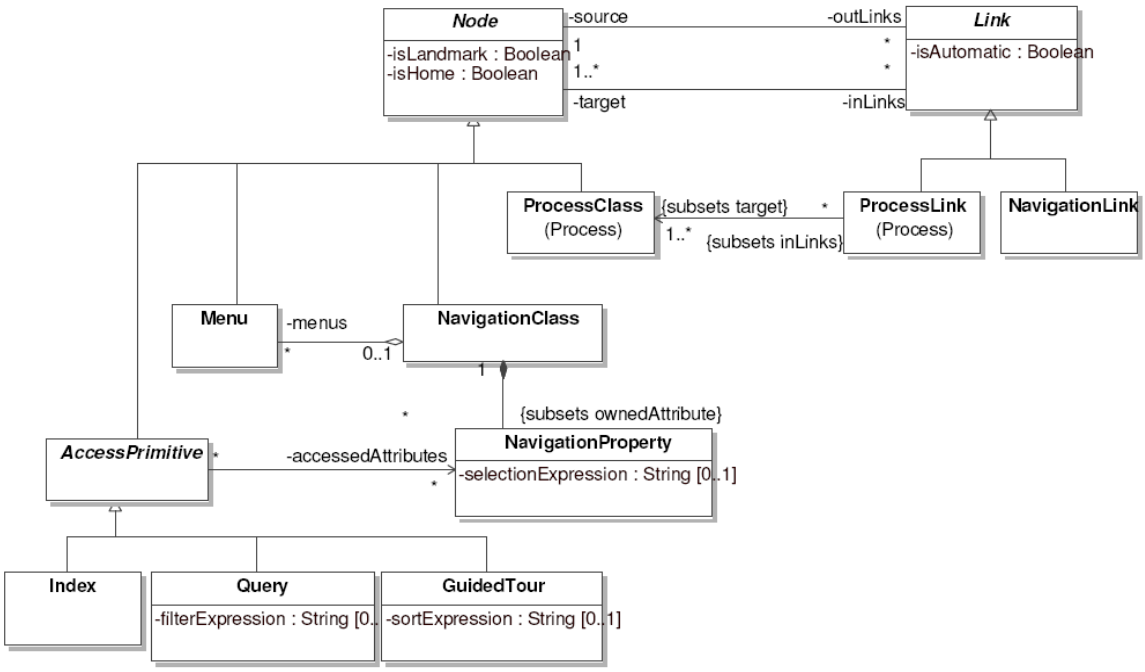
3.2 Content csomag

A tartalom modellezése webalkalmazások és nem webes alkalmazások esetén nem különbözik az UWE-n belül. Ezért szabványos UML elemeket, például osztályokat, asszociációkat és csomagokat használunk struktúra modellezéshez. Továbbá a viselkedés modellezéséhez felhasználhatóak az UML állapotátmenet és szekvencia diagramjai. Speciális webalkalmazásoknál szükség lehet a felhasználók, vagy a környezet modellezésére. Ezen modellekhez használhatunk felhasználó modelleket amik szignifikánsan elválasztják a felhasználókat a tartalmi modelltől. ([1] 3. fejezet alapján)

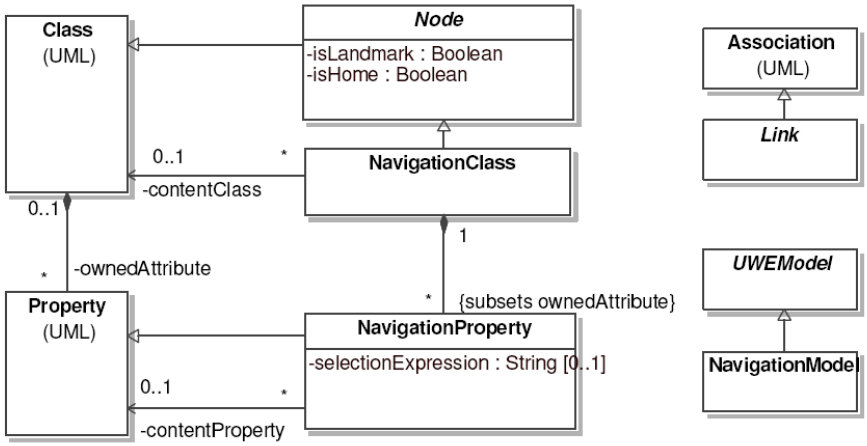
3.3 Navigation csomag

A navigációs metamodell gerincét a Node és Link absztrakt metaosztályok és ezen osztályok közötti asszociációk alkotják. A Node és Link alosztályai a navigációs modell felépítéséhez

szükséges webspecifikus metaosztályok: NavigationClass és ProcessClass, ezekhez kapcsolódik a NavigationLink és ProcessLink, továbbá Menu és a hozzáférési primitívek Index, GuidedTour és Query.



2. ábra : A Navigation csomag



3. ábra : A Navigation csomag kapcsolódása az UML metamodelljéhez

Egy *node*, azaz csomópont bármilyen csomópont lehet a navigációs gráfon. Ez általában azt jeleni, hogy mikor ezen csomópontot elérjük a navigálás során, a felhasználó kap valamilyen információt és opcionálisan lehetősége van egy vagy több (inter)akció végrehajtására. Egy csomópont nem feltétlenül egy webalkalmazás lapját jelöli, de persze ez is előfordul. A weblapokon megjelenő információkkal a prezentációs modell foglalkozik.

Egy *link* egy él a navigációs gráfon, azaz két csomópontot köt össze. Jegyezzük meg, hogy mint ahogy egy node se mindig egy lapot reprezentál, ugyanígy egy link se mindig lapok közötti navigálást jelöl. A prezentációs modell határozza meg, hogy az az információ amit két csomópont és egy link határoz meg, megjelenik-e egyszerre egy lapon, vagy felhasználói interakció szükséges-e ahhoz, hogy egyik csomópontból a másikba navigáljunk.

Egy *navigation class* a hipertext struktúra egy navigálható csomópontját reprezentálja, illetve kapcsolatot biztosít a navigációs modell és tartalmi modell között. Egy *navigation link* egy olyan él, ami bármely két csomópontot összeköthet, kivéve a *process class* típusú csomópontokat. Ha akár a forrás akár a cél csomópont *process class* típusú, akkor *process link* élt használunk.

Ha alternatív navigációs utakat akarunk kezelni, akkor *menu* elemet használhatunk. A *menu* elem nem feltétlenül jelenti azt a menüt, ami egy felhasználói felületen megjelenik, mivel két menüvel összekapcsolt csomópont akár egymás mellett is ábrázolható a prezentációs modellben.

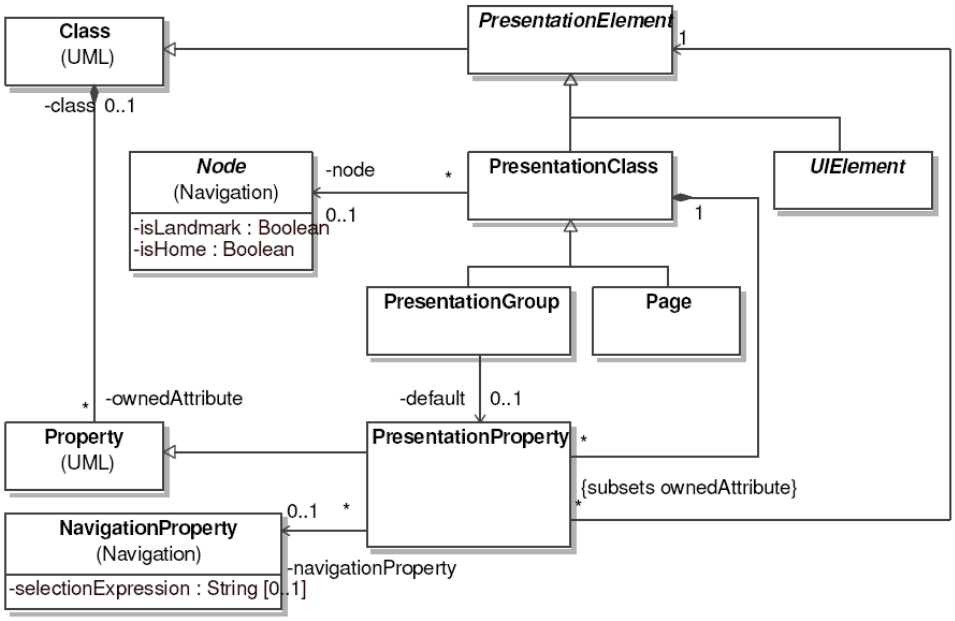
Egy *index* lehetővé teszi egy content class példány kiválasztását egy már korábban keletkezett példányhalmazból. Ez azt jelenti, hogy a navigációs útbeli megelőző elem kontextusából kerül ki a content class példányok halmaza és a felhasználó kiválaszthat egyet ezek közül. A megkapott content class példányok halmaza a bejövő navigation link alapján dől el. Itt három különböző eset lehetséges:

1) A megelőző elem egy *query*. Ekkor a halmaz egyszerűen a lekérdezés eredménye.
2) A megelőző elem egy *navigation class*. Ekkor a halmaz a megfelelő content class attribútumainak kollekciójából kerül ki.
3) A megelőző elem egy *menu*. Ebben az esetben úgy választunk, mintha a menüt megelőző navigation class közvetlenül kapcsolódna az indexünkhöz és a 2. pontban leírt szabályok szerint járunk el.

