VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO PODPORU PROCESU VÝVOJE PODLE RYSŮ

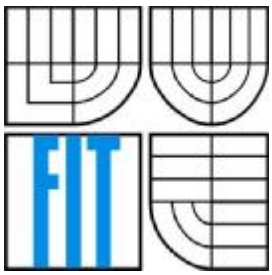
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVLÍNA TICHÁ

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO PODPORU PROCESU VÝVOJE PODLE RYSŮ

INFORMATION SYSTEM SUPPORTING FEATURE DRIVEN DEVELOPMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVLÍNA TICHÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. MAREK RYCHLÝ

BRNO 2007

Licenční smlouva

Licenční smlouva v kompletním znění je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Výňatek z licenční smlouvy:

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ■ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   z důvodu utajení v něm obsažených informací.
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Zadání diplomové práce

Řešitel: **Tichá Pavlína, Bc.**
Obor: Informační systémy
Téma: **Informační systém pro podporu procesu vývoje podle rysů**

Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s agilním procesem vývoje softwarových projektů podle rysů (metodika Feature-Driven Development, FDD).
2. Analyzujte proces FDD. Zaměřte se na pět základních podprocesů a jejich průběh, produkty podprocesů, způsoby vyjádření a sledování rysu, plánování podle rysů, formování týmů a zodpovědností (princip vlastnictví třídy), prezentaci stavu procesu, atd.
3. Navrhněte informační systém pro podporu procesu FDD, který umožní řízení softwarového projektu podle FDD.
4. Implementujte navržený informační systém jako webovou aplikaci.
5. Proveďte zhodnocení dosažených výsledků a diskutujte další možný vývoj projektu.

Literatura:

- Feature Driven Development -- The portal for all things FDD. [http://www.featuredrivendevelopment.com/]
- Peter Coad, Eric Lefebvre, Jeff De Luca: Java Modeling In Color With UML. (6. kapitola) [http://www.pcoad.com/download/bookpdfs/jmcuch06.pdf]
- Steve Palmer and John M. Felsing: A Practical Guide to Feature-Driven Development.
- UML Resource Page. [http://www.uml.org/]

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 - 2.

Podrobné závazné pokyny pro vypracování siplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz

Vedoucí: **Rychlý Marek, Mgr.,** UIFS FIT VUT

Datum zadání: 28. února 2007
Datum odevzdání: 22. května 2007

Abstrakt

Tato práce se zabývá agilní metodikou Vývoje podle rysů (Feature Driven Development - FDD). Pro podporu této metodiky byl vytvořen informační systém, který poskytuje všem členům vývojového týmu prostředky pro následování této metodiky. Tento víceuživatelský systém je implementován jako webová aplikace. Umožňuje sestavení seznamu rysů, plánování projektu, podporu spolupráce uvnitř rysového týmu a názorné sledování pokroku projektu. Pro lepší vizualizaci pokroku je sestavena široká paleta reportů určena pro management společnosti a koncového zákazníka.

Klíčová slova

Vývoj podle rysů, FDD, rys, rysové týmy, vlastnictví tříd, ASP.NET, C#

Abstract

This thesis deals with the Featured Driven Development (FDD) agile methodics. To support this methodics, an information system has been created, providing all team-members with instruments to follow the methodics. This multi-user system is implemented as a web-based application, enabling creation of a feature list, plan a project, support the cooperation among the feature team members and watch the project progress in an illustrative way. To improve the visualization possibilities, a wide range of reports aimed at the company management and the client is provided.

Keywords

Feature driven development, FDD, feature, feature-team, class ownership, ASP.NET, C#

Citace

Tichá Pavlína: Informační systém pro podporu procesu vývoje podle rysů. Brno, 2007, diplomová práce, FIT VUT v Brně.

Informační systém pro podporu procesu vývoje podle rysů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením Mgr. Marka Rychlého.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Pavlína Tichá
14.5.2007

Poděkování

Na tomto místě bych ráda poděkovala vedoucímu mé diplomové práce Mgr. Marku Rychlému, který mi poskytoval rady ohledně metodiky FDD a také cenné konzultace ohledně návrhu systému.

Obsah

1 Úvod

Metodika Feature-Driven Development (dále jen FDD) je jedním z hlavních představitelů agilního pojetí vývoje softwaru. Metodika přináší zcela nové přístupy do tvorby softwarových produktů a snaží se reflektovat potřeby jejich odběratelů. Ti upřednostňují především rychlost a nízkou cenu vývoje, která však nesmí být na úkor kvality. Také mají právo aktivně se účastnit vývoje a být informováni o stavu, v jakém se jejich projekt nachází.

Metodika FDD přináší do problematiky vývoje projektů mnoho nových myšlenek a vývoj se díky ní stává rychlejší a efektivnější. Proto také vznikla tato diplomová práce. V rámci semestrálního projektu byly prostudovány principy a možnosti metodiky FDD, na jejichž základě byl poté implementován informační systém pro podporu této metodiky.

První kapitola popisuje vznik FDD a jsou zmíněni její zakladatelé. Je zaveden pojem rys (feature) a jeho vysvětlení v širších souvislostech. Jsou vysvětleny také základní principy a procesy metodiky FDD.

Druhá kapitola se soustřeďuje na průzkum používání metodiky FDD v praxi. V rámci této kapitoly jsou také uvedeny a zhodnoceny existující systémy pro podporu této metodiky.

Podstatou třetí kapitoly je analyzovat metodiku FDD pro potřeby vyvíjené aplikace. Jsou zde zmíněna možná úskalí a nejasnosti metodiky, pro něž je načrtnuto řešení v budoucím systému.

Čtvrtá kapitola shrnuje požadavky kladené na vyvíjený systém. Požadavky jsou rozčleněny do dvou sekcí a na jejich základě modelovány případy použití pro jednotlivé aktéry systému.

Další kapitola se věnuje výběru technologií vhodných k implementaci navrženého systému. Je zde uvedeno, jaké prostředky musí tyto technologie poskytovat.

Pro detailní návrh celého systému je rezervována kapitola šestá, ve které je návrh architektury aplikace, návrh databáze a objektově orientovaný návrh diagramu tříd. Mimo jiné jsou zde i návrhy určitých modulů systému jako je reportovací systém nebo systém rolí a práv.

V kapitole sedmé je popsána implementace systému pro podporu vývoje podle rysů. Jsou zde zmíněny klíčové kroky při implementaci a vyjmenovány podpůrné technologie využívané pro potřeby webové aplikace.

Závěrečné kapitoly této diplomové práce hodnotí dosažené výsledky tvorby požadované webové aplikace. Zaměřují se na rozbor možných rozšíření této aplikace a hodnotí přínos vzniku tohoto systému.

2 Feature-Driven Development

FDD bylo poprvé představeno roku 1999 v knize Java Modeling in Color with UML [1]. Autory této publikace jsou Peter Coad, Eric Lefebvre a Jeff de Luca. Právě Peter Coad zavádí v roce 1997 pojem rys (feature) a spojuje iterativní přístup k tvorbě softwaru se svými zkušenostmi z průmyslu. Poprvé bylo FDD nasazeno při vývoji softwaru pro banku v Singapuru roku 1997 a podle tohoto projektu byl směřován jeho další vývoj.

2.1 Základní charakteristika

Metodika FDD je zařazena mezi agilní metody. Anglický název Feature-Driven Development je možné přeložit do češtiny jako vývoj řízený rysy. Přičemž literatura se rozchází v překladu slova feature, které odpovídá rysu, vlastnosti, ale také charakteristice. V publikaci [9] se uvádí překlad „užitná vlastnost“, který poskytuje nejlepší český ekvivalent anglickému výrazu. V této práci se ale bude vycházet ze zadání a tento klíčový pojem se bude označovat pouze jako rys. I přesto, že v následujících kapitolách bude rys podrobně vysvětlen, je pro pochopení principu FDD nutné jej alespoň stručně popsat. Rys je klientem validovatelný blok funkčnosti, který vytváří základní stavební kámen při implementaci a sledování postupu projektu. Je to v podstatě rozdělení problému na menší snáze řešitelné podproblémy.

FDD se skládá z pěti sekvenčních procesů, z nichž poslední dva jsou iterativně opakovány. V prvních třech procesech se tvoří celkový model systému. Tato skutečnost napovídá, že FDD klade důraz na fázi modelování a na vytváření modelu obecně. Mohlo by se zdát, že se FDD zpronevěřuje některým zásadám manifestu agilních technologií[1]. Ale na rozdíl od jiných agilních metodik, kde je základním prvkem vývoje iterace, FDD se skládá pouze z rysů. Což jsou v porovnání s celou iterací velmi mále úseky. Proto je nutný dobrý objektový model, který napomáhá integrovat všechny rysy do funkčního celku. Má za úkol poskytnout zevrubný pohled na systém a stanovit hlavní vývojovou linii, po které se bude tým ubírat [5].

Díky tomu, že FDD byla vyvinuta v praxi, tak se snaží reflektovat nejenom potřeby zákazníků, ale také účastníků vývojového týmu. Jako moderní metodika má za cíl dodat zákazníku fungující produkt v co nejkratším čase a zároveň vytvořit pro vývojáře práci zábavnou. Je kladen silný důraz na výsledný produkt, a proto se snaží zvýšit efektivnost a produktivitu práce a přitom eliminovat nejistotu a zmatky při vývoji [1].

Jedním z důležitých aspektů FDD je pravidelné dodávání nových betaverzí. Jelikož celý vývoj je řízen rysy, kde doba potřebná pro jejich implementaci by neměla přesáhnout dva týdny, tak je

[1] www.agilemanifesto.org

vývojový tým schopen dodat zákazníkovi nový funkční meziprodukt přibližně každé dva týdny. Tato skutečnost představuje velkou výhodu, jelikož zákazník i vedení projektu mají stálou představu o tom, jak pokračuje vývoj a zda se ubírá správným směrem. V případě jakýchkoliv připomínek mohou upravit požadavky a vést tak projekt k správnému řešení.

2.2 Rysy

Rys je základním stavebním kamenem metodiky FDD. Je definován jako malá část funkčnosti, která přináší užitek z pohledu zákazníka. Každý rys by měl být takového rozsahu, aby jej bylo možné dokončit v průběhu maximálně dvou týdnů. To umožňuje vývojářům produkovat fungující výsledky každé dva týdny. Proto je možné sledovat viditelné pokroky ve vývoji a poskytnout zákazníku jasnou představu o stavu projektu.

Důležité vlastnosti rysů [5]:

- **měřitelnost**: schopnost rozhodnout, zda implementovaný rys odpovídá požadavkům zákazníka. Zákazník sám musí být schopen určit, zda výsledná funkčnost rysu odpovídá jeho požadavkům. Neměla by tedy nastat známá situace: „Zákazník si přeje něco, ale systém dělá úplně něco jiného“. Klient sám musí mít možnost validovat implementované rysy.
- **srozumitelnost**: schopnost daný rys přesně popsat, tedy určit, co bude jejím výsledkem. Popis by měl být jasný především zákazníkovi a ne pouze vývojářům.
- **realizovatelnost**: tato vlastnost je velmi důležitá především pro určení správného rozsahu rysu. Je důležité umět rozhodnout o tom, zda je daný rys realizovatelný a zda doba realizace odpovídá nanejvýš dvěma týdnům. Pokud by byla doba potřebná k implementaci daného rysu delší, pak by se musel rys rozdělit do několika menších rysů. Tímto přístupem získané rysy jsou na relativně stejné úrovni složitosti. Nenastává tedy situace známá z případů použití, kdy jeden je možné dokončit za týden a jiný může trvat několik měsíců.

Jelikož jsou rysy základním stavebním kamenem metodiky FDD, tak je určen přesný formát pro jejich zápis. Tato šablona poskytuje zároveň určité vodítko pro implementaci rysů. Určuje, jaké třídy se bude rys týkat a jaké metody by se měly implementovat. Příkladem může být rys „Výpočet průměrného platu zaměstnance“, který se pravděpodobně bude týkat třídy `Zamestnanec` a metody `vypocet_prum_platu`..

Šablona pro zápis rysů:

```
<akce> <výsledek> <objekt>
```

Akce	Činnost, která má být v rámci daného rysu provedena. Je to například: vypsaní, vypočítání, určení, namalování…
Výsledek	Představuje v české větě předmět. V podstatě je to artefakt, nad nímž je provedena akce.
Objekt	Osoba, místo nebo věc. Pod tímto pojmem jsou zahrnuty také role, časové údaje nebo intervaly a popisy vstupních dat.