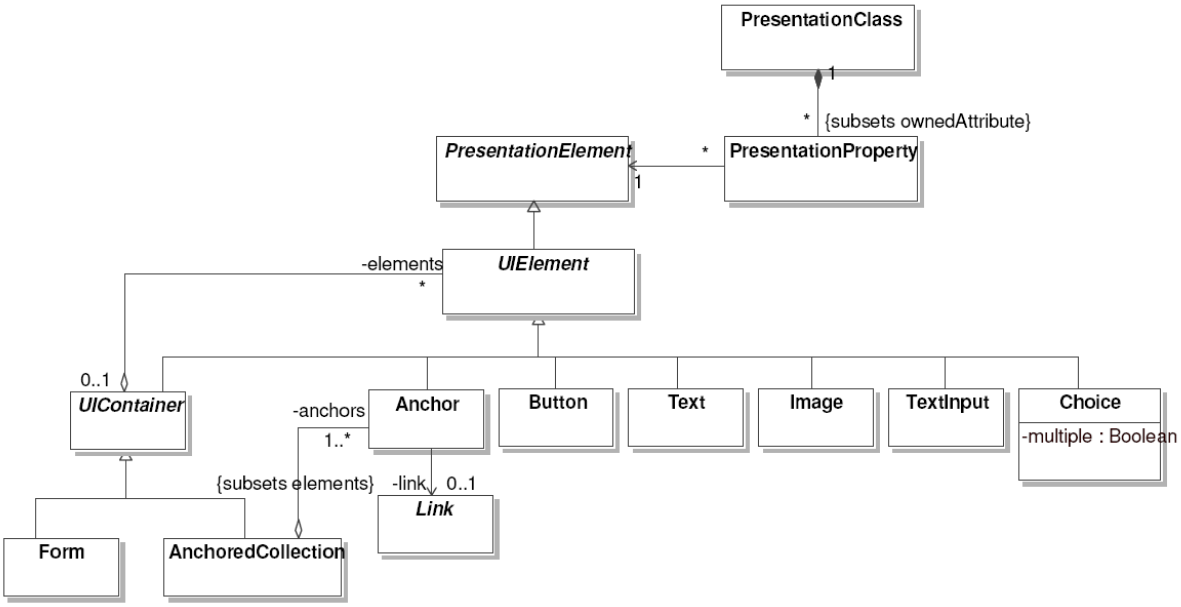
Egy *queryt* tipikusan arra használunk, mint amire a neve is utal, tartalom lekérdezésére egy adatforrásból. Az indexszel ellentétben a query nem content class példányok halmazát kapja meg, hanem valamilyen adatbázis, vagy hasonló adatforrás adatait. Ha a query valamilyen paramétert igényel a lekérdezéshez, akkor ennek a beviteléhez lehetőséget kell biztosítani, és ezt jelölni kell a prezentációs modellben. Ha a query nem igényel paramétereket, akkor automatikusan végrehajtódik, mikor a navigációs gráfban elérjük. Egy tipikus példa a query alkalmazására: kérdezzük le a 10 legnépszerűbb könyvet a könyvadatbázisból.
A *guided tour* elem navigation class példányok egymás utáni feldolgozását biztosítja. Bemenetként content class példányok egy rendezett halmazát kapja meg. Míg kimenete egy navigation class-ra mutató navigation link. A felhasználó előre-hátra navigálhat a bemeneti halmaz elemei között, de egyszerre csak egy elemet választhat ki. A példányok sorrendjét egy szűrőfeltétellel adjuk meg. ([1] 4. fejezet alapján)

3.4 Presentation csomag

A prezentációs modell egy webes alkalmazás felhasználói felületének (user interface - UI) egy absztrakt nézetét jeleníti meg. A modell elvonatkoztat az olyan konkrét elemektől, mint színhasználat, betűtípus, vagy hogy a UI elemek hova kerüljenek pontosan a weblapon. E helyett a felhasználói felület alapvető struktúráját írja le. Például milyen UI elemeket használunk a csomópontok reprezentálásához (szöveg, kép stb.). Ezek az elemek nem azt írják le, hogy milyen konkrét megjelenítési technológiát használunk, hanem hogy milyen funkcionalitás szükséges a UI adott pontján.



4. ábra : A Presentation csomag gerince



5. ábra : Prezentációs elemek

A *PresentationElement* absztrakt szuperosztálya minden prezentációs csomagbeli elemnek. Egy *presentation class* egy navigációs csomóponthoz kapcsolódó prezentációs elemcsoportot definiál. Ha elérjük az említett csomópontot, akkor láthatóvá válik az egész elrendezett elemcsoport.

A *Presentation property*-ket arra használjuk, hogy definiáljuk egy presentation class tartalmát. A tartalmazandó presentation elem nevét használjuk fel a presentation property típusaként.

A *page* elem ugyanazzal a szemantikával rendelkezik, mint a presentation class, azzal a különbséggel, hogy nem szerepelhet egy másik presentation class belsejében. Ez azt jelenti, hogy egy page mindig gyökérelemként fog szerepelni egy összetett prezentációs struktúrában.

A *presentation group* presentation class-ok olyan csoportját definiálja, amik a navigáció függvényében felváltva jelennek meg ugyanazon a területen. Megadhatunk egy alapértelmezett presentation class-t, ami akkor jelenik meg, ha a navigáció során még egyetlen másik, kapcsolódó navigációs csomópontot sem értük el.

Az *UIElement* azon prezentációs elemek absztrakt szuperosztálya, amelyek a tartalom megjelenítéséért vagy szerkesztéséért felelősek. Az UIElementeknek olyan presentation class-ban kell szerepelniük, amihez kapcsolódik navigációs csomópont. Az UIElement alosztályait négy csoportba sorolhatjuk:

- UI konténerek, mint a Form . Ezek tartalmazhatnak egyéb UI elemeket.
- Statikus elemek, mint az Image és Text. Tartalom megjelenítésére szolgálnak.
- Felhasználói adatbevitelt segítő elemek, mint a TextInput és Choice.
- Anchor és Button. Mindkettő változást idéz elő a navigációs vagy folyamat modellben.

A *form* olyan felhasználói felület elemeket foglal egybe, amikkel adatokat adunk meg egy folyamathoz.

Az *AnchoredCollection* egy olyan UI konténer, ami csak anchorokat tartalmazhat. Egy menu, vagy index prezentálásához használhatjuk.

Egy *anchor* lehetővé teszi a felhasználónak, hogy átmenetet generáljon a navigációs modellben. Az UWE presentation modell nem határozza meg, hogy jelenjen meg ez az anchor. HTML kódban például egy anchor elem (<*a*>) vagy egy button is ugyanúgy használható.

A *button* általában egy olyan elem, amivel a felhasználó valamilyen interakciót kezdeményezhet a webes alkalmazással. Leggyakrabban adatbemeneti elemekkel kapcsolatban ezen adatok elküldésére és query vagy valamilyen folyamat végrehajtására használják.

A *text* statikus szöveg megjelenítésére használatos.
Az *image* elem statikus kép megjelenítésére szolgál.
A *TextInput* lehetővé teszi a felhasználónak szöveg bevitelét.
A *choice* egy olyan elem, ami arra szolgál, hogy kiválaszthassunk egy, vagy több elemet egy lehetséges halmazból. Egy webes alkalmazásban sok lehetőség van ezen funkcionalitás konkrét HTML elemekkel való megvalósítására:

- <select> elemmel.
- Rádiógombok csoportjával, ha csak egy elemet kell kiválasztani több közül.
- Checkboxok csoportjával, ha több értéket kell kiválasztani.
- Egyetlen checkboxal, ha a kiválasztott tulajdonság Boolen típusú.

([1] 5. fejezet alapján)

3.5 A Process csomag

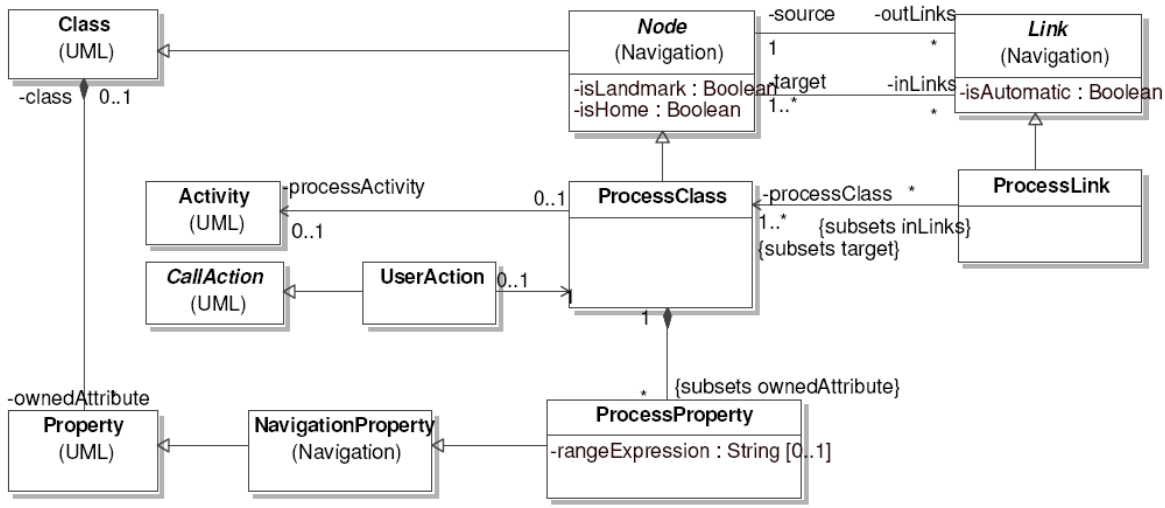
A process csomag olyan modellezési elemeket biztosít, amik lehetővé teszik üzleti folyamatok (business process) UWE webes alkalmazás modellbe történő integrálását. Ez az integrálási folyamat három részből áll:

- Az üzleti folyamatok navigációs modellbe integrálása
  Ehhez két metaosztályt a *ProcessClass*-t és *ProcessLink*-et használjuk. Ezekkel definiálhatjuk, hogyan érünk el egy folyamatot navigáció során és hogyan folytatódik a navigáció a folyamat végeztével.
- A folyamatot támogató felhasználói felület definiálása
  A folyamatok gyakran igényelnek felhasználói felületet az adatok beviteléhez és prezentációjához. Ezeket könnyen definiálhatjuk az UWE prezentációs modell segítségével.
- Működés definiálása
  Az üzleti folyamatok működését UML aktivitás diagramokkal definiáljuk. De a következő szemantikai megszorítások figyelembe vételével:
  - Egy speciális *UserAction* elemet használunk a vezérlési folyam azon pontján, ahol felhasználói adatbevitelre van szükség. A UserAction elemet hozzárendeljük egy process classhoz (ugyanazzal a névvel rendelkeznek), hogy

azonosítani tudjuk, milyen adatokat akarunk bevinni és ehhez milyen presentation class elemet kell megjeleníteni. Ha a felhasználó nyugtázta a bevitt adatokat a vezérlési folyamat folytatódik.

- Gyakori eset, hogy egy csomópontnak inputra van szüksége egy a navigációs gráfon őt megelőző csomóponttól. Ezt a szituációt az UWE egy aktivitás paraméter csomóponttal modellezi.
- Azok az akció csomópontok, amik nem user action csomópontok meghívhatnak műveleteket az input paraméter objektumokon és ezek minden példányán, ami a folyamat alatt keletkezik. Egyéb contextusok elérésének modellezése a modellezőre van bízva.
- Ha olyan eset áll elő, hogy a folyamat során létrehozunk, vagy kiválasztunk egy content class példányt és ezt át akarjuk adni a következő csomópontnak, akkor ennek a modellezéséhez újra a paraméter csomópontokhoz folyamodunk.
- Más folyamatokat beágyazhatunk a sajátunkba úgy, hogy meghívjuk a megfelelő folyamatot használva az UML CallBehaviorActions technikát.

A fent említett modellelemeket és a közöttük lévő kapcsolatokat a lenti ábra szemlélteti.



6. ábra : A Process csomag

A *Process class*-okat üzleti folyamatok navigációs modellbe való integrálására használjuk, továbbá azon adatok definiálására amikkel a felhasználó interakcióba lép az üzleti folyamat során. A navigációs modellben a process classokat *process link*ek segítségével csatolhatjuk navigációs csomópontokhoz. Ez definiálja, hogyan érhetünk el egy folyamatot a navigáció során. Ha a folyamat több olyan lépést tartalmaz, ahol különböző felhasználói felületeket kell használni, akkor minden lépésnél szerepelni kell egy process class-nak a hozzá csatlakozó, korábban már említett user action csomóponttal egyetemben. Egyértelmű, hogy a felhasználói felületek presentation modellekkel kerülnek leírásra. A navigation modellbe csak egy process class kerül be, ez lesz a fő process class és ehhez kell elkészíteni az aktivitás modellt, ami majd leírja az üzleti folyamatot.
A *process property*-k a process classok tulajdonságait írják le. Minden egyes propertyhez tartozik egy UI elem és egy leírás, ami megadja, hogyan kerülnek felhasználásra a felhasználói felületről bekért adatok a folyamat során.

A fenti metamodell leírás az eredeti angol nyelvű specifikáció alapján készült, persze nem szó szerinti fordítás. Több helyen kihagytam nem túl lényeges dolgokat, de igyekeztem minden jelentős részt megemlíteni. Nem törekedtem sebészi pontosságú fordításra, néhol átfogalmaztam, lényegre törőbbé tettem a dolgokat. Mindenképpen szükségesnek éreztem ezt is beemelni a dolgozatba, mivel ezen ismeretek hiányában nagyon nehéz lenne az UWE specifikus fejlesztési részek megértése. ([1] 6. fejezet alapján)

3.6 UWE profil

Az UWE metamodelljét hozzákapcsolták egy UML profilhoz. Az UML profilnak az előnye, hogy szinte minden UML CASE eszköz fel van készítve a használatára. Az UML kiterjesztésének talán a legegyszerűbb módja az új sztereotípiák bevezetése. Ezt az utat követte az UWE is. A sztereotípiák szemantikája megfelel az ugyanezen nevű metamodell elemeknek. A modellek értelmezését nagyban megkönnyíti a következő táblázat ismerete.

UWE stereotype	UML base class	Used in	Icon
«anchor»	class	presentation model	
«anchored collection»	class	presentation model	
«button»	class	presentation model	
«choice»	class	presentation model	
«form»	class	presentation model	
«guided tour»	class	navigation model	
«image»	class	presentation model	
«index»	class	navigation model	
«menu»	class	navigation model	
«navigation class»	class	navigation model	
«navigation link»	association	navigation model	
«navigation property»	property	navigation model	
«page»	class	presentation model	
«presentation class»	class	presentation model	
«presentation group»	class	presentation model	

UWE stereotype	UML base class	Used in	Icon
«presentation property»	property	presentation model	
«process class»	class	navigation/process model	
«process link»	association	navigation model	
«process property»	property	navigation/process model	
«query»	class	navigation model	
«text input»	class	presentation model	
«text»	class	presentation model	
«user action»	action	process model	

7. ábra : UWE sztereotípiák

([1] 7. fejezet alapján)

4 Kezdődik a fejlesztés

A technológiai háttér megismerése után választanom kellett több lehetőség közül, hogy milyen konkrét alkalmazás tervezését végezzem el. A választásom egy egyetemi dokumentum-nyilvántartó rendszerre esett. Úgy éreztem ez áll legközelebb hozzám, el tudtam képzelni, milyen lenne ha a valóságban is használnék egy ilyen rendszert.
A tervezést az UWE készítői által ajánlott MagicDraw nevű tervezőszoftver segítségével végeztem. A szoftver használata során szerzet tapasztalataimról az összegzés témakörön belül számolok be.