Příklad rysů:

- Vypočítat docházku zaměstnanců
- Nakreslit graf docházky
- Vypsání zaměstnanců na obrazovku

Rysy jsou seskupovány do **množin rysů**. Toto seskupení je prováděno na základě podnikatelských souvislostí. Šablona pro zápis množiny je následující:

```
<akce> <objekt>
```

Příklad množiny rysů:

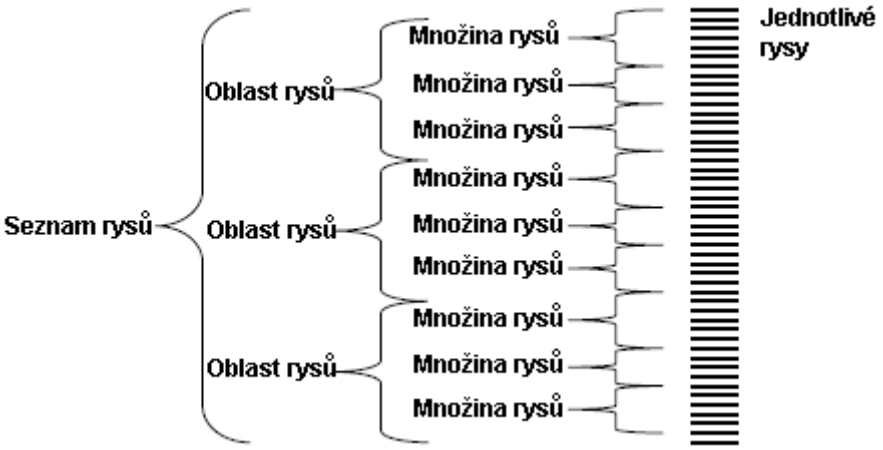
- Vytvořit schéma docházky

Množiny rysů jsou soustředěny v hlavní množině, která je označována jako **oblast rysů** a má následující šablonu:

```
<objekt> management
```

Příklad oblasti rysů:

- Správa zaměstnanců.



Obr. 2-1. Struktura seznamu rysů

2.3 Postupy v FDD

FDD jako metodika nepodceňuje význam postupů při vývoji softwaru, ale nestaví jej na první místo. Peter Coad v publikaci [1] dokonce upozorňuje na to, že mnoho vývojářů je tak zaslepeno postupy, že si často neuvědomují, že jsou placeni za výsledný produkt a ne za postupy použité při vývoji. Příliš složité postupy a nutnost studovat stostránkové specifikace demoralizuje členy týmu a může vést pouze k slepému následovaní postupů a ztrátě vlastního názoru a nápadů. Pro vývojáře je mnohem přínosnější stručný a přehledný návod před mnoha stranami nic neříkající specifikace [5]. Čas, který je v mnoha případech potřebný k nastudování postupů, je mnohem přínosnější využít k samotnému vývoji.

Metoda FDD upřednostňuje přehledný a věcný rámec, který je nutný ke kvalitnímu vývoji, ale nenařizuje vývojářům veškeré podrobnosti postupu. Pokud tedy FDD určuje, že se má provést návrh a jeho inspekce, pak striktně nenařizuje jak se má návrh vypracovat a jak provést inspekci [1]. Metoda pouze říká, *co* se má provést, ale nic neříká o tom, *jak* se to má provést. Zde se projevuje flexibilita a volnost plynoucí z podstaty agilního přístupu. Určitá část odpovědnosti je tedy ponechána na hlavním programátorovi, který musí být dostatečně zkušený, aby zvolil správné nástroje a metody pro jednotlivé kroky vývoje. Na rozdíl od metodik s přesně vyspecifikovanými kroky a postupy poskytuje FDD možnost přizpůsobit vývoj aktuální situaci a požadavkům zákazníka.

Hlavní důvody pro zavedení postupů do metody FDD:

1. **Přechod k velkým projektům a opakovatelným úspěchům**
   Pokud je při vývoji zvolen správný a dobře pracující postup, pak se stává pro členy týmu automatickou součástí práce a umožňuje jim zaměřit se více na výsledky postupů než na jejich jednotlivé kroky provádění.
2. **Kratší doba zapracování nových zaměstnanců**
   Při jednoduchém postupu je uvedení nových pracovníků mnohem rychlejší. Nový zaměstnanec v mnohem kratším časovém úseku pochopí všechny souvislosti a může začít efektivně pracovat.
3. **Zaměření na výsledky projektů**
   Výsledky projektu musí být vyvážené. Je nutné mít tedy při celém vývoji na paměti komplexnost výsledku a ne, aby určitá část byla dokonale vypracována na úkor jiných částí.

2.3.1 Popis postupu

FDD vychází z předpokladu, že nejlepší jsou krátké, jednoduché a jasné postupy. Nejlepší postupy jsou podle Peter Coada ty, které byly napsány pouze na jedné nebo dvou stranách. Vzor pro popis postupů je v angličtině označován zkratkou ETVX : Entry, Task, Verification a eXit. Tento předpis je využíván také při popisu jednotlivých fází metody FDD. Skládá se z následujících částí:

- Entry - Specifikace jasných a přesných vstupních kritérií
- Task - Seznam úkolů v rámci daného procesu včetně identifikace řešitelů a určení odpovědných osob
- Verification - Nástroje a metody verifikace. Tento bod určuje míru funkčnosti, při které je možné prohlásit proces za dokončený.
- Exit - Výstupní kritéria projektu. Zahrnuje fyzické výstupy, dokumenty, prototypy apod.. V podstatě se zde definují konkrétní výsledky, jaký má být konečný produkt, v jakém formátu a co bude jeho výstupem.

Jasně definovaný postup vede k mnohem efektivnějšímu pokroku. Pokud by každý vývojář zvolil svůj vlastní postup, tak by docházelo ke zmatkům a práce by přestala být efektivní.

2.4 Role

Metodika FDD klasifikuje role do tří kategorií: klíčové, podpůrné a dodatečné.[2] Každá role má přesně definováno, co je jejím úkolem v jednotlivých fázích vývoje. Je také určeno, jaká role se bude jaké fáze účastnit a nést za ni případně odpovědnost. Jednomu členu týmu může být přiděleno více rolí a zároveň jedna role může být sdílena více lidmi. V následujících podkapitolách jsou jednotlivé role blíže popsány.

2.4.1 Klíčové role

Projektový manažer (Project Manager) je administrativním a finančním vedoucím projektu. Jednou z jeho úloh je zajistit celému týmu takové pracovní podmínky, aby byli všichni spokojení a mohli efektivně odvádět svoji práci. Má rozhodující slovo v otázkách rozsahu projektu, časového harmonogramu a personálního obsazení týmu.

Vrchní architekt (Chief Architect) nese odpovědnost za celkový návrh systému. Zajišťuje pravidelné pořádání schůzek ohledně návrhu. Má poslední slovo při řešení problémů a při klíčových rozhodnutích ve fázi návrhu. Pokud je to nutné, muže být tato role rozdělena do dvou rolí, konkrétně doménového architekta a technického architekta.

Vývojový architekt (Development Manager) má na starosti problematiku vývoje. Řeší problémy se zdroji a možné konflikty, které mohou nastat uvnitř vývojového týmu. Často se jeho úkoly kryjí s úkoly projektového manažera a hlavního architekta.

Hlavní programátor (Chief Programmer) je zkušený vývojář, který se podílí na analýze požadavků a návrhu systému. Je odpovědný za vedení malých programátorských týmu při analýze, návrhu a vývoji rysů. Z množin rysů vybírá ty, které se budou implementovat v další iteraci a určuje vlastníky jednotlivých tříd. Spolupracuje s dalšími hlavními programátory při řešení technických problému mezi jednotlivými týmy. Každý týden podává zprávu o pokrocích projektu za vlastní tým.

Vlastník třídy (Class Owner) pod vedením hlavního programátora realizuje přidělené třídy. Provádí návrh, kódování, testovaní a sepisuje dokumentaci k daným třídám. Je odpovědný za vývoj třídy, které je vlastníkem. Z členů týmu nesoucích tuto roli se formují rysové týmy.

Doménový expert (Domain Expert) může představovat koncového uživatele, klienta, sponzora nebo obchodního analytika. Jedná se o člověka, který dokonale rozumí cílové oblasti, ve které bude daná aplikace nasazena. Proto zadává požadavky na systém a spolupracuje s vývojáři, aby přesně pochopili, jakou má zákazník představu o výsledném produktu. Po celý čas vývoje dbá na to, aby se dodržovaly stanovené požadavky a kontroluje průběžné výsledky.

2.4.2 Podpůrné role

Manažer pro dodávky (Release Manager) kontroluje celkový pokrok ve vývoji. Hodnotí kontrolní zprávy od hlavních programátorů a pořádá s nimi pravidelná setkání. O celkové úspěšnosti informuje projektového manažera.

Jazykový specialista (Language Guru) je člen týmu, který důkladně ovládá programovací jazyk nebo technologií využívanou při vývoji. Tyto své znalosti je schopen předávat vývojářům a zajišťovat tak vzdělávání členů týmu. Tato role je zvláště důležitá, pokud tým zavádí novou technologii.

Konstruktér (Build Engineer) je osoba odpovědná za nasazení, údržbu a chod vytvořeného projektu. Má na starosti také správu verzí a publikování dokumentace.

Nástrojář (Toolsmith) má za úkol programovat nebo obstarávat podpůrné nástroje pro potřeby vývojářů. Muže také být pověřen udržováním databáze nebo webových stránek týkajících se projektu.

Systémový administrátor (System Administrator) spravuje a konfiguruje servery, sítě a pracovní stanice, které vývojáři potřebují ke své práci. Jeho práce spočívá v údržbě softwaru potřebného k vývoji.

2.4.3 Dodatečné role

Tester (Tester) ověřuje, zda vytvořený systém odpovídá požadavkům zákazníka. Může se jednat o člena týmu, ale také o externího člena, který zastává roli nezávislého hodnotitele.

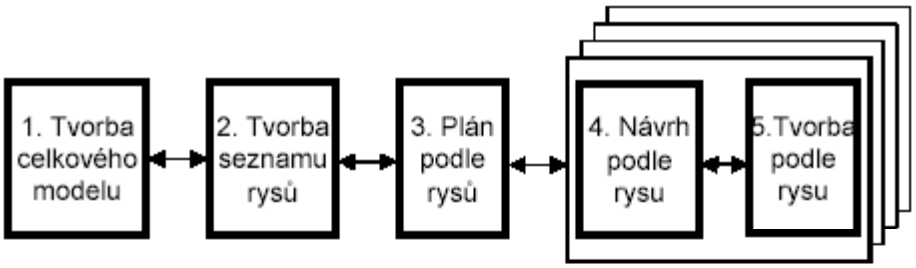
Správce nových verzí (Deployer) se stará o konverzi dat do formátu odpovídajících novému systému a podílí se na vydávání nových verzí (včetně betaverzí).

Pisatel (Technical Writer) je autorem uživatelské dokumentace a manuálu.

2.5 Procesy v rámci FDD

V rámci FDD je definováno pět procesů.

- Tvorba celkového modelu (Develop an overall model)
- Tvorba seznamu rysů (Build a feature list)
- Plán podle rysů (Plan by feature)
- Návrh podle rysů (Design by feature)
- Tvorba podle rysů (Build by feature)



Obr. 2-2: Posloupnost procesů v metodice FDD [7]

V prvních třech procesech je vytvořen celkový model systému a poslední dva procesy jsou v iteracích opakovány pro každý rys. V následujících podkapitolách budou jednotlivé procesy popsány.

2.5.1 Tvorba celkového modelu

První fáze se účastní doménoví experti a ostatní členové vývoje pod vedením systémového architekta. Doménový expert nejprve seznámí členy vývojového týmu s oblastí, pro kterou bude projekt tvořen, s cílem projektu a s hlavními požadavky na výslednou aplikaci. Všechny tyto informace jsou na

vysoké úrovni abstrakce a jsou předávány formou náčrtků, obrázků a popisů v přirozeném jazyce. Členům týmu jsou také poskytnuty ke studiu dokumenty popisující danou doménovou oblast. Cílem této části je nastínit členům týmu charakter projektu a směr, kterým se bude ubírat.