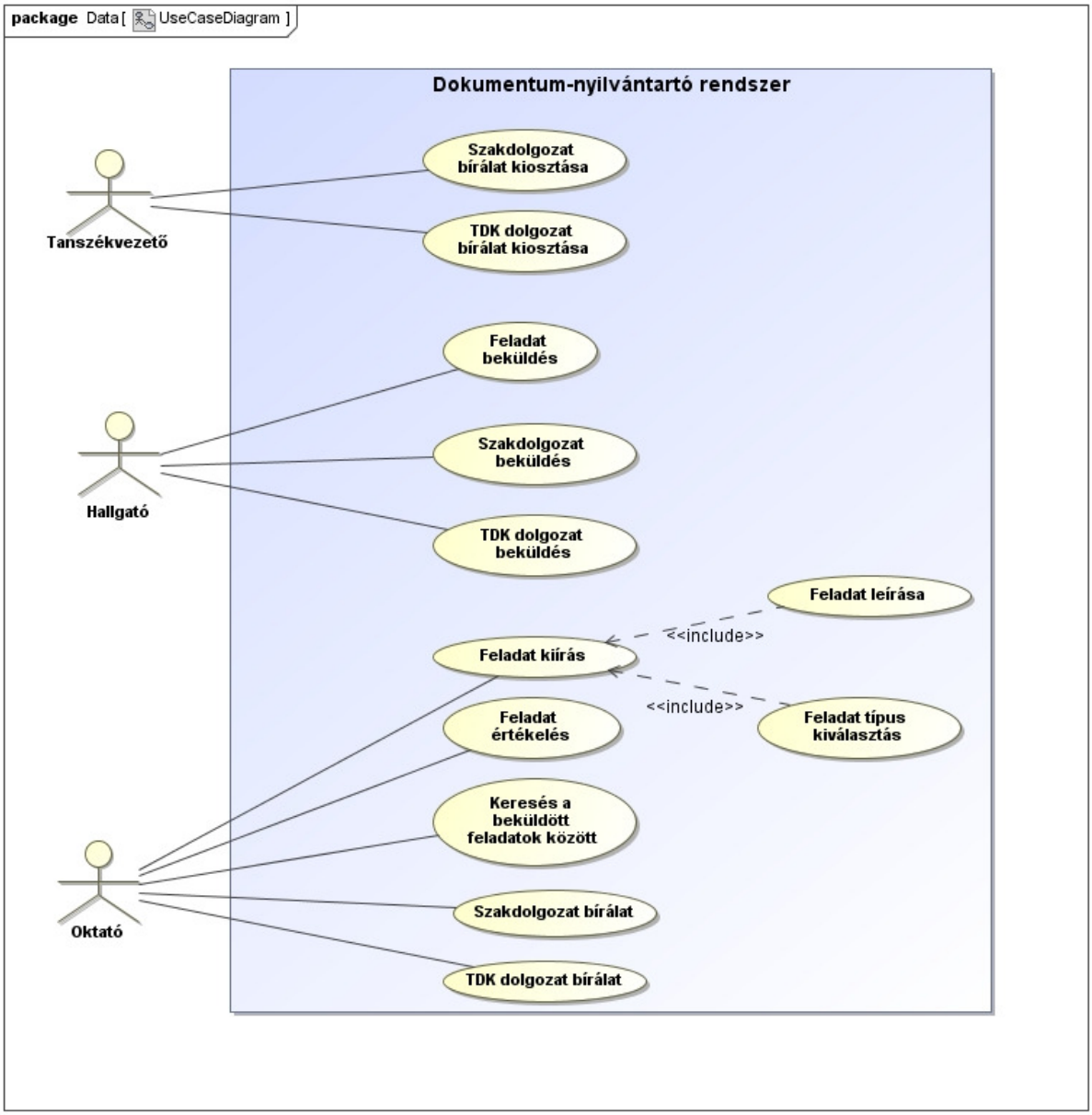
4.1 Követelmények modellezése

A fejlesztés megkezdése előtt fontos lépés a követelmények felmérése. Ilyenkor a rendszerrel szemben támasztott legfontosabb követelményeket kell felkutatnunk. A kevésbé lényeges részeket elhanyagolhatjuk. Ha mégis kifelejtenénk valamit, akkor a későbbiek során rengeteg plusz munkánkba kerülhet a kezdeti hiba korrigálása. Az eredményeket egy lentebb látható használati eset diagramon (*8. ábra*) összegzem.
A nyilvántartó rendszert három fő felhasználócsoport fogja használni:

- Hallgatók
- Oktatók
- Tanszékvezetők

Ők, mint aktorok szerepelnek a modellben. Az egyszerűség kedvéért tételezzük fel, hogy a rendszer rendelkezik egy háttéradatbázissal, ahol már szerepelnek a konkrét, regisztrált aktor objektumaink. Ilyen eset lehet például, ha a rendszerünket a jelenleg használt tanulmányi rendszerhez, a Neptunhoz kapcsoljuk hozzá.
A tervezés szempontjából a legegyszerűbb dolgunk a tanszékvezető aktorunkkal van, neki csak két fontos lehetőséget biztosít a rendszer, szakdolgozatok és TDK dolgozatok bírálatainak kiosztását. A hallgatók a rendszer segítségével küldhetnek be szakdolgozatot, TDK dolgozatot és az oktatók által kiosztott feladatok megoldásait. Rendszerünk az oktatók számára biztosítja a legtöbb lehetőséget. Ha van számukra kiosztott szakdolgozat, vagy TDK dolgozat, annak elvégezhetik a bírálatát. Fontos része a rendszernek a hallgatóknak szóló

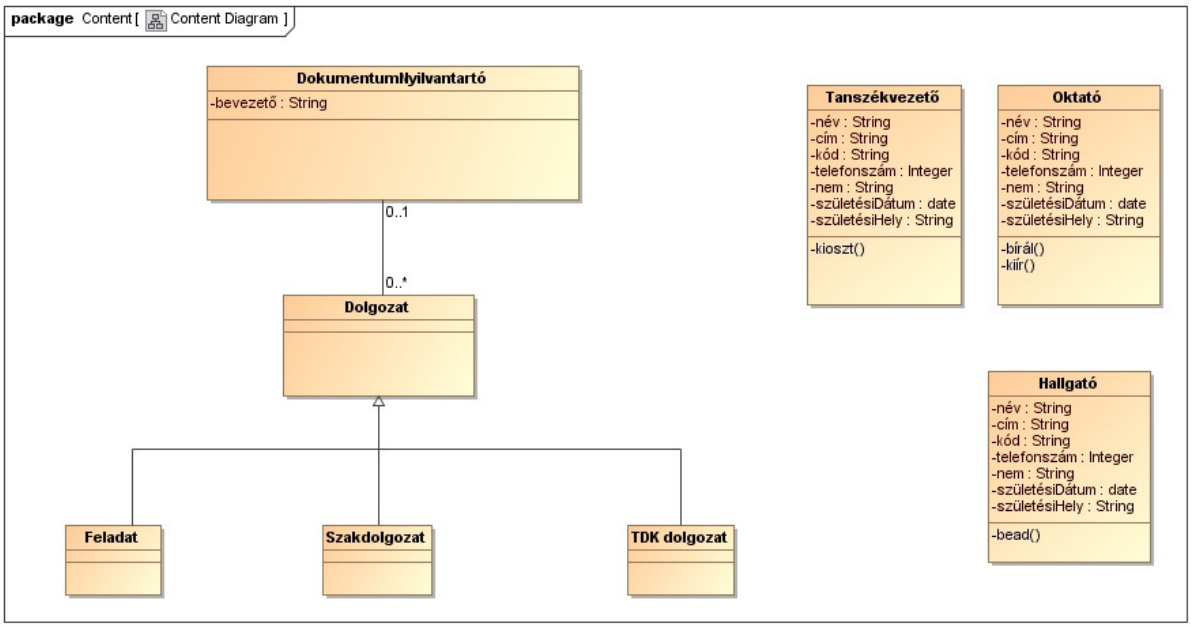
feladatok kiírása, mivel az oktatók ezt az opciót fogják a legtöbbet használni. Az oktatónak ki kell választania a feladat típusát: jegyre értékel, vagy elfogad, nem fogad el típusú e, majd meg kell adnia a feladat leírását. Ha vannak már elkészült, beadott feladatok azokat értékelheti. Osztályozhatja, elfogadhatja, vagy ha nem felelt meg visszaküldheti indoklással karöltve. A magas hallgatói létszám miatt természetes, hogy keresési, válogatási lehetőséget kell biztosítani a beküldött feladatok között.



8. ábra : Használati eset diagram

4.2 Tartalommodell

Ezen modell célja, hogy különválassza a tárolt, perzisztens tartalmat a hypertext / navigációs struktúrától és a prezentációs résztől. A modellezéshez egyszerű UML osztálydiagramokat használunk. Gondoljuk végig milyen osztályokra is lesz szükségünk? Először is feltétlen kelleni fog egy *DokumentumNyilvantarto* osztály, amiben a beküldött *Dolgozat* típusú dokumentumokat tároljuk valamilyen formában, ez tipikusan valamilyen adatbázis lehet. A Dolgozat osztály egy általánosítás, mivel a konkrét esetekben nem egy általános dolgozat, hanem különböző speciális dokumentumok kerülnek elküldésre, ezért szükséges a további három osztály: *Feladat*, *Szakdolgozat* és *TDK dolgozat* definiálása. Mindhárom osztály érthető okokból a Dolgozat leszármazottja. A további három osztály, *Tanszékvezető*, *Oktató*, *Hallgató* nemcsak a tárolt felhasználói adatok reprezentációja miatt szükséges, hanem mivel ezek tartalmazzák az alkalmazás működése szempontjából kiemelt fontosságú *kioszt()*, *bírál()*, *kiír(), bead()* metódusokat.



9. ábra : Tartalom diagram

4.3 Navigációs modell

Egy weboldalakat modellező rendszertől elvárható, hogy megmondja, hogy kapcsolódnak egymáshoz az oldalak. Ennek a modellezéséhez olyan diagramra van szükségünk, ami csomópontokból és linkekből áll. Ezt a szerepet tölti be az UWE rendszerében a navigációs diagram. A legegyszerűbben úgy juthatunk navigációs diagramhoz, ha alkalmazzuk a tartalom-navigációs modelltranszformációt. Ez a MagicDraw eszközben már támogatott, pár kattintással elérhető opció. Eredményként kapjuk a fenti tartalom diagramon szereplő osztályokat <<*navigationClass*>> sztereotípiával ellátva. A transzformáció eredményeként több csomópontot is kaphatunk, mint amire ténylegesen szükségünk lesz. A tervezés során kiderült, hogy nem lesz szükségünk a Tanszékvezető navigációs osztályra, mivel a felvázolt tevékenységekben csak mint felhasználó játszik szerepet, ezért ezt az osztályt töröltem a diagramról. Természetesen ennyi elem nem elég egy viszonylag komplex rendszer navigációjának modellezéséhez. Az UWE metamodelljében található navigation csomag elemeit használhatjuk a további modellezés során. Ezen modellelemeket általában <<*navigationLink*>> élekkel kötjük össze.

A felhasználói szerepkörök szétválasztásához, amint az a lenti ábrán (10. ábra) látható, csomagokat használtam. Egy másik lehetőség lett volna az, ha minden egyes felhasználóhoz külön-külön készítek egy navigációs modellt, de az előbbi jobb választásnak tűnt számomra, mivel a modell így jobban szemlélteti a rendszer egészét. Nagyobb, komplexebb rendszereknél belátható, hogy az utóbbi lehetőség az ésszerűbb választás a modell bonyolultsága miatt.

Az UWE metamodell process csomagjánál leírtak szerint szükséges az üzleti folyamatok navigációs modellbe integrálása. Ezért vannak a navigációs modellben <<*processClass*>> csomópontok és <<*processLink*>> élek is. A kétféle éltípus jobb elkülönítése miatt a <<*navigationLink*>> jelölést elhagyjuk. Az irányított élek egyirányú navigációt tesznek lehetővé. Ahol nem irányított az él, ott lehetséges a visszalépés is.

A kezdőlapunk minden felhasználó esetén a *DokumentumNyilvántartó*. Ezt jelöljük is meg egy *{isHome}* címkeértékkel. Ez egy olyan lap, amit az alkalmazáson belül mindenhonnan elérhetünk.

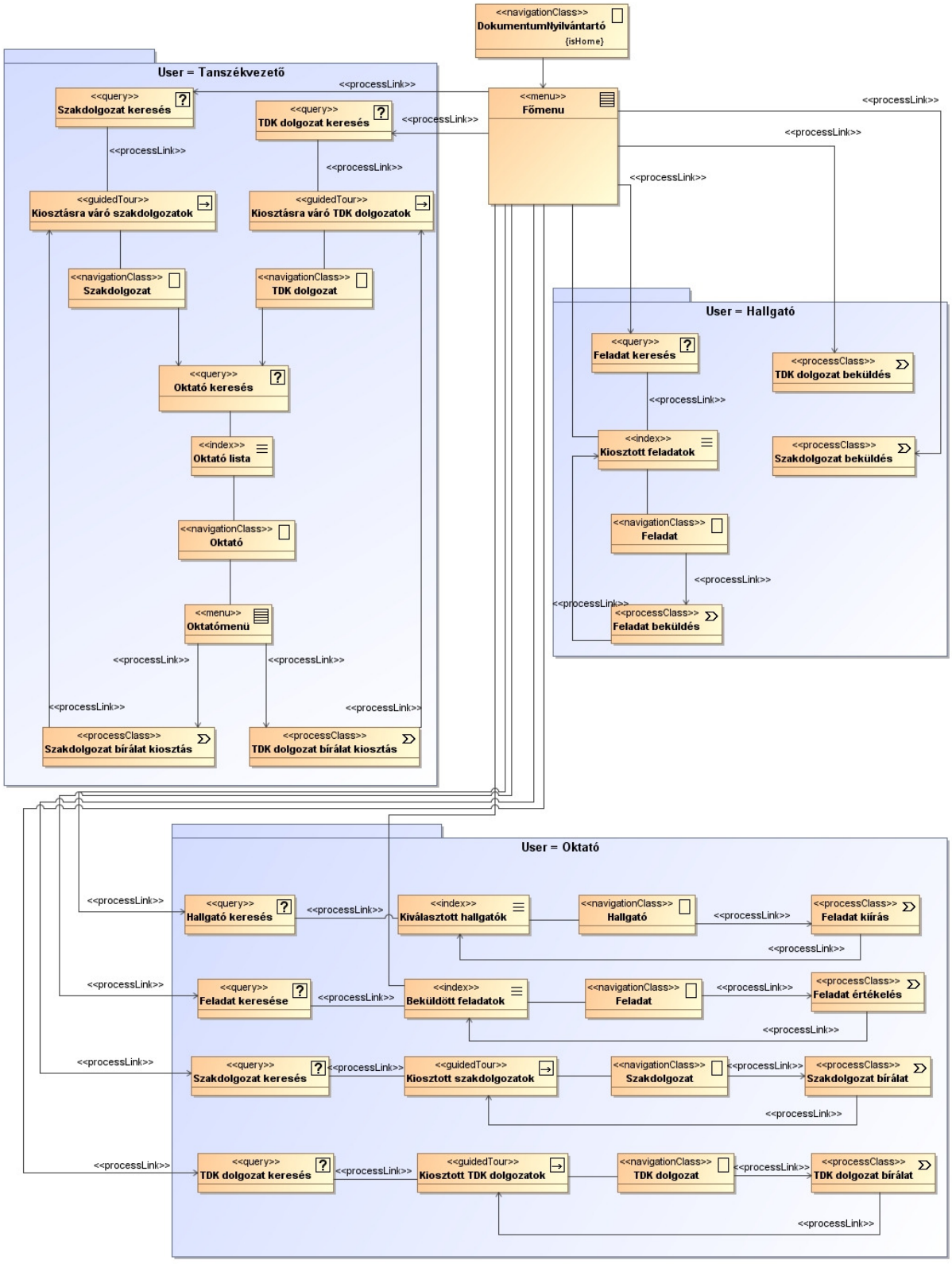
Egy olyan alkalmazást szeretnénk, ahol elérhetőek azok a műveletek, amiket a használati eset diagramnál leírtunk. Ha egy csomópontból több más csomópontot akarunk elérni, akkor

használhatjuk a <<*menu*>> sztereotípiát. Hozzunk létre egy ilyet és nevezzük el *Főmenünek*. Mivel a kezdőlapunk tartalmazza a Főmenüt, ezért az is elérhető bárhonnan.

A legegyszerűbb navigációval a Hallgató felhasználók rendelkeznek. A Főmenüből közvetlenül elérhetik a *Szakdolgozat beküldés* és *TDK dolgozat beküldés* műveleteket. A Feladat beküldés művelet eléréséhez előbb ki kell választaniuk egy konkrét *Feladat*ot, akár a kezdőlapról, ahonnan az összes kiosztott feladat elérhető (*Kiosztott feladatok*), vagy kereséssel (*Feladat keresés*). A feladat listához a navigációs csomag <<*index*>> elemét, a kereséshez a <<*query*>> elemet használhatjuk. A Feladat beküldés végrehajtása után a rendszer megjeleníti a maradék feladatok listáját.

Kicsivel bonyolultabb a helyzet a Tanszékvezető felhasználók esetében. Hogy elérjük a két lehetséges tevékenységet, a *Szakdolgozat bírálat kiosztás*t és a *TDK dolgozat bírálat kiosztás*t, két kiválasztást kell végrehajtanunk. Egyrészt szükséges egy Szakdolgozat, vagy TDK dolgozat kiválasztása, másrészt ki kell választani melyik oktatónak kívánjuk kiosztani az adott dolgozatot. A kereséseknél azt a mechanizmust használjuk ki, hogy ha nem adunk meg keresési feltételt, akkor az összes dolgozat listázásra kerül. Ez az állítás igaz az összes keresésre a rendszeren belül. A keresési eredmények listázása a korábbi index elemmel ellentétben itt <<*guidedTour*>> elem segítségével történik. A guidedTour jelentősége abban áll, hogy nem egy listából választhatjuk ki a kívánt listaelemet, hanem egyenként végignézhetjük a dolgozatokat előre-hátra lépkedve. Ha kiválasztottuk a nekünk szimpatikus dolgozatot, akkor a következő lépés a bíráló oktató kiválasztása. Ez a lépés hasonlóan valósul meg, mint a korábbi feladat kiválasztás a hallgatók esetében. Ez a keresés, listázás, kiválasztás modellezési minta sok helyen megfigyelhető a lenti diagramon. Mivel két fajta dolgozatunk van és mindkettőhöz kell oktatót is választanunk, ezért az oktató kiválasztásához vezető lépéssort csak egyszer modellezem. Ennek később hátulütője is lesz, mivel szükségünk lesz egy újabb menüre a kiválasztás végén (*Oktató menü*). Ez „csak” egy technikai elem, mivel nem indulhat ki több navigációs él csak és kizárólag menü elemből. A bírálat kiosztása után a rendszer visszalépteti a felhasználót a kiosztásra váró dolgozatokhoz.

Az Oktató felhasználó részletes ismertetését már nem teszem meg, mivel látható, hogy a fentebb leírt minták szerint történik a modellezés. Annyit említenék meg, hogy a többféle dokumentumok listái közül azért a *Beküldött feladatok* érhető el a Főmenüből és látható a kezdőlapon, mivel valószínűleg ez lesz a leggyakrabban használt lista.