V následujícím kroku doménový expert společně s vývojáři vytvoří základní model, který bude v dalších krocích postupně zpřesňován. Tento model představuje pouze základní vyjádření účelu vyvíjeného systému. Po vytvoření tohoto modelu doménový expert vysvětluje a zpřesňuje požadavky na očekávané řešení. Členové týmu nyní pracují v malých týmech pod vedením vrchního architekta. Každý z malých týmů pracuje na určité oblasti systému a při pravidelných schůzkách předkládá své poznatky a začleňuje získané výsledky do celkového modelu. Tato fáze probíhá v několika iteracích, kdy členové týmu zkoumají danou doménovou oblast z různých úhlů a získávají stále více informací od doménových expertů.

Cílem této první fáze je vytvoření základního modelu, který je později iterativně aktualizován na základě poznatků vzniklých při implementaci jednotlivých rysů ve čtvrté a páté fázi.

Vstupní kritéria: Klient je připraven přikročit k vývoji požadovaného systému. Byli vybráni doménoví experti, vrchní architekt a hlavní programátor, kteří povedou tuto fázi. Klient má sestaven seznam požadavků na výsledný systém.

Úkoly:

- Sestavení týmu pro tuto fázi
- Seznámení s doménovou oblastí – doménový expert seznámí zbytek týmu s oborem, kterého se týká
- Studium relevantních dokumentů
- Vytvoření neformálního seznamu rysů
- Vytvoření malých týmu pro tvorbu modelu
- Tvorba celkového modelu – z modelů, které vzniknou v malých týmech je vybrán jeden, který bude označen za hlavní a bude tvořit základní linii celého projektu. Tento model je později pod vedením vrchního architekta upravován na základě modelů jiných týmů.
- Sepsání poznámek k modelu

Verifikace: průběžná posudková řízení jak v rámci týmu, tak i externí se zákazníkem

Výstupní kritéria: k dokončení této fáze jsou potřebné následující výsledky.

- Diagram tříd s uvedením tříd, odkazů, metod a atributů. Je v podstatě vytvořen objektový model. Pokud existují, tak také sekvenční diagramy.
- Neformální seznam rysů
- Poznámky k významným alternativám modelování

2.5.2 Tvorba seznamu rysů

Druhá fáze je zaměřena na vytvoření seznamu rysů, jejich rozdělení do množin a množin do oblastí. Tento seznam ještě není konečný, ale měl by již obsahovat nezbytné rysy pro zajištění funkčnosti výsledného systému. Rysy jsou tvořeny na základě celkového modelu vytvořeného v první fázi a měly by odpovídat všem požadavkům vzneseným na systém.Při sestavování seznamu je využíván nejenom základní model z první části, ale také všechny poznámky a dokumenty.

Po vytvoření všech rysů je nutné rozdělit je do množin podle kontextu a obchodních souvislostí. Pokud je systém dostatečně rozsáhlý, tak můžou být množiny ještě sdružovány do oblastí. Množin a oblastí by mělo být právě tolik, aby rozdělení rysů bylo logické a přehledné.

Rysy jsou hierarchicky rozděleny do čtyř úrovní podle priority. Úroveň A představuje rysy, které musí být realizovány. V úrovni B už jsou zastoupeny ty rysy, které by si zákazník přál, aby byly realizovány. V úrovni C a D jsou rysy, které nejsou podstatné pro funkčnost výsledného projektu. Rysy ze skupiny C budou přidání pouze pokud to bude možné a rysy zařazené ve skupině D jsou pouze potenciální a jsou zmíněny z důvodu možné implementace v budoucnosti. Při určování priority rysů se vždy zohledňují následky, které nastanou, pokud daný rys nebude implementován (sankce od klienta, nefunkčnost systému atd.) [1]

V této fázi již musí mít zákazník přesnou představu, jaké rysy jsou nezbytné pro funkčnost systému a kdy je daný seznam úplný. Většinou se právě podle seznamu vlastností určuje rozsah projektu a sjednává konečná cena.

Vstupní kritéria: Je vytvořen úspěšně model z první fáze a sestavení doménového a vývojového týmu.

Úkoly:

- Sestavení týmu pro tuto fázi – členy jsou opět stálý vývojáři a doménoví experti
- Vytvoření seznamu rysů – rysy jsou sestaveny z neformálního seznamu první fáze a také mohou vznikat přepisem metod z objektového modelu. Další rysy jsou získávány při brainstormingu na základě podnětů jednotlivých členů týmu.
- Určení priority rysů a množin rysů
- Rozdělení rysů – vývojáři pod vedením vrchního architekta zváží kolik času bude potřeba k vytvoření jednotlivých rysů. Pokud je tento časový interval delší než dva týdny, pak jsou tyto rysy rozděleny na více menších rysů.

Verifikace: průběžná posudková řízení jak v rámci týmu, tak i externí se zákazníkem

Výstupní kritéria: Tým musí vytvořit detailní seznam rysů s určenou prioritou a rozdělením do množin a oblastí.

2.5.3 Plán podle rysů

V této třetí fázi je využíván seznam rysů z předešlého kroku. Je využit k sestavení plánů vývoje a stanovení milníků při vývoji. Plánování se týká především určením pevným datům jednotlivých milníků. Nejdůležitějším datem je předpokládané ukončení vývoje. Toto datum je většinou pevně zapsáno ve smlouvě o dílo a následný prodlužováním vývoje již společnost ztrácí peníze. Mezi další milníky patří dokončení jednotlivých rysů a množin rysů. Tyto časové údaje už ale mohou být posouvány a modifikovány podle aktuálního stavu vývoje.

Každé množině rysů je přiřazen vlastník, který ponese odpovědnost za jejich realizaci. Také je určeno předpokládané datum dokončení rysu.

V této fázi je také přiřazen vlastník jednotlivým třídám nebo skupinám tříd. Jak bude vysvětleno dále tento vlastník představuje programátora, který bude vyvíjet danou třídu a ponese odpovědnost za její úspěšné dokončení [5].

Vstupní kritéria: Úspěšně sestavený seznam vlastností z druhé fáze.

Úkoly:

- Sestavení týmu pro tuto fázi – tým se skládá z projektového manažera, vývojového manažera a hlavních programátorů
- Sestavení sekvencí množin rysů – tým sestaví časovou posloupnost tvorby rysů a určí předpokládané datum dokončení
- Přiřazení důležitých obchodních procesů šéfům programátorů
- Přiřazení tříd vývojářům

Verifikace: průběžná interní posudková řízení v rámci týmu a externí posudková řízení prováděná senior manažerem. Je potřeba se vyvarovat časté chybě vrchním programátorů, kteří předpokládají, že všichni ostatní vývojáři jsou stejně schopní a výkonní jako oni sami. Tato chyba vede k určení nereálně krátké doby pro realizaci rysů. Proto musí být kontrolováno, zda jsou časové odhady reálné.

Výstupní kritéria: pro opuštění této fáze musí být vytvořen vývojový plán s následujícími milníky:

- Datum dokončení celého projektu
- Pro každý rys, množinu a oblast musí být určen vlastník a datum dokončení
- Přiřazení vlastníků třídám

2.5.4 Návrh podle rysů

Tato fáze je spolu s fází implementace iterativně opakována pro každý rys nebo množinu rysů. Dle stanovených kritérií a logických souvislostí vybírá při každé iteraci vrchní programátor jeden rys nebo množinu rysů. Následně identifikuje, které třídy jsou důležité pro jeho implementaci a kontaktuje vlastníky těchto tříd. Vlastníci tříd jsou seskupeni do rysových týmů, které se tvoří

dynamicky pouze pro realizaci daného rysu. Tento tým sestaví podrobný sekvenční diagram, vypracuje detailní návrh pro implementaci daného rysu a stanoví plán pro realizaci. Tento proces se označuje zkratkou DBF neboli „Design By Feature“, tedy návrh podle vlastností.

Vstupní kritéria: Je vytvořen plán podle rysů z třetí fáze.

Úkoly:

- Sestavení rysového týmu
- Podrobné seznámení s doménou daného rysu
- Studium relevantních dokumentů
- Vývoj sekvenčního diagramu – rysový tým vytvoří formální sekvenční diagram pro daný rys. Dodá k němu alternativní návrh a poznámky. Vrchní programátor pak zařadí daný diagram do celkového modelu
- Aktualizace objektového modelu – každý vlastník třídy aktualizuje třídy a její metody podle sekvenčního diagramu. Přidají se typy parametrů, návratové hodnoty, výjimky apod.
- Pokud je to nutné, tak jsou přizváni externí odborníci.

Verifikace: interní inspekce je v podstatě prováděna při sestavování sekvenčního diagramu a externí inspekce je prováděna pouze v tom případě, že nastanou nějaké problémy.

Výstupní kritéria: k dokončení této fáze jsou potřebné následující výsledky.

- Určení rysu a odpovídajících dokumentů
- Detailní sekvenční diagram
- Diagram tříd
- Aktualizovaný objektový model
- Poznámky k alternativám návrhu

2.5.5 Tvorba podle rysů

Tato poslední fáze představuje implementaci jednotlivých rysů a je známá pod zkratkou BBF, nebo-li „Build By Feature“, tedy realizace podle rysů. Každý vlastník třídy, který je členem rysového týmu vystaví metody potřebné pro rysy. Také vytváří pro danou třídu testovací případy a provádí všechny testy. Rysový tým provádí inspekci kódu, aby zaručil, že je korektně implementovaný a dostatečně prověřený. Pokud jsou všechny třídy dostatečně prověřeny, tak je možné, aby hlavní programátor integroval daný rys do hlavní aplikace.

V jednom časovém intervalu je většinou realizováno více rysů. Existuje tedy souběžně několik nezávislých rysových týmů. Jeden vlastník třídy přitom může být členem několika rysových týmů. Šéf architekt tedy už v předešlé fázi musí správně určit, které vlastnosti je možné implementovat ve stejném časovém období. Dokončené rysy jsou v konečné fázi začleněny do systému.

Vstupní kritéria: Úspěšné dokončení předchozí fáze.

Úkoly:

- Implementace tříd a metod
- Inspekce kódu
- Testování
- Integrace do systému

Verifikace: průběžná posudková řízení jak v rámci týmu, tak i externí se zákazníkem

Výstupní kritéria: k dokončení této fáze jsou potřebné následující výsledky.

- Implementované třídy a metody daného rysu, na kterých byla provedena inspekce kódu a testy
- Případné další třídy potřebné důležité pro funkcionalitu
- Dokončený rys integrovaný do systému

2.6 Klíčové principy FDD

V předchozích kapitolách bylo vysvětleno, jaká je podstata vývoje podle metodiky FDD. Byly popsány jednotlivé fáze vývoje a zmíněny některé velmi zajímavé praktiky. Přestože většinou nejsou ničím novým v oblasti softwarového inženýrství, tak při společném používání prokazují v praxi dobré výsledky. Proto jsou v této kapitole jednotlivé principy detailně popsány a je kladen důraz na jejich přínos v metodice FDD.

2.6.1 Objektové modelování domén

Tato praktika se uplatňuje v první fázi vývoje, kde je tvořen celkový model systému. Doménu, tedy oblast, ve které bude výsledná aplikace nasazena, je doporučeno modelovat objektově. Při objektovém modelování je možné znázornit vztahy mezi třídami, dědičnost a specializace. U každé třídy jsou uvedeny metody, které představují interakci mezi třídami. Mimo diagram tříd se doporučuje vytvořit také sekvenční diagramy. Z těchto diagramů je zřejmá komunikace na úrovni tříd.

Při návrhu tohoto modelu je kladen důraz na odpoutání od pozdější architektury aplikace. V této fázi má být vytvořen pouze abstraktní model, jehož komponenty budou detailně navrženy a implementovány v pozdějších fázích. Tímto přístupem se FFD snaží zabránit roztříštěnosti návrhu.

Autor metodiky [1] doporučuje využít při tvorbě modelu tzv. barevné modelování (UML colors), které zvýší přehlednost a srozumitelnost modelu. Barevné modelování je založeno na dlouhodobé analýze a sledování vznikajících modelů, kde bylo zjištěno, že určité typy tříd se vyskytují ve všech modelech a mají pouze jiné názvy podle problematiky projektu. Z tohoto

pozorování vyplynulo, že je možné třídy rozdělit do čtyř hlavních skupin a ty barevně odlišit. Skupinám tříd byly přiřazeny barvy následovně:

- **růžová** – časové momenty a intervaly
- **žlutá** – role
- **modrá** – popisy
- **zelená** – osoby, místa a věci

Obohacení celkového modelu není v FDD předpisem, ale jedná se pouze o doporučení, které poskytuje lepší orientaci v modelu.