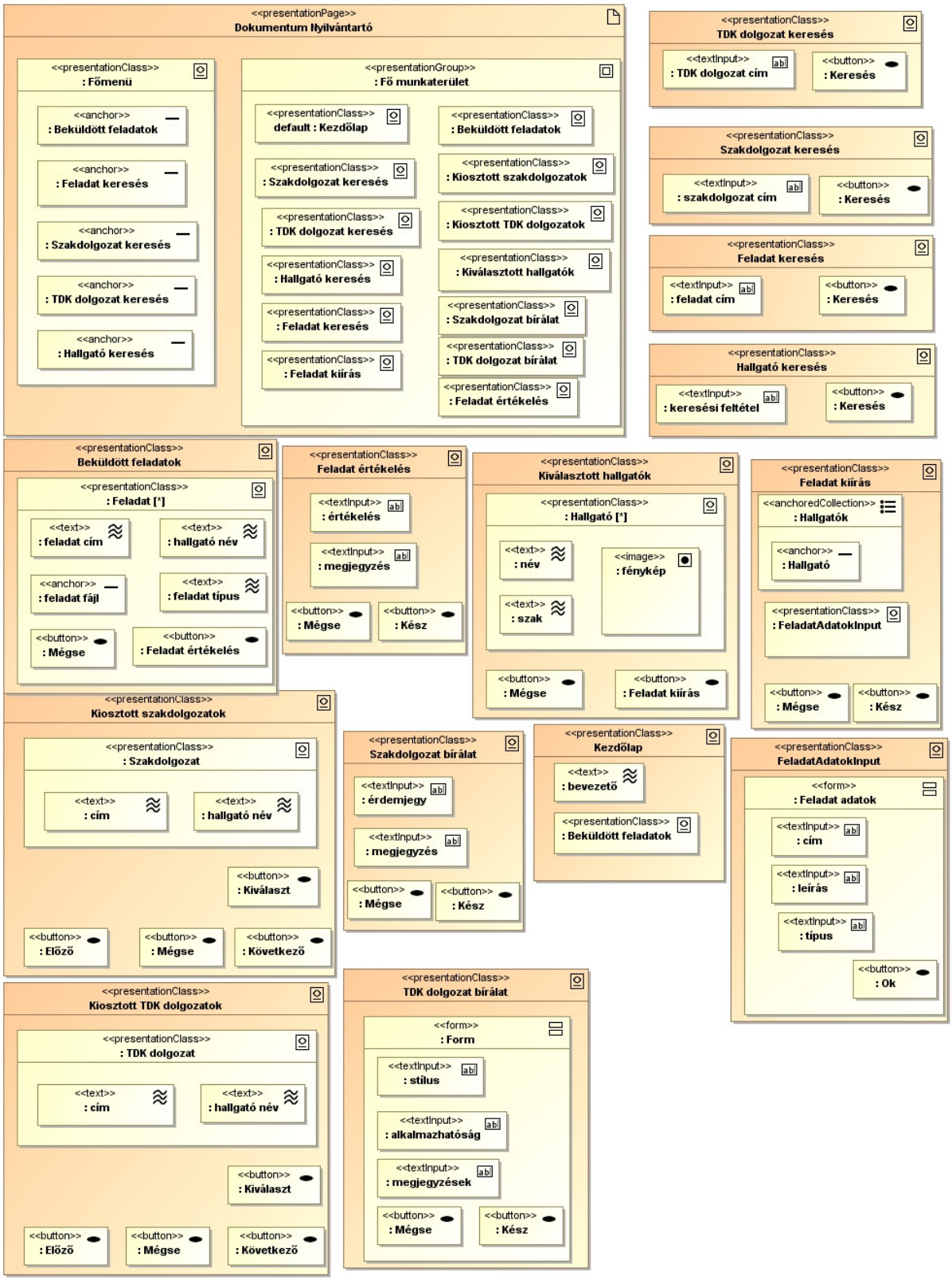


10. ábra : Navigációs diagram

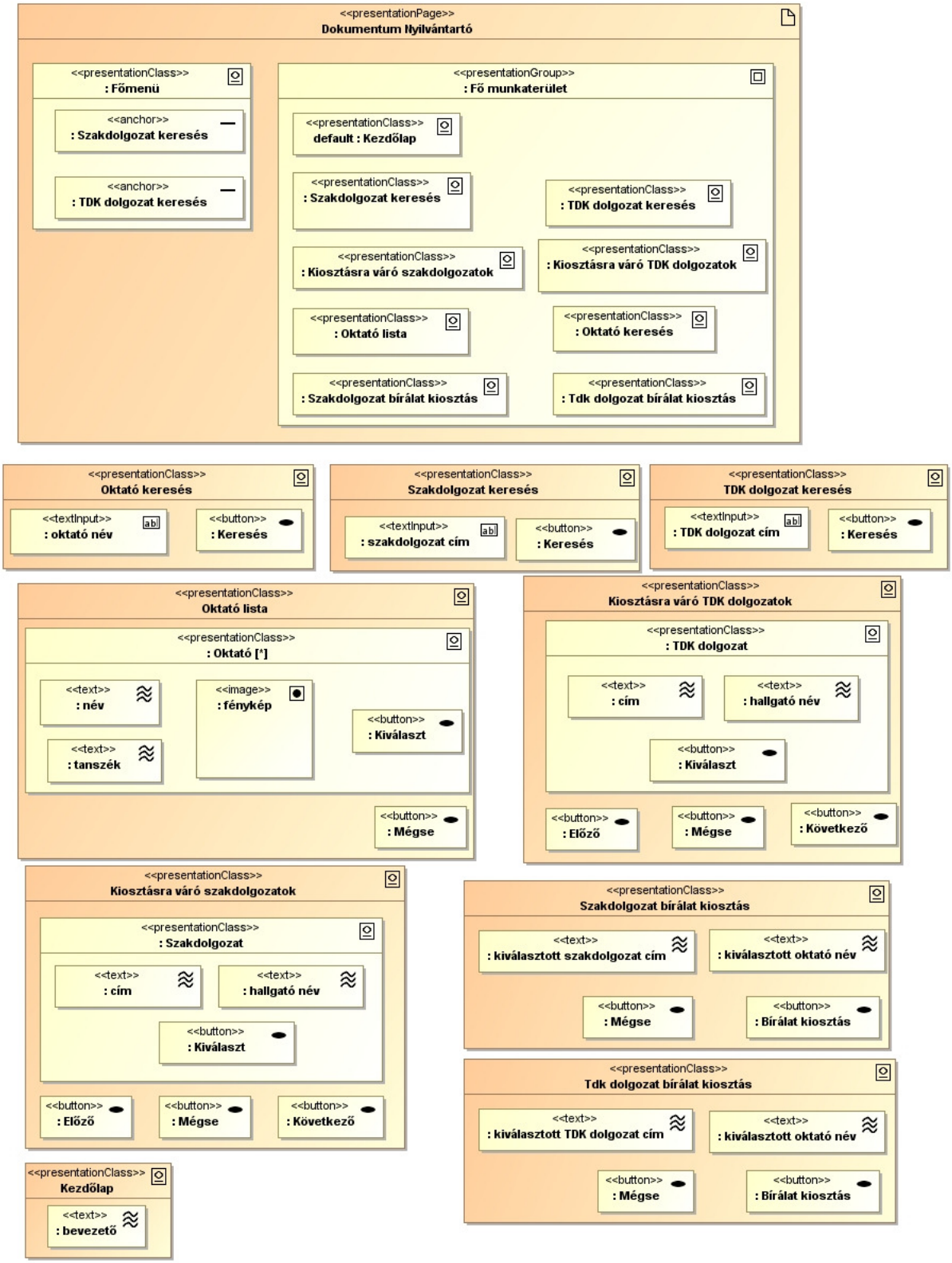
4.4 Prezentációs modell

A navigációs modell nem definiálja, hogy melyik navigációs, vagy process osztály melyik weboldalon jelenik meg. Ezt az információt a prezentációs modell segítségével írhatjuk le. A modellünk a webes alkalmazásunk felhasználói felületének egy absztrakt nézetét adja meg. Elvonatkoztatunk a konkrétumoktól, mint a színhasználat, betűtípus és hogy melyik elem pontosan hol helyezkedjen el. E helyett azt írjuk le, hogy melyik navigációs csomópontot, milyen UI elemek felhasználásával jelenítsük meg. Ezen modell előnye, hogy a döntéshozók még a valódi implementálás előtt dönthetnek a megjelenítés részleteiről.

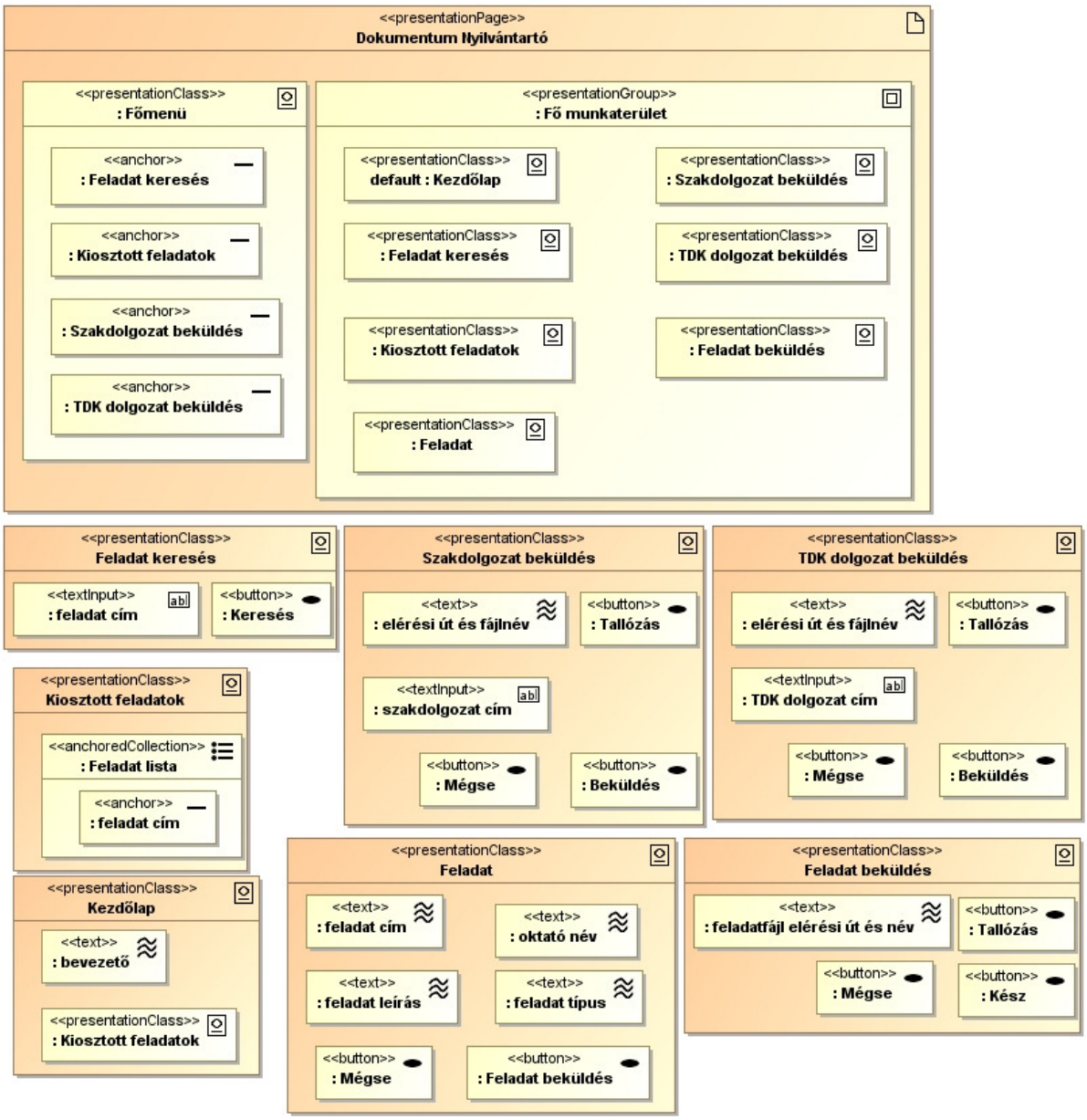
A modellünk alapjául a navigációs modell szolgál. A navigáció során, amint elérünk egy csomópontot a csomópont nevével megegyező <<*presentationClass*>> lap kerül megjelenítésre. A weboldalunkon általában több navigációs csomópont tartalma jelenik meg egyszerre. Ennek a modellezésére használjuk a <<*page*>> elemeket. Ezen elem nem szerepelhet más presentationClass belsejében és tartalmazhat több presentationClass és presentationGroup elemet. A <<*presentationGroup*>> elem speciálisnak számít. A benne szereplő presentationClass lapok közül, mindig csak egy jelenik meg, a szerint, hogy melyiket értük el a navigáció során. Megadható alapértelmezett (default) elem, ami addig jelenik meg, amíg egyetlen navigációs elemet sem értünk el a felkínált alternatívák közül. Ilyen presentationGroup a modellünkben a *Fő munkaterület.*
A tartalommodell DokumentumNyilvántartó osztályában definiált bevezető attribútumot most használjuk fel a Kezdőlap elemen, ami addig látható, amíg nem történik egyéb navigáció a rendszerben. Üdvözlőszöveg, rövid leírás az elérhető funkciókról, figyelmeztetések, változások megjelenítésére használhatjuk.
([8] 7.2.5 fejezet alapján)



11. ábra : Prezentációs diagram – Oktató



12. ábra : Prezentációs diagram – Tanszékvezető

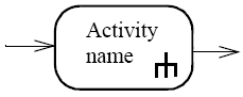


13. ábra : Prezentációs diagram – Hallgató

4.5 Folyamatmodell

A folyamatok modellezése már a navigációs modell véglegesítésekor megkezdődött, mivel már ott megjelentek a <<processClass>> csomópontok. Ezt a lépést az UWE metamodellje „üzleti folyamatok navigációs modellbe integrálásának” nevezi. Ezt a lépést bővebben a navigációs modell leírásakor ismertettem.

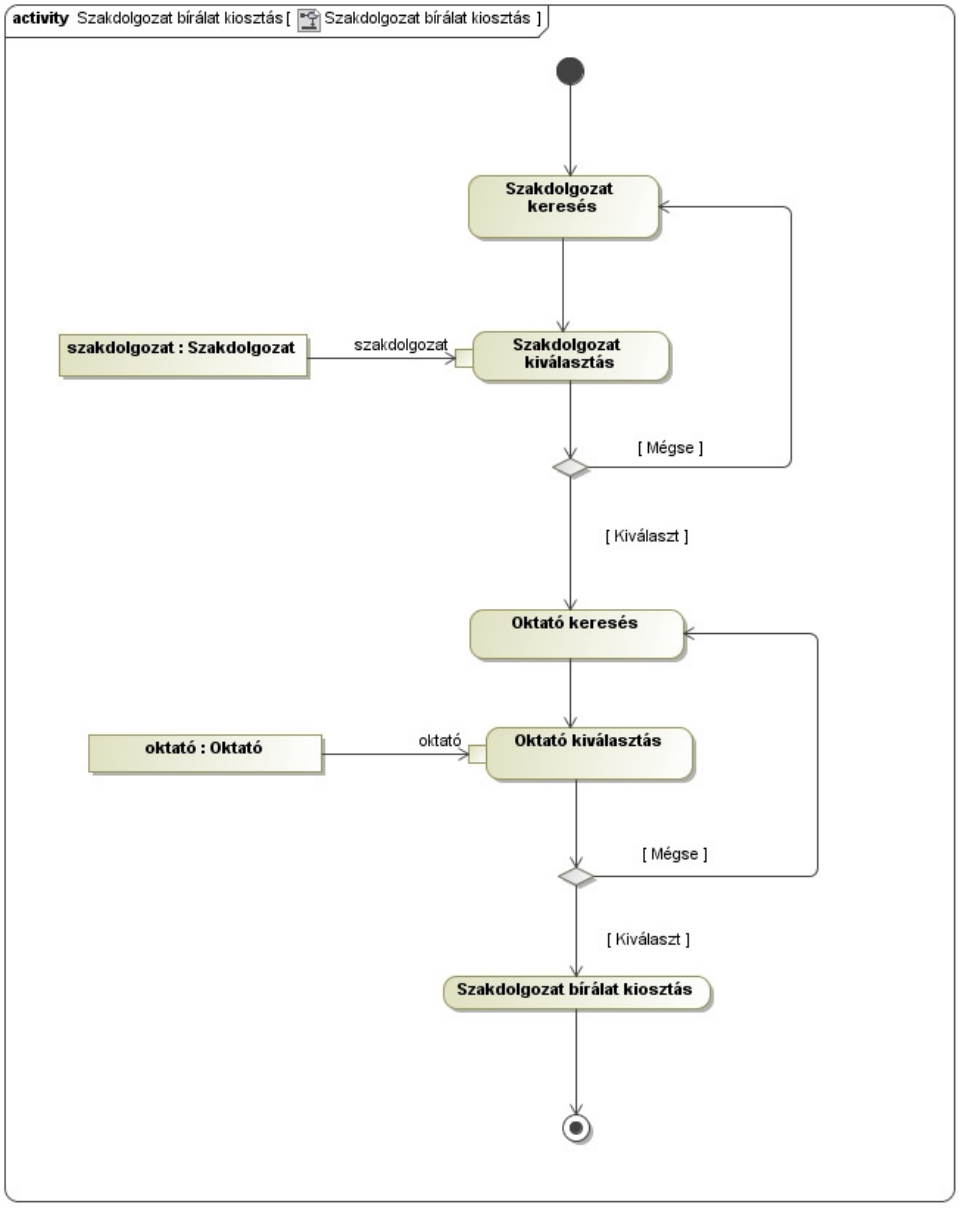
A következő lépés minden egyes folyamat konkrét folyamat modelljének (process flow) elkészítése. A folyamatok egyszerűsége, és hasonlósága miatt csak két példát szerepeltetek a lentebbi ábrákon (14. és 15. ábra). Az UWE ezekhez a modellekhez sima UML aktivitás diagramok használatát javasolja. Ez azért is előnyös, mert a fejlesztőnek nem kell újabb modellezési eljárást elsajátítani, hanem az elterjedt UML szabvány használható.
Bonyolult folyamatok esetében a könnyebb átláthatóság miatt lehetőség van beágyazott aktivitás diagramok használatára az UML CallBehaviorAction technika használatával. Bonyolult hangzása ellenére ez egy nagyon egyszerű technika. Ha van egy összefüggő folyamatunk, azt jelölhetjük egyetlen aktivitással és egy szimbólummal, majd egy másik aktivitás diagramon, aminek ugyanaz lesz a neve, mint a fenti aktivitásnak kifejthetjük a működését.