2.6.2 Vývoj podle rysů

Jak bylo již výše zdůrazněno, metoda FDD je založena na rysech. Při implementaci systému se postupuje po jednotlivých rysech a ne jak je obvyklé z tradičních metodik po modulech nebo funkcích. Rysy jsou pojmenovány tak, že jejich významu rozumí i odběratel aplikace a nejenom zasvěcení členové vývojového týmu. Navíc vývojáři produkují po skončení každé iterace, tzn. cca každé dva týdny, fungující meziprodukt, který může klient validovat a případně nasměrovat vývoj takovým směrem, aby odpovídal požadavkům. Zde se také projevuje hlavní charakteristika FDD a to zaměření celého vývoje na fungující aplikaci. V systému nejsou implementovány zbytečné metody a třídy, které nejsou nutné k požadované funkcionalitě.

Vývoj podle rysu je velmi kladně přijímán i samotnými vývojáři. Umožňuje jim každé dva týdny odevzdávat fungující meziprodukt. Tento fakt se pozitivně odráží v morálce a spokojenosti vývojářů, kteří za sebou vidí vykonanou práci a fungující výsledek.

2.6.3 Vlastnictví tříd

Myšlenka vlastnictví kódu konkrétní osobou není vlastní pouze metodice FDD, ale vyskytuje se i v jiných přístupech softwarového inženýrství. Zde je vlastnictví kódu vymezeno třídou a vlastník odpovídá za její návrh a úspěšnou implementaci.

Vlastnictví kódu sebou přináší řadu výhod. Vlastník třídy sám implementuje všechny metody a zná tedy veškeré souvislosti. Pokud je ke třídě potřeba implementovat určitá rozšíření, tak jejich realizaci provádí zase vlastník a není tedy nebezpečí, že by z neznalosti narušil integritu celé třídy [12]. Zároveň se dokazuje, že vlastnictví kódu vede k vyšší kvalitě práce, protože vývojáři si uvědomují vlastní odpovědnost za tyto části kódu, a proto si nedovolí odevzdat špatně fungující kód.

Je nutné ale zmínit také nevýhody, které sebou vlastnictví kódu přináší a proč se od něj jiné agilní metodiky odpoutávají. Především je to problematické převzetí kódu jinou osobou, pokud daný vlastník tým opustí. V tom případě trvá určitý čas než další vývojář převezme tento kód a pochopí přesně všechny vztahy a principy. Další nevýhodu představují vznikající zpoždění, pokud jeden vývojář požaduje změny na konkrétní třídě příslušející jinému členu vývojového týmu [12]. Tato

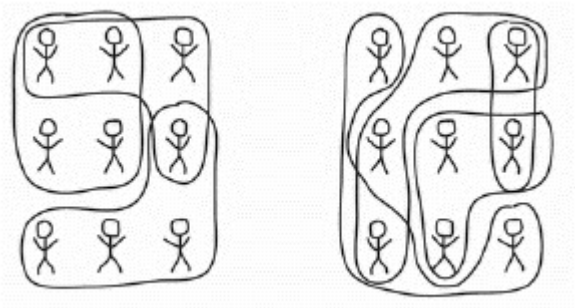
nevýhoda je ale v FDD eliminována existencí tzv. rysových týmů, které budou popsány v následující kapitole.

2.6.4 Rysové týmy

V metodice FDD je pro implementaci konkrétního rysu sestaven vždy jedinečný tým, který se skládá z vlastníků tříd, které je potřeba implementovat pro dosažení funkčnosti daného rysu. Je tedy zřejmé, že tyto týmy dynamicky vznikají a zanikají podle právě realizovaného rysu. Tím se do jisté míry předchází už zmiňovaným zpožděním na základě čekání na jiného vlastníka kódu. Vlastníci tříd totiž mají určitou volnost při výběru tříd, které budou v jaké iteraci realizovat a volí rysy tak, aby minimalizovali potenciální čekaní jednotlivých vývojářů.

Tým se obvyklé skládá ze tří až šesti vývojářů. V jednom časovém období může být realizováno více rysů a tedy vedle sebe existuje několik takových týmů. Jeden vývojář může být členem více týmů najednou, jak je znázorněno na obr. 2-3. V každém týmu má nejvyšší postavení šéf, který je zároveň vlastníkem tohoto rysu. Ten zaručuje, že daný rys je implementován správně a kompletně.

Tyto rysové týmy vedou také ke zlepšení morálky ve vývojovém týmu, protože vývojáři se po skončení každé iterace stávají členy jiného týmu, a proto neupadají do stereotypu. Zároveň si sebou z každého týmu odnášejí rozdílné návyky a zkušenosti, které mohou zhodnotit při své další práci.



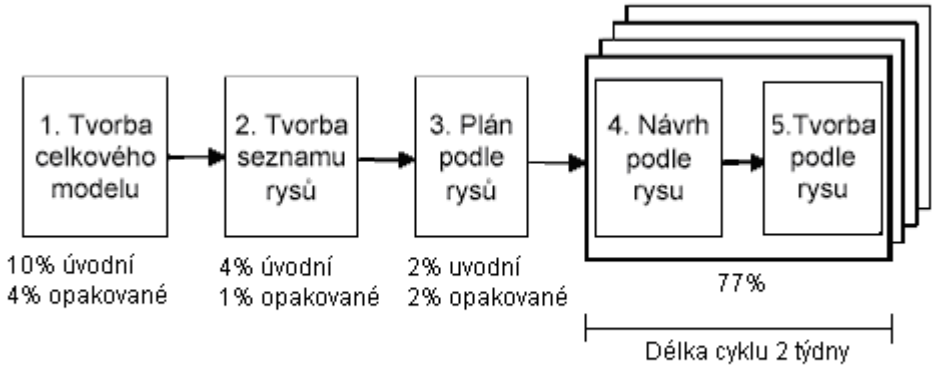
Obr.2-3: Dynamicky tvořené rysové týmy

2.7 Sledování pokroku ve vývoji

Metoda FDD poskytuje podrobný časového rozpis celého vývoje systému. Ve třetí fázi vývoje je plánováno dokončení jednotlivých milníků, které by mělo být při správném dodržování zásad FDD splněno. Není tedy překvapením, že FDD podává také přibližnou představu o tom, kolik času by se mělo strávit v jednotlivých fázích vývoje. V následující kapitole bude uvedeno, kolik procent z celkové doby vývoje je stráveno v jednotlivých fázích. Také bude popsán způsob, jakým se sleduje pokrok ve vývoji, tedy dodržování termínů a procentuální úspěšnost zpracování jednotlivých rysů.

2.7.1 Doba strávená v jednotlivých fázích

Celý cyklus vývoje je znázorněn na obr. 2-4, z kterého je zřejmé kolik času je věnováno jednotlivým částem vývoje. Toto rozvržení je pouze orientační a může se měnit podle povahy vyvíjeného projektu.



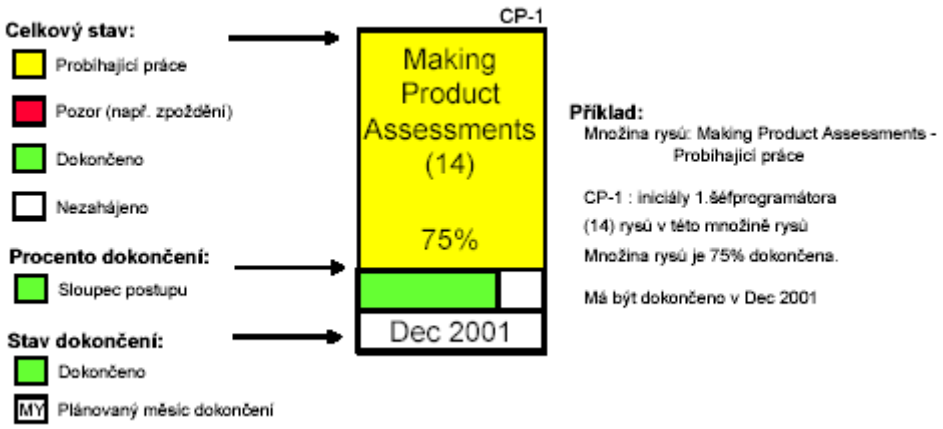
Obr. 2-4: Doba strávená v jednotlivých fázích metodiky FDD

Celý projekt začíná tvorbou celkového modelu, která by měla trvat přibližně 10 % celkového času potřebného pro projekt. Jelikož v této fázi je vytvořen i konceptuální seznam rysů, tak druhá fáze již není tak časově náročná a zabírá přibližně 4 % času. Následující fáze plánování zabere přibližně 2% času. Zbývající iterativně se opakující fáze (návrh a implementace rysů) zaujímají 77 % celkového času vývoje. Zde je patrné, jaká důležitost je přiřazována samotnému kódování jednotlivých rysů. Ve fázích návrhu je strávena pouze čtvrtina celkového času a zbytek je věnován realizaci rysů. První tři fáze mohou být provedeny i po skončení každé iterace, aby se mohly v návrhu uplatnit změny zjištěné při realizaci rysů. Tak je zajištěno, že systém pružně reaguje na potřebné změny. Při tomto opakování prvních tří fází jsou fázi tvorby celkového modelu věnovány ještě 4 %, druhé fázi 1 % a fázi plánování 2 % z celkového času [5].

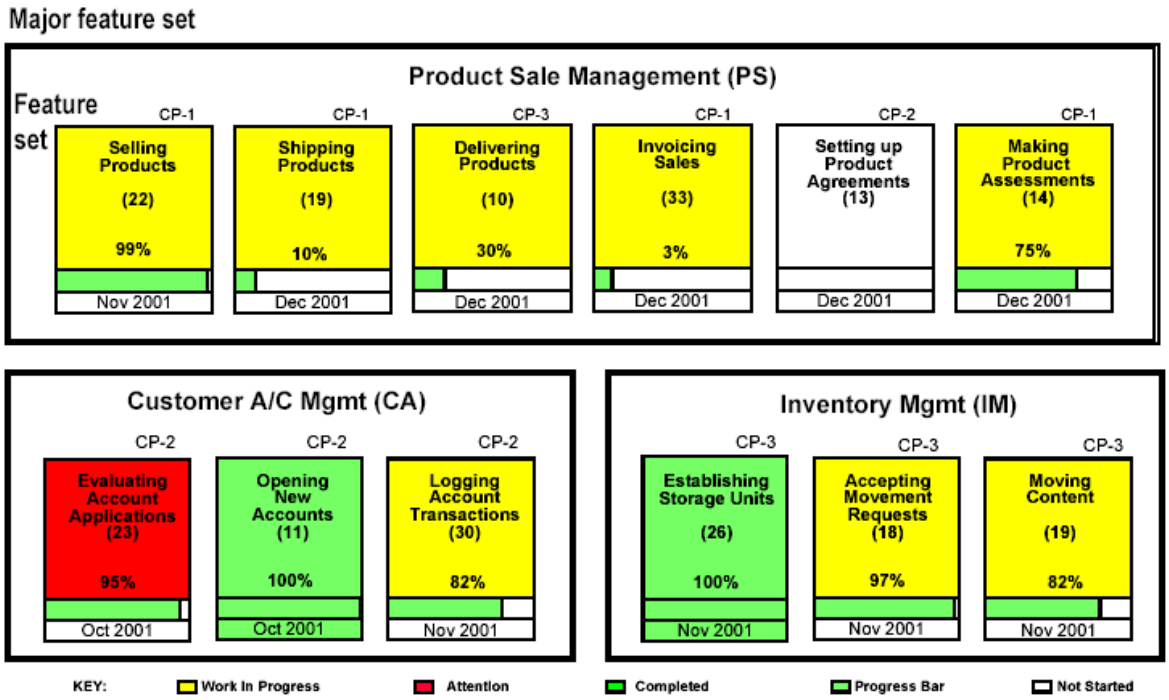
2.7.2 Kontrola vývoje

Každý týden se schází hlavní programátoři s manažerem pro dodávky (Release manager) a ústní formou hodnotí pokrok při realizaci rysů. Jednotliví hlavní programátoři hodnotí, jak pokračují ve vývoji jim přidělené rysové týmy. Vzhledem k tomu, že jsou přítomni všichni hlavní programátoři, je zajištěno, že si všichni mohou udělat představu o řešení rysů v jiných týmech a vznést nějaké podnětné připomínky nebo se inspirovat postupem jiného týmu. Na konci každého takového sezení je sepsána zpráva a aktualizovány „boxy pokroku“. Tyto zprávy jsou v určitých časových intervalech zasílány vedení společnosti a zákazníkům.