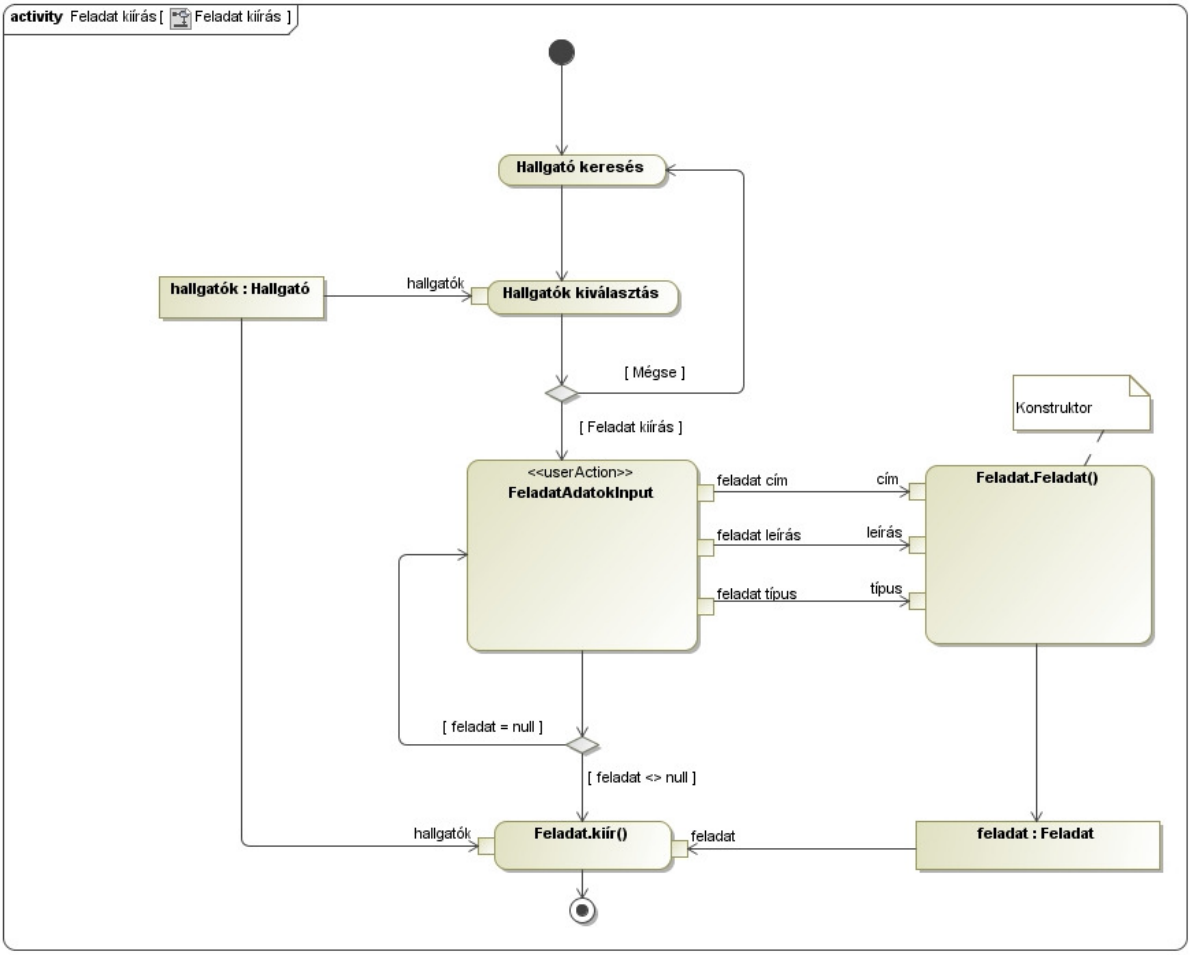
([5] 12.3.14 fejezet alapján)

Az UWE fejlesztési elve szerint opcionális modellezési lépés a folyamat struktúra modell elkészítése. Ez a modell a rendszer összes folyamata közötti kapcsolatokat modellezi. Opcionalitását az indokolja, hogy egyszerű rendszerek esetében nincs, vagy csak nagyon kevés a kapcsolat az elkülönülő folyamatok között. Ez a helyzet meglátásom szerint jelen rendszer esetében is.

A második modellezett folyamat (Feladat kiírás) tervezése során használtam egy <<*userAction*>> aktivitást, mivel felhasználói adatbevitelre volt szükség. A hozzá tartozó prezentációs osztály szerepel a megfelelő ábrán (*FeladatAdatokInput* 11.ábra : Prezentációs diagram – Oktató).



14. ábra : Szakdolgozat bírálat kiosztás aktivitás diagram



15. ábra : Feladat kiírás aktivitás diagram

5 Összefoglalás

5.1 MagicDraw

A MagicDraw egyik nagy előnye, hogy mindenhol futtatható, ahol rendelkezésre áll Java 5 vagy 6. Azt állítják a készítői, hogy ez a leggyorsabb eszköz modellek készítéséhez. Ezt jómagam is tapasztaltam, mikor pár hét használat után, már szinte maguktól épültek a modelljeim. A különböző modellelemek kényelmesen kiválaszthatók némi gyakorlás után, a nevek beírásakor legördülő menüből automatikus kiegészítés érhető el a már meglévő modellelemek nevei alapján. A modellelemek megjelenése nagy mértékben testreszabható, színt választhatunk, eltüntethetjük az attribútumokat és a metódusokat stb. A program indításakor a nyitólapon egy kattintással megnyithatóak a nemrég használt projektek. Egy modellelem kijelölésekor a jobb alsó sarokban felbukkanó jelre kattintva beállíthatjuk az elem optimális méretét. A fent felsorolt kényelmi lehetőségek mindegyike nagyon hasznos volt számomra a fejlesztés során, csak ajánlani tudom mindenkinek ezt a fejlesztőeszközt.

5.2 Az UWE értékelése

A szakdolgozat témája webes felületek automatikus generálása UML modelleke alapján. Ennek ellenére sehol nem generáltunk kódot a dolgozat során. Sajnos ki kell jelentsük, hogy jelenleg az UWE nem képes a modelljeinkből automatikusan kódot generálni. A fejlesztése a mai napig folyamatos, és már letölthető egy Eclipse fejlesztőeszközhöz készített plugincsomag, ami a fejlesztők állítása szerint JavaServer Faces (JSF) platformra képes automatikusan kódot generálni. A jelenleg elérhető adatok szerint a mostani modelleket jelentősen ki kell bővíteni, hogy lehetővé váljon a kódgenerálás. Szükségessé válik egy script nyelv az Object-Graph Navigation Language (OGNL) erőteljes használata a modellezéshez.
Az eddig elkészített modellek a Modellvezérelt fejlesztés témakörön belül említett platform független modellek (PIMs). Az OGNL használatával kiegészített jövőbeli modellek pedig a platform specifikus modellek (PSMs) lehetnek majd.
A jövőbeli kilátásokat figyelembe véve úgy gondolom, hogy az UWE egy nagyon ígéretes fejlesztési elv. A háttere messzemenőkig kidolgozott és igény szerint bővíthető. A tervezői

figyelik a legújabb modellezési és tervezési technikákat, módszereket és felhasználják a legjobbnak bizonyulókat. Ha fogadnom kellene, hogy melyik lesz az a mai módszer ami esetleg egy jövőbeli Egységes Web Modellező Nyelv („Unified Web Modeling Language”) előfutára lehet, akkor jelenlegi tudásom szerint az UWE-re fogadnék.

5.3 Jövőbeli tervek, fejleszthetőség

A dolgozat célja az UWE lehetőségeinek bemutatása volt. Úgy gondolom, hogy a jelenlegi tudásomnak megfelelő szinten képes voltam ezt megvalósítani. Nem véletlen, hogy egy ilyen egyszerű rendszert mutattam be, nem az volt a cél, hogy oldalakon keresztül diagramok tengerét mutassam be – bár lehet sokaknak így is annak tűnik – , hanem, hogy az olvasó értse is ami a diagramokon szerepel. A rendszert tovább lehet fejleszteni olyan szintre, ahol akár ténylegesen használhatóvá válhat dokumentumok nyilvántartására. Ha tudjuk milyen környezetbe kell implementálnunk, akkor megvalósítható a belépés, kilépés, regisztráció. A navigációs és prezentációs részeket mindenképp át kell dolgozni, hogy kényelmesebb legyen a felhasználóknak. További dokumentumok, például kérvények kezelése minden további nélkül implementálható a már meglévők mintájára. Az UWE fejlődésével, ha mi is követjük a változásokat akár a kódgenerálás is valósággá válhat.

Irodalomjegyzék:

[1] UWE Metamodel and Profile User Guide and Reference
http://uwe.pst.ifi.lmu.de/download/UWE-Metamodel-Reference.pdf

[2] Nora Koch: Model-Driven Web Engineering: UWE Approach
http://uwe.pst.ifi.lmu.de/publications/MDWE-UWE_URJC_280508.pdf

[3] UWE Tutorial
http://uwe.pst.ifi.lmu.de/teachingTutorial.html

[4] Andreas Kraus, Alexander Knapp, Nora Koch: Model-Driven Generation of Web Applications in UWE
http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-261/paper03.pdf

[5] OMG Unified Modeling Language (OMG UML), Superstructure
http://www.omg.org/spec/UML/2.2

[6] Meta-Object Facility
http://en.wikipedia.org/wiki/Meta-Object_Facility

[7] Model Driven Architecture (MDA) FAQ…
http://www.omg.org/mda/faq_mda.htm

[8] Web Engineering: Modelling and Implementing Web Applications
(Publisher: Springer London 2008)