Byl zde zmíněn pojem „box pokroku“, je to vizuálním prostředkem, pomocí něhož jsou informováni klienti a vrchní vedení o stavu projektu. Ukázka boxů je na obr. 5. Každý rys představuje právě jeden box. V tomto boxu je uvedeno jméno rysu, vlastník rysu, percentuální vyjádření pokroku při zpracování rysu a předpokládaný termín dokončení. Pro zvýšení přehlednosti se v boxech využívá určité barevné schéma, kde zelené barva představuje dokončení rysu, bílá barva dosud nezapočatý vývoj, červená nebezpečí (případné komplikaci ve vývoji) a žlutá barva úspěšně probíhající realizaci rysu. Tyto boxy se stejně jako rysy, které představují, sdružují podle množin a oblastí do větších boxů viz. obr. 2-6.



Obr. 2-5: Box pokroku [7]



Obr. 2-6: Boxy pokroku sdružené podle příslušnosti rysů k množinám [1]

3 FDD v praxi

Stále více společností dnes využívá agilního přístupu k vývoji softwaru. K tomuto směru je vedou požadavky zákazníků a také tempo dnešní doby. V tomto tržním světě může uspět pouze ta společnost, která je schopna dodat funkční software v přesně ohraničeném čase a s požadovanou funkcionalitou. Proto se také upouští od dříve využívaných klasických metodik (např. vodopádový model), které kladou velký důraz na fázi modelování a plánovaní. Neumožňují ale dodatečné změny v návrhu, a proto neposkytují dostatečnou flexibilitu, kterou dnešní zákazníci očekávají a za kterou platí.

Metodika FDD, stejně jako všechny metodiky agilního programování, klade velký důraz na roli člověka při vývoji softwaru. Metodika by tu neměla být proto, aby lidem práci ztěžovala, ale naopak aby ji usnadňovala a činila ji zábavnou. Z tohoto důvodu se také přechází od klasických metodik, které tento požadavek bohužel velmi často nesplňují. Nutí lidi psát stostránkové dokumenty a doba nutná k překonání prvotních kroků metodik mnoho vývojářů odradí a nevzbudí v nich potřebné zapálení pro realizovaný projekt. Na druhou stranu moderní agilní přístupy jako je extrémní programování velmi často od všech zákonitostí klasických metodik upouštějí a tím staví vývojáře do úplně neznámé oblasti. Agilní techniky jsou některými lidmi a společnostmi právě proto přijímány s rozpaky. Obzvláště při prvním seznámení působí velmi převratně a zdá se, že do vývoje přinášejí mnoho volnosti a nejasností. Z tohoto důvodu je velmi výhodná právě metodika FDD, která dodržuje teze manifestu agilního vývoje softwaru[2] a zároveň uplatňuje i známé principy tradičního přístupu (sestavení celkového modelu, plánování).

3.1 Výsledky výzkumu

Podle průzkumu [8] provedeného Scottem Amberem, hlavním vedoucím pro agilní vývoj v IBM Rational's Methods Group, celých 41% tázaných společností využívá alespoň jednu agilní metodiku. Průzkum byl veden v březnu 2006 a zúčastnilo se jej 4 232 společností. Výsledky jsou uvedeny na obr. 3-1. První příčku mezi agilními technologiemi obsadilo extrémní programování a na pomyslném druhém místě se umístnilo FDD. FDD využívá v přepočtu 12,3% všech oslovených společností. Toto číslo jistě již není zanedbatelné a poukazuje na oblíbenost a použitelnost metody FDD v praxi.

[2] http://agilemanifesto.org/sign/display.cgi?ms=all

METODY	ODPOVĚDI
Agile MSF	191
Agile Unified Process (AUP)	216
Crystal Clear	91
Dynamic System Development Method (DSDM)	26
Extreme Programming (XP)	954
Feature Driven Development (FDD)	502
Scrum	460
Ostatní	171

Obr. 3-1: Používání agilních metodik ve společnostech [8]

Článek [10] od stejnojmenného autora Scotta Ambera poukazuje na skutečnost, že většina agilních metodik je použitelná pouze pro týmy o maximálně deseti členech. Tato skutečnost ale neplatí u metodiky FDD. Již v roce 1997, při formování základu metodiky, byly myšlenky FDD použity při vývoji softwaru pro banku United Overseas Bank v Singapuru, kde byl vývojový tým tvořen 50 členy. Vzápětí následovaly úspěšné projekty ve významných společnostech jako Sprint a Motorola, které trvaly až 18 měsíců a podíleli se na nich až 250-členné týmy. I tento fakt, že metodika FDD není určena pouze pro malé projekty a malé týmy, vede k oblíbenosti a aplikovatelnosti tohoto přístupu.

3.2 Software pro podporu vývoje metodou FDD

Jak již bylo zmíněno, FDD se využívá ve velkém množství společností, a proto vzniká poptávka po softwaru, který by ulehčil vývoj pomocí této metodiky. Existující aplikace lze rozdělit do dvou skupin a to na klasické desktopové a moderní webové aplikace. Nejprve zde bude provedeno srovnání obou skupin a následně zmínění konkrétních aplikací.

Klasické desktopové aplikace jsou stále hojně využívány a představují pro uživatele dobré řešení, pokud se často nemění účastníci týmu. Tyto aplikace je nutné na cílovém počítači nainstalovat a to může skýtat případný problém. Jak již bylo zmíněno, velmi důležitou roli v životě každého projektu hraje doménový expert, tedy odběratel aplikace. Ten by měl mít možnost neustále kontrolovat a sledovat stav projektu, a proto by měl mít přístup i k systému pro podporu vývoje. Tato skutečnost přináší u desktopových aplikací jisté omezení právě z důvodu potřebné instalace u každého zákazníka. Na druhou stranu webová aplikace poskytuje možnost přístupu bez instalace z libovolného místa. Z tohoto důvodu je mnohem vhodnější pro systémy podporující FDD.

3.2.1 FDD Tools Project

Velmi jednoduchý program FDD Tools Project[3] poskytuje pouze základní prostředky pro podporu FDD. Tato klasická desktopová aplikace vyvinutá v jazyce java umožňuje zaznamenávání rysů a jejich sdružování do nadřazených kategorií. U každého rysu je k dispozici vizuální prostředek pro zobrazení pokroku, který je nutné ručně aktualizovat.

3.2.2 FDD Tracker

Aplikace FDD Tracker[4] představuje mnohem komplexnějšího pomocníka při vývoji podle FDD. Umožňuje přístup k projektu z pohledu všech rolí, které se vyskytují v týmu. Přehledným způsobem v tabulkách reprezentuje jednotlivé rysy a již v základním výčtu je zřetelně rozlišuje barvami podle jejich stavu, který odpovídá pokroku při řešení projektu. Zároveň je možné detailní zobrazení, sledování a aktualizace jednotlivých rysů. Také je možné zobrazit status pokroku celého projektu v mnoha přehledných grafech a jejich export do formátu HTML, XML, Excel, Word, RTF, Lotus a dalších.

3.2.3 Cognizant FDD

Aplikace Cognizant FDD[5] integrovaná v Microsoft Visual Studio Team Foundation Server pokrývá všech pět kroků metodiky FDD a definuje role uživatelů, kteří mohou k systému přistupovat. Mimo základní práci s rysy přináší také správu chyb v projektu a správu kódu.

3.2.4 FDD Project Management Aplication

FDD Project Management Aplication[6] představuje jediného zástupce webových aplikací pro podporu FDD. Tento systém vyvinutý studentem Harvardské univerzity se snaží pokrýt celou problematiku FDD. Od počátečního výčtu rysů a přidělení statusů po sledování pokroku celého projektu poskytuje uživateli dostatečné prostředky pro podporu vývoje podle FDD. Bohužel neposkytuje dostatečný management uživatelů a možnost tvorby rysových týmu. Také postrádá vazbu mezi rysem a třídou, čímž se ztrácí jeden z důležitých pilířů FDD.

---

[3] http://fddtools.sourceforge.net/

[4] http://www.itps.com.au/dotnetnuke/

[5] http://www.cognizant.com/html/content/microsoft/techfddvsts.asp

[6] http://www.fddpma.net/fddpma/projectGroupWorkplaceView.jsf

4 Analýza systému

V rámci této kapitoly bude provedena analýza systému, který bude podporovat vývoj podle metodiky FDD. Pro správné provedení analýzy systému bylo nejprve nutné důkladně prozkoumat metodiku FDD, zjistit její přednosti a také slabiny. Systém by měl totiž následovat metodiku nenásilnou formou a poskytovat intuitivní nástroje pro její podporu. Z toho důvodu se v této kapitole probírají postupně jednotlivé kroky metodiky a je zde uvedeno, jaká úskalí případně skýtají. Vždy je také zmíněno, jak se budou konkrétní nejasnosti řešit v rámci realizovaného systému. Budou zde také určeny konkrétní cíle metodiky a zároveň i vyvíjeného systému.

4.1 FDD a členové vývoje

Ještě než bude přistoupeno k rozboru jednotlivých kroků metodiky, je potřeba si ujasnit personální obsazení v týmu vyvíjeného projektu. Jeden zaměstnanec společnosti může zároveň pracovat na více projektech a měl by mít centralizovaný přístup ke všem svým projektům. V každém projektu může tento zaměstnanec zastávat různé role a podle nich nabývat i pravomocí v rámci projektu. Metodika FDD určuje velký počet rolí a některé z nich představují i externí pracovníky nebo zákazníky společnosti. Na tento fakt se musí v systému pamatovat a umožnit, aby měli k systému přístup i externí členové, kteří nejsou zaměstnanci společnosti. Tím by se řešilo obsazení projektového týmu. V rámci projektu tedy mohou spolupracovat jak zaměstnanci společnosti tak externí pracovníci. Systém FDD také umožňuje přidělit každému uživateli v rámci projektu jiná práva podle mu přidělených rolí.

4.2 Analýza tvorby celkového modelu

Prvním krokem FDD je sestavení celkového modelu. Právě z toho důvodu [11] je tato metodika mnohými projektovými manažery zavrhována. Údajně sestavení podrobného modelu v první fázi neposkytuje dostatečnou flexibilitu v dalších krocích. To by byla pravda, kdyby Metodika FDD nedovolovala dodatečné změny v modelu po provedení každé iterace. Změnová řízení jsou možná, nicméně je pravdou, že na tyto operace metoda rezervuje velmi málo času. Je kladen velký důraz na znalosti a zkušenosti návrhářů a vedoucích týmu.

Vývoj celkového modelu je velmi dobře prozkoumanou oblastí a předmětem této diplomové práce není vyvinout nástroj pro tvorbu modelu v UML[7]. Proto nebude systém přímo podporovat

[7] UML - Unified Modeling Language je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů.

tvorbu modelu, ale bude poskytovat dostatečné prostředky pro zobrazení a archivaci modelů vzniklých v jiných modelovacích nástrojích.

4.3 Analýza tvorby seznamu rysů

Druhý krok reprezentuje srdce FDD – seznam rysů. FDD definuje tři úrovně seznamů. První úroveň jsou rysy, k nim nadřazené množiny rysů a nejvýše postavené oblasti rysů. Metodika FDD přímo nespecifikuje, v jakém pořadí mají být tyto úrovně sestavovány. Zda sestupně nebo vzestupně. V žádné ze studovaných publikaci nebylo řečeno, zda se má nejprve sestavit podrobný seznam rysu, ty sdružit do množin a ty následně do oblastí. Nebo zda je správná cesta sestupná a to první rozdělit problematiku na oblasti, ty následně na množiny rysů a z množin získávat až naposledy rysy. Rozdíl není pouze v přístupech, ale také v možné izolované existenci rysu nebo množiny rysu bez příslušnosti k nadřazenému prvku. Druhý přístup tento jev nepodporuje, zatímco u prvního jej nelze vyloučit. Není možné rozhodnout, který z přístupů je správný, proto budou v budoucím systému povoleny obě cesty a bude záležet pouze na projektovém manažerovi, kterou z nich zvolí.

4.4 Analýza plánu podle rysů

Sestavení plánu a určení milníku vývoje je charakteristickým prvkem třetího kroku FDD. V této fázi by měl být určen pevný začátek a konec projektu a další milníky týkající se rysů, množin a oblastí. V této fázi se ale skrývá další nejasnost FDD. Při tomto kroku by už podle metodiky měly být určeny předpokládané začátky a konce rysů. To by ale znamenalo podrobné naplánování celého projektu, které by už neumožňovalo přizpůsobit dostatečně projekt měnícím se požadavkům a také volnost vlastníků rysů určit, které rysy v jaké iteraci realizovat. Při skutečnosti vlastnictví kódu v metodice FDD by to znamenalo naplánování nejenom rysů ale i času jednotlivých členů rysových týmu po celý čas vývoje.

Tento problém byl na základě [12] vyřešen následovně. Rys je časově nenáročná část funkčnosti, která by měla být realizována maximálně v 14 dnech ale většinou je hotova během dvou až tří dnů. Proto nejsou kromě začátku a konce projektu přiřazeny žádné pevné termíny, ale intervaly, ve kterých by měl byt rys realizován. Projekt je tedy dle požadavků projektového manažera rozdělen na intervaly a v době plánování jsou jednotlivé rysy do těchto intervalů přiřazeny. Vlastníci rysu se pak mohou sami rozhodnout, které rysy a v jakém pořadí budou v rámci intervalu implementovány. Mohou tedy přizpůsobit výběr rysu aktuální vytíženosti vývojářů, kteří tvoří rysový tým.

4.5 Analýza návrhu a tvorby rysů

Návrh a tvorba rysů, jakožto kroky opakující se ve všech iteracích, kladou velký důraz na rysový tým a plnění stanovených plánů. Vlastník rysu potřebuje dostatečně silný podpůrný nástroj pro naplánovaní rysu, a to především práce jednotlivých vývojářů, kteří mohou být zároveň zapojeni ve více týmech. Vlastník třídy může mít v rámci realizace rysu za úkol implementovat celou třídu, ale také pouze určitou metodu nebo skupinu metod třídy. Proto bude pro potřeby systému zaveden pojem dílčí práce na rysu. Dílčí práce reprezentuje právě popis toho, co má konkrétní člen v rámci své třídy implementovat. Tento pojem se bude vázat vždy pouze k jedné třídě a tedy i k jednomu členu týmu.

V rámci realizace rysu metodika FDD určuje stálé sledování pokroku při vývoji, ale už nic neříká o tom, jak má být pokrok měřitelný. Zda míru pokroku jednotlivých rysů určují v procentech sami vlastníci rysu globálně nebo zda ji určují i ostatní členové týmu. V realizovaném systému bude využit kompromis. Tedy vlastník rysu zadá údaje o tom, kdo má realizovat jakou dílčí práci a v jakém časovém období. V rámci dílčí práce už samotní vývojáři určují, z kolika procent je dokončena. Po označení splnění dílčí práce na 100% validuje a verifikuje odvedenou práci ještě vlastník rysu a až on práci na rysu uzavře.

5 Požadavky

Při analýze požadavků na výsledný informační systém je vycházeno ze snahy pokrýt co nejlépe potřeby takovéhoto typu aplikace. Je následována specifikace metodiky FDD a jednotlivé požadavky jsou uvedeny nejprve přehledným popisem a následně je uveden diagram případů použití pro jednotlivé typy uživatelů přistupujících k systému.

5.1 Výčet požadavků

Výsledný informační systém musí plně pokrývat všechny zásady a doporučení metodiky FDD. Systém bude implementován jako webová aplikace s přehledným a uživatelsky příjemným ovládáním.

5.1.1 Nefunkční požadavky

- Systém bude implementován jako webová aplikace

5.1.2 Funkční požadavky

5.1.2.1 Systém pro více společností

- K systému bude mít možnost pod cetralizovanou správou přistupovat více společností. Každá společnost bude mít vytvořen svůj datový prostor a bude odstíněna od ostatních společností.
- K systému se bude přistupovat přes přihlašovací údaje a po přihlášení bude identifikováno, do jaké společnosti uživatel patří a budou mu poskytnuta dle jeho práv data a funkce společnosti.
- Společnosti budou mít možnost přidávat, mazat a editovat centrální systémoví správci. Tito systémoví správci založí společnost, určí jí tzv. doménového administrátora, který již bude řídit a spravovat data a funkčnost společnosti.
- Systémoví administrátoři nebudou mít přístup k datům jednotlivých společností.

5.1.2.2 Role a práva v systému

- V systému bude vytvořeno několik rolí, které se budou následně přidělovat jednotlivým uživatelům.
- Každá role bude definovat, jaké práva daný uživatel v systému má a určovat přístup k jednotlivým sekcím aplikace.

- Bude definována množina práv, které bude možné dynamicky přidělovat jednotlivým rolím.
- Jeden uživatel v rámci společnosti bude moci sdílet více rolí s různou úrovní práv.
- Definovat nové role či případně modifikovat práva rolí pro celý systém budou mít pouze systémoví administrátoři a tyto role se svými právy budou platné pro všechny společnosti.
- V rámci každé společností bude možné definovat nové uživatele a přidělit jim role v rámci celé společnosti. O jednotlivých uživatelích budou uchovávány osobní údaje, které bude mít správce společnosti možnost kdykoliv modifikovat.
- Každý uživatel bude nabývat určitých práv v rámci celé společnosti, ale tyto práva se nebudou přenášet do projektů. Uživatelům se budou nově definovat role, které budou platné pouze v rámci projektu
- Každý uživatel bude mít přiřazen libovolný počet rolí jak v rámci projektu, tak v rámci společnosti.

5.1.2.3 Projekty

- V každé společnosti bude podle potřeby možné zakládat nové projekty a zadávat jim všechny potřebné informace dle metodiky FDD.
- Pro každý projekt bude možné tvořit projektový tým. To znamená, že k projektu budou přiřazeny nové nebo již existující osoby ze společnosti a budou jim definovány role plané pro tento projekt.

5.1.2.4 Podpora vývoje podle FDD

- Systém musí pokrývat všech pět kroků, které uvádí metodika FDD při vývoji software.
- Pro každý projekt bude vytvořen dokumentový sklad, kde budou mít uživatelé možnost vytvářet složky a uchovávat potřebné soubory.
- Pro pokrytí prvního kroku metodiky musí být možné evidovat celkový model vyvíjeného systému, který budou moci prohlížet vybraní členové projektového týmu
- Druhý krok metodiky bude podporovat tvorbu hierarchického seznamu rysů skládajícího se z oblastí rysů, množin rysů a rysů samotných.U rysů, množin i oblastí musí být možné definovat základní údaje, stejně jako odpovědného vlastníka a milníky pro daný vývoj.
- V rámci třetího kroku metodiky bude možné sestavit plán projektu.
- Vlastníkovi rysu bude umožňovat naplánovat rysy v rámci jedné iterace a sestavit rysové týmy. Vlastník rysu zvolí, kteří členové projektu vytvoří rysový tým a přidělí jim role v rámci daného rysu.

5.1.2.5 Vedení rysu

- Bude implementováno přehledné zobrazení prací na rysech a jejich případné přeplánováni.
- V rámci rysu bude možné sledovat vývoj a plnění předepsaných termínů vývoje.

5.1.2.6 Podpora pro vývojáře

- Každý vývojář bude mít k dispozici osobní sekci, kde bude mít zobrazeno, jaké dílčí práce a v jakých termínech má splnit pro určité rysy.
- Vývojář bude moci editovat, zda už s prací začal a kdy ji dokončil. V řádech procent bude uvádět svůj pokrok v přidělené práci na rysu.
- Systém bude poskytovat přehledný výčet prací a bude hlídat, zda na nějaké práci nedochází ke zpoždění.
- Práce budou nabývat čtyř různých stavů a to: práce nebyla zahájena, práce byla zahájena a probíhá podle plánu, v práci dochází ke zpoždění a práce byla dokončena.
- Jednotlivé stavy rysu nebo práce na rysu budou v přehledu barevně odlišeny a vlastník rysu bude mít možnost změny jejich stavu.

5.1.2.7 Sledování pokroku ve vývoji

- Systém musí poskytovat potřebné mechanismy pro sledování pokroku na vývoji projektu.
- Musí sledovat plnění stanovených milníku a upozornit odpovědné osoby pokud dochází ke zpoždění nebo jiným problémům v projektu.
- Sledování pokroku projektu budou umožňovat reporty, které bude možné generovat ve formátu HTML, PDF a RTF pro potřeby prezentace stavu projektu cílovému zákazníku.

5.2 Případy použití

Jedním z požadavků na výslednou aplikaci je i možnost definování nových rolí. Každé konkrétní roli mohou příslušet jiná práva, která definují oprávnění pro přístup k jednotlivým funkcím a datům systému. V systému musí být pevně zavedeny pouze dva konkrétní aktéři: systémový administrátor a správce společnosti. Ostatní role a jejich práva bude možné dynamicky měnit. Proto budou v této kapitole uvedeni pouze tzv. navržení aktéři, kteří odpovídají tomu, jak jsou rozděleny pravomoci jednotlivých účastníků vývoje v rámci FDD. Tyto role plynou z analýzy metodiky FDD a měly by pokrývat základní role členů vývojového týmu, uvedené v kapitole 2.4.1, pracujícího podle principů metodiky. Podpůrné a dodatečné role FDD je možné dodefinovat pomocí založení nové role a přiřazení požadovaných práv. Při následujícím výčtu rolí ale nejsou uvažovány.

Při sestavování případů použití je třeba vzít také v úvahu, že pravomoci jednotlivých rolí v rámci FDD se velmi často překrývají a jeden případ použití může být sdílen více rolemi. S ohledem

na tuto skutečnost jsou jednotlivé případy použití generalizovány pro abstraktní aktéry, u kterých je uvedeno jaké role mediky FDD zastřešují.

Pro zachování přehlednosti této kapitoly mohou být případy vytvoření, editace a odstranění určité entity sjednoceny pod jeden případ použití. Pro tuto trojici bude v této kapitole používán zástupný případ použití pod jednotným názvem, a to evidence určité entity.

5.2.1 Aktér Uživatel

Aktér Uživatel představuje nadřazeného aktéra, který je rodičem všech ostatních aktérů. Všichni další aktéři vzniknou specializací aktéra Uživatel a dědí tedy případy použití u něj uvedené. Uživatel představuje v systému abstraktní roli každého aktéra při přihlášení a odhlášení ze systému. U dalších aktéru systému bude jako vstupní podmínka pro všechny jejich případy použití požadováno přihlášení uživatele do systému.

PŘIHLÁŠENÍ UŽIVATELE

Vstupní podmínky: uživatel není přihlášen do systému; *cíle:* přihlášení do systému a specializace uživatele na základě přidělených rolí; *postup:* uživatel přistoupí na úvodní stránku aplikace a do přihlašovacího formuláře zadá údaj o loginu a heslu. Na základě těchto informací provede systém autentizaci uživatele. Po úspěšné autentizaci má uživatel přístup k privátním stránkám aplikace. Pokud autentizace nedopadla úspěšně, je uživateli zobrazeno chybové hlášení a nabídnuta možnost opětovného přihlášení.

ODHLÁŠENÍ UŽIVATELE

Vstupní podmínky: uživatel musí být přihlášen do systému; *cíle:*bezpečné ohlášení ze systému; *postup:* na základě této akce je uživatel odhlášen ze systému.

PROHLÍŽENÍ SVÉHO PROFILU

Vstupní podmínky: uživatel musí být přihlášen do systému; *cíle:*zobrazení informace o vlastním profilu (osobní údaje, účast v projektech, přiřazené role atd.); *postup:* aktér Uživatel zvolí požadovanou akci a zobrazí se mu detail svého profilu.

EDITACE VLASTNÍCH OSOBNÍCH ÚDAJŮ

Vstupní podmínky: uživatel musí být přihlášen do systému; *cíle:* modifikace vlastních osobních údajů včetně možnosti změny hesla; *postup:* po volbě této položky má aktér možnost změnit své osobní údaje. Pokud zadá i volbu změny hesla, je vyzván k zadání stávajícího a vyplnění hesla nového. Nové heslo musí být do systému pro kontrolu zadáno dvakrát. Pokud uživatel zadá špatně staré heslo nebo nové heslo není zadáno dvakrát stejně, pak je zobrazeno upozornění a uživatel má možnost zadat údaje znova.

5.2.2 Aktér Systémový administrátor

Aktér Systémový administrátor představuje správce rolí a společností, které mají v rámci systému vytvořenou doménu.

VYTVOŘENÍ NOVÉ ROLE

*Vstupní podmínky:*žádné; *cíle:*vytvoření nové role pro potřeby řízení dle metodiky FDD; *postup:* Systémový administrátor zadá požadavek na vytvoření nové role a definuje jméno a identifikátor této role. Jako identifikátor může sloužit libovolný řetězec znaků. K roli přiřadí z nabídnutého výčtu potřebná práva

MODIFIKACE ROLE

Vstupní podmínky: žádné; *cíle:*změna jména role a přidání či odebrání prav dané roli; *postup:* Po volbě role ze zveřejněného seznamu administrátor modifikuje dostupné údaje a změní práva přiřazená dané roli

ODSTRANĚNÍ ROLE

Vstupní podmínky: žádné; *cíle:*odebrání role ze systému a následné odebrání této role u všech uživatelů; *postup:* administrátor zvolí akci smazaní role. Po upozornění, že budou odstraněna tato role u všech uživatelů, dojde k odebraní této role ze systému

EVIDENCE SPOLEČNOSTI

*Vstupní podmínky:*žádné; *cíle:*založení nové společnosti, modifikace údajů o společnosti a smazání společnosti; *postup:* Je zvolena akce přidání společnosti, modifikace společnosti nebo smazání společnosti. Pokud bude společnost smazána budou ztraceny i všechny informace o jejích uživatelích a datech.

EVIDENCE AMINISTRÁTORŮ SPOLEČNOSTI

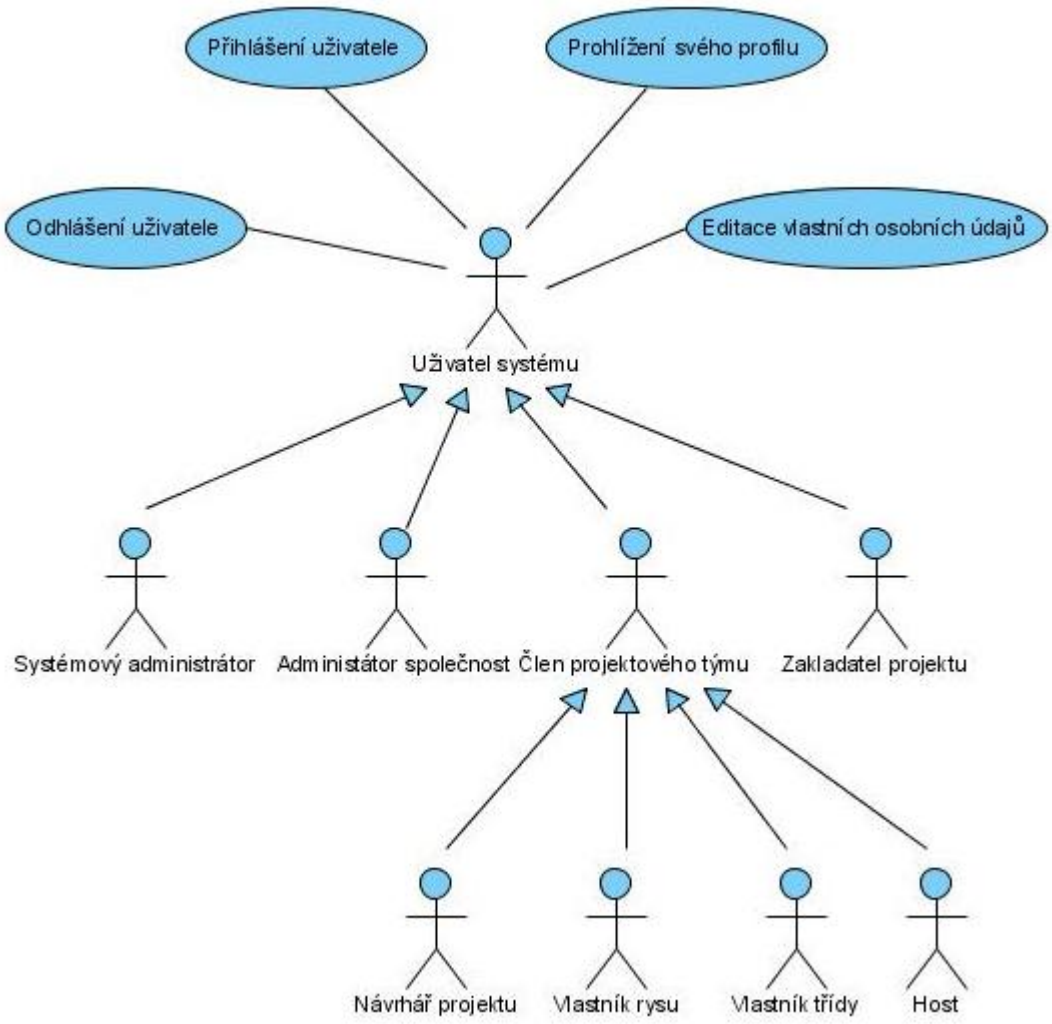
Vstupní podmínky: existující společnost; *cíle:* vytvořit, modifikovat nebo zrušit administrátory společnosti; *postup:* Po volbě konkrétní akce jsou pro společnost přidáni, editováni nebo smazáni administrátoři.

5.2.3 Aktér Administrátor společnosti

Role, která je přiřazena správci domény společnosti. Tento aktér má možnost definovat nové členy, editovat a mazat stávající členy společnosti v systému. Je definován systémovým administrátorem a personalistiku v rámci systému společnosti.

EVIDENCE UŽIVATELŮ V RÁMCI SPOLEČNOSTI

Vstupní podmínky: existující společnost; *cíle:* vytvořit, modifikovat nebo zrušit členy společnosti; *postup:* Po volbě konkrétní akce jsou pro společnost přidáni, editováni nebo smazáni její členové.



5.2.4 Aktér Správce projektu

V metodice FDD nabývá podoby tohoto aktéra role projektového manažera. Jedná se o agendu kolem založení, editace a mazaní projektů. Zahrnuje mimo jiné i personální vedení celého projektu.

ZALOŽENÍ PROJEKTU

Vstupní podmínky: žádné; *cíle:* vytvořit nový projekt v rámci společnosti; *postup:* Po volbě konkrétní akce je v rámci společnosti vytvořen projekt. Při založení nového projektu tento aktér zadá základní informace včetně stanovení klíčových milníků v rámci projektu.

MODIFIKACE/SMAZÁNÍ PROJEKTU

*Vstupní podmínky:*žádné; *cíle:* modifikovat nebo odebrat projekt; *postup:* Po volbě konkrétní akce je v rámci společnosti odebrán nebo modifikován projekt. V rámci modifikace se rozumí změna údajů o projektu včetně úpravy dat milníků.

PŘIŘAZENÍ/ODEBRÁNÍ EXISTUJÍCÍHO UŽIVATELE K PROJEKTU

Vstupní podmínky: existující uživatelé v rámci dané společnosti a role projektového manažera pro tento konkrétní projekt; *cíle:*přiřazení abstraktní role člena projektového týmu, který se bude podílet na vývoji projektu nebo naopak odebrání této role; *postup:* Na základě žádosti o tuto operaci je zobrazen seznam všech uživatelů dané společnosti a jsou označeni ti, kteří již jsou členy týmu. Projektový manažer vybere v jednom kroku nové členy týmu nebo z týmu členy vyloučí.

VYTVOŘENÍ NOVÉHO UŽIVATELE PRO DANÝ PROJEKT

Vstupní podmínky: žádné; *cíle:* vytvořit nového uživatel pro potřeby projektu; *postup:* Pokud je zvolena položka vytvoření nového uživatele zobrazí se formulář pro zadání dat o novém uživateli. Pokud jsou zadány korektně všechny požadované údaje, je vytvořen nový uživatel pro danou společnost a zároveň se stává členem projektového týmu.

PŘIŘAZENÍ/ODEBRÁNÍ FDD ROLÍ ČLENŮM TÝMU

Vstupní podmínky: žádné; *cíle:* přiřazení nebo odebrání rolí metodiky členům projektového týmu; *postup:* Po výběru konkrétního člena týmu je v rámci jedné webové stránky zobrazen seznam všech rolí, kterých může uživatel nabývat. Aktér správce projektu vybere ty role, které budou členu přiřazeny nebo na druhou stranu role, které budou odebrány. Po odeslání těchto údajů jsou role uživatele v rámci konkrétního projektu modifikovány.

SLEDOVÁNÍ POKROKU PROJEKTU

Vstupní podmínky: žádné; *cíle:* zobrazení stránky s informacemi o pokroku projektu; *postup:* Po zvolení projektu je zobrazena stránka s procentuálním grafickým vyjádřením pokroku na projektu. Zároveň je zobrazena tabulka s výčtem rysů, kde je uveden pokrok a stav jednotlivých rysů. Na základě jejich stavu (Akce na projektu dosud nezačaly, zpoždění rysu atd.) jsou jednotlivé rysy barevně odlišeny.

ZOBRAZENÍ REPORTŮ

Vstupní podmínky: žádné; *cíle:* zobrazení reportů o stavu pokroku na projektu. Tyto reporty musí být možné generovat ve formátech HTML, PDF a RTF; *postup:* Po výběru sekce reportů je zobrazen seznam dostupných a odkazy na dokumenty s reporty v různých formátech. Aktér má možnost si reporty stáhnout nebo zobrazit v daném webovém prohlížeči.

5.2.5 Aktér Člen projektového týmu

Tento aktér zastupuje všechny role metodiky FDD. Jednotliví uživatelé se stávají tímto aktérem, pokud jsou správcem projektu přiřazeni k vývoji konkrétního projektu. Tento aktér má přístup k informacím o projektu.

ZOBRAZENÍ INFORMACÍ O PROJEKTU

Vstupní podmínky: žádné; *cíle:*zobrazení souhrnných informací o projektu (jméno, milníky, počet rysů, členové týmu atd.); *postup:* Aktér zvolí jeden z projektů, jehož týmu je členem a zobrazí se mu dostupné informace o tomto projektu.

VYTVOŘENÍ/SMAZÁNÍ ADRESÁŘŮ V DOKUMENTOVÉM SKLADU

Vstupní podmínky: přidělené právo k adresáři nebo právo na vytvoření adresáře; *cíle:*vytvoření nebo smazání adresáře; *postup:* Aktér zvolí pozici v hierarchii dokumentového skladu. Při akci smazání adresáře, je upozorněn, že budou smazány i všechny soubory, které obsahuje. Po vytvoření adresáře se daný přidá na dané místo v hierarchii dokumentového skladu.

NAHRÁNÍ/SMAZÁNÍ SOUBORU V DOKUMENTOVÉM SKLADU

Vstupní podmínky: přidělené právo k souboru nebo právo na odstranění souboru; *cíle:*nahrání nebo smazání souboru; *postup:* Aktér zvolí adresář v dokumentovém skladu. Při akci smazání souboru, je upozorněn, že soubor bude nevratně odstraněn z adresáře. Při nahrávání souboru je po uživateli požadována správná cesta a soubor je nahrán do zvoleného adresáře.

PROHLÍŽENÍ ADRESÁŘE V DOKUMENTOVÉM SKLADU

Vstupní podmínky: přidělené právo k adresáři; *cíle:*listování v hierarchii adresáře; *postup:* procházení ve stromu souborů a adresářů, které adresář obsahuje

STÁHNUTÍ SOUBORU V DOKUMENTOVÉM SKLADU

Vstupní podmínky: přidělené právo k souboru; *cíle:*stáhnutí souboru z dokumentového skladu; *postup:* aktér vybere požadovaný soubor a zadá požadavek na jeho stáhnutí.

5.2.6 Aktér Návrhář projektu

Tímto aktérem mohou být v metodice FDD zastoupeny všechny role, které se aktivně účastní prvních dvou kroků metodiky a mají právo zaznamenávat a modifikovat výsledky těchto dvou kroku. Pro potřeby této aplikace je to projektový manažer a hlavní programátor.

EVIDENCE OBLASTÍ RYSŮ

Vstupní podmínky: žádné; *cíle:*založení, modifikace a smazání oblastí rysů; *postup:* po volbě sekce oblastí rysů je aktérovi zobrazen seznam všech existujících oblastí a nabídnuta volba nové oblasti. Při tvorbě nové oblasti musí aktér vyplnit všechny požadované údaje a pokud jsou známy tak i časové milníky oblasti. Pro editaci nebo smazání oblasti je třeba vybrat danou volbu u položek zobrazeného seznamu oblastí. Při mazání oblasti je aktér dotázán, zda si přeje odstranit danou oblast i přesto, že budou ztraceny všechny množiny rysů a rysy, které obsahuje.

EVIDENCE MNOŽIN RYSŮ

Vstupní podmínky: žádné; *cíle:*založení, modifikace a smazání množin rysů; *postup:* po volbě sekce množin rysů je aktérovi zobrazen seznam všech již vytvořených množin. Na stejné webové stránce má aktér také možnost zvolit vytvoření nové množiny. Při tvorbě nové množiny rysů musí

aktér vyplnit všechny požadované údaje a pokud jsou známy tak i časové milníky. Každý nově tvořený milník je možné zahrnout do nadřazené oblasti rysů, pokud nějaká v projektu existuje. K editaci a mazání konkrétní množiny je možný přístup přes odkazy umístněné u jednotlivých množin v seznamu. Při mazání množiny je aktér dotázán, zda si ji přeje odstranit i přesto, že budou ztraceny všechny rysy, které obsahuje.

EVIDENCE RYSŮ

Vstupní podmínky: žádné; *cíle:*založení, modifikace a smazání rysů; *postup:* po volbě sekce rysů je aktérovi zobrazen seznam všech rysů v rámci tohoto projektu. U jednotlivých rysů je zobrazena položka na smazaní a editaci rysu. Po volbě vytvoření nového rysu je zobrazen formulář, kde je mimo popisných informací o novém rysu možné zadat i nadřazenou množinu rysu a časové milníky rysu. V rámci vytvoření rysu je možné přiřadit rysu vlastníka s rolí hlavního programátora.

VYTVOŘENÍ PLANŮ PROJEKTU

Vstupní podmínky: existující seznam rysů; *cíle:*vytvoření plánu vývoje, kdy jednotlivé rysy budou přiřazeny do iteračních období; *postup:* období mezi začátkem a koncem projektu je rozděleno na dvoutýdenní intervaly. Na stránce je zobrazena časová osa, v níž je vždy aktivní pouze jeden interval. Pro daný interval je možné přiřadit rysy, které dosud nejsou zahrnuty v žádném jiném intervalu. Na stránce může aktér volit filtr pro daný interval, na jehož základě jsou zobrazeny vyhovující rysy. Pokud je rys přiřazen do plánu, tak je nastaven plánovaný a koncový datum rysu na hraniční data intervalu.

5.2.7 Aktér Vlastník rysu

Jedná se o zkušeného programátora, který se podílí na návrhu svěřeného projektu. Tento aktér může vlastnit určité rysy a za daný rys nese odpovědnost. V rámci vlastnictví rysu se stává vedoucím rysového týmu. Jméno aktéra odpovídá roli vlastníka rysu metodiky FDD

EVIDENCE DÍLČÍCH PRACÍ NA RYSU

Vstupní podmínky: žádné; *cíle:*vytvoření, editace nebo smazání dílčí práce potřebné pro implementaci daného rysu; *postup:* Po zvolení akce na plánování prací na rysu se zobrazí stránka s grafickým znázorněním posloupnosti jednotlivých prací na rysu. U každé dílčí práce je dostupná volba na editaci a smazání. V rámci editace je možné kromě informačních údajů (jméno,popis ...) změnit také příslušnost k vlastníku třídy a plánované milníky. Při volbě nové dílčí práce je zobrazen formulář, kde je nutné vyplnit třídu, které se práce týká a vlastníka třídy, který bude zodpovídat za implementaci této práce. Každá nová dílčí práce musí mít také uvedeno plánované datum zahájení a dokončení. Po odeslání vyplněného formuláře je vytvořena v rámci rysu nová dílčí práce a je uložena do seznamu prací konkrétního vlastníka tříd.

PROHLÍŽENÍ RYSOVÉHO TÝMU

Vstupní podmínky: žádné; *cíle:*zobrazení osob, které tvoří rysový tým; *postup:* Po požadavku na tuto akci je zobrazen tabulkový výčet členů týmu. U každého člena týmu je uvedeno, kolik dílčích pracích v rámci projektu zastává.

ZOBRAZENÍ PRACÍ ČLENŮ RYSOVÉHO TÝMU

Vstupní podmínky: žádné; *cíle:*zobrazení dílčích prací člena týmu v rámci rysu. Jsou zobrazeny časové milníky a plánované plnění prací; *postup:* Aktér zvolí jednoho člena týmu a u něj jsou zobrazeny všechny jeho práce. Jednotlivé práce jsou barevně odlišeny, podle stavu, který určuje, zda jsou práce pněny podle plánu. U jednotlivých prací je také zobrazeno z kolika procent jsou hotovy.

ZOBRAZENÍ POKROKU RYSU

Vstupní podmínky: žádné; *cíle:*zobrazení z kolika procent je rys hotový a jak jsou splněny dílčí práce; *postup:* Po požadavku na zobrazení pokroku rysu je zobrazeno, z kolika procent je projekt hotový a zobrazen výčet všech dílčích prací a jejich pokrokem. Pokud aktér potřebuje může pomocí přepínačů filtrovat páce podle stavu nebo vlastníka.

UZAVŘENÍ DÍLČÍ PRÁCE

Vstupní podmínky: dílčí práce je splněna na 100%; *cíle:*označení dílčí práce jako dokončené; *postup:* Aktér v rámci modifikace dílčí práce označí práci jako kompletní a tím ji uzavře.

5.2.8 Aktér Vlastník třídy

I tento aktér odpovídá roli vlastníka třídy v metodice FDD. Tento aktér představuje tvůrce kódu, který pracuje pod vedením hlavního programátora v rámci rysového týmu na realizaci rysu. Tento vývojář modifikuje pokrok v rámci svěřených tříd a má přístup k zobrazení prací ostatních členů rysového týmu.

ZOBRAZENÍ VŠECH PRACÍ NA PROJEKTU

Vstupní podmínky: žádné; *cíle:*zobrazení souhrnných seznamu všech prací na projektu; *postup:* Po požadavku na tuto akci je zobrazena tabulka s výčtem všech prací, které náleží v rámci tohoto projektu aktérovi. Jednotlivé práce jsou barevně odlišeny, podle stavu, který určuje, zda jsou práce pněny podle plánu. U jednotlivých prací je také zobrazeno z kolika procent jsou hotovy.

MODIFIKACE POKROKU A REÁLNÝCH DAT U PRACÍ

Vstupní podmínky: žádné; *cíle:*změna započetí nebo dokončení práce a procent, z kolika je práce hotova; *postup:* Aktér zvolí jednu z prací a u ní položku editace. V rámci této editace zvolí datum zahájení nebo dokončení práce a z kolika procent je práce hotova. Pokud uvede aktér datum dokončení a práce není splněna na 100% není mu toto umožněno.

5.2.9 Aktér Host

Jedná se o externího uživatele, který není zaměstnancem společnosti. V terminologii metodiky FDD představuje zákazníka nebo doménového specialistu. Stává se na přechodnou dobu členem projektového týmu.

ZOBRAZENÍ REPORTŮ

Vstupní podmínky: žádné; *cíle:* zobrazení reportů o stavu pokroku na projektu. Tyto reporty musí být možné zobrazit ve formátech HTML, PDF a RTF; *postup:* Po výběru sekce reportů je zobrazen seznam dostupných a odkazy na dokumenty s reporty v různých formátech. Aktér má možnost si reporty stáhnout nebo zobrazit v daném webovém prohlížeči.

6 Zvolené technologie

V požadavcích na výsledný informační systém nebylo definováno, jaké technologie se mají k vývoji použít. Tato skutečnost poskytla dostatečnou volnost pro výběr nejvhodnější nástrojů. Jelikož se jedná o dynamickou webovou aplikaci, která by měla být využívána v prostředí IT společnosti pro podporu vývoje, je nutné použít dostatečně silný nástroj k pokrytí všech požadavků na funkčnost.

V současnosti existuje poměrně velké množství prostředků k vývoji webových aplikaci, jak z pohledu volby programovacího jazyka tak i ze strany typu použité databáze a podpůrných nástrojů. S ohledem na objektový přístup k návrhu celé aplikace je potřebné zvolit takové technologie, které tento přístup nejenom podporují, ale také usnadňují. Zároveň je vhodné dbát na moderní přístup k vývoji webových aplikací a nepoužívat skriptovací jazyky nebo proprietární konvence značkování. Při těchto zastaralých přístupech se prolínají úseky HTML a úseky kódu ve skriptovacím jazyku. Tento styl vývoje aplikace má negativní dopad na výkon a také rapidně roste počet řádek kódu jednotlivých webových stránek. Z tohoto pohledu je výhodnější zvolit takovou technologií, která podporuje tzv. kód v pozadí (code behind). Tento přístup odděluje kód od prezentace a tak zjednodušuje údržbu webu a především odstíní design webové stránky od samotného kódu.

Po uvážení všech těchto skutečností bylo zvoleno pro vývoj ASP.NET 2.0 od společnosti Microsoft. Pro psaní kódu byl z bohaté palety podporovaných jazyků .NET zvolen populární C#. Jako webový server byl zvolen Internet Information Server (dále jen IIS) v dostupné verzi 6.0. Z možných databází bylo vybráno MS SQL a přístup přes ADO.NET.

7 Návrh

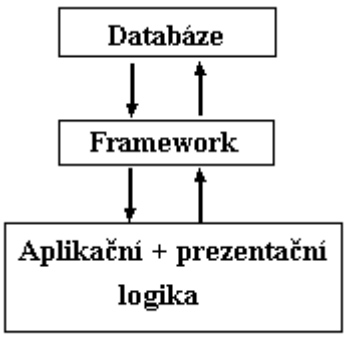
V této kapitole jsou popsány jednotlivé kroky při návrhu systému. Nejprve je popsána volba architektury vyvíjené aplikace. Další kapitola pojednává o návrhu datové vrstvy. Je zde uveden ER diagram[8] a rozbor jednotlivých entit a vazeb mezi nimi. V další kapitole je uveden podrobný diagram tříd a jejich popis.

Pro vizualizaci návrhu je použito diagramů vytvořených v jazyce UML (Unified Modeling Language, unifikovaný modelovací jazyk). Tento univerzální jazyk poskytuje dostatečné prostředky pro vizuální modelování systému.

7.1 Návrh architektury

Pro správný návrh je nejprve nutné provést architekturní rozčlenění aplikace do souvisejících celků zvaných vrstvy. Takovými vrstvami jsou logicky oddělené části aplikace jako: aplikační logika, datová vrstva, atd. Vícevrstvá architektura aplikace předurčuje lépe srozumitelný, spravovatelný a rozšiřitelný programový kód než-li monolitická aplikace. Tento přístup také přináší mnohem větší znovupoužitelný potenciál kódu a zlepšuje kohezi jednotlivých částí.

Navrhovaný systém je rozdělen do tří vrstev: datová vrstva, framework a vrstva aplikační a prezentační logiky. Umístění aplikačního rámce mezi databázi a aplikační a prezentační logiku poskytuje odstínění jednotlivých vrstev a do jisté míry nezávislost na zvolené databázi. Při změně databáze by se provedla pouze změna frameworku bez nutnosti zásahu do aplikačního kódu.



Obr. 7-1: Schéma systému

[8] ER diagram – entitně relační diagram

Mimo jiné framework zabezpečuje mapování dat relační databáze na objekty. Tím je v aplikační vrstvě zachován jednotný objektově orientovaný přístup. Každé tabulce relační databáze odpovídá ve frameworku smyluplný objekt s příslušnými atributy a metodami. Tím je aplikační logika odstíněna nejen od zvoleného typu databáze, ale i její vnitřní struktury. Při budoucí úpravě datového modelu dojde pouze ke změnám ve frameworku, které se aplikační logiky nemusejí v mnoha případech dotknout.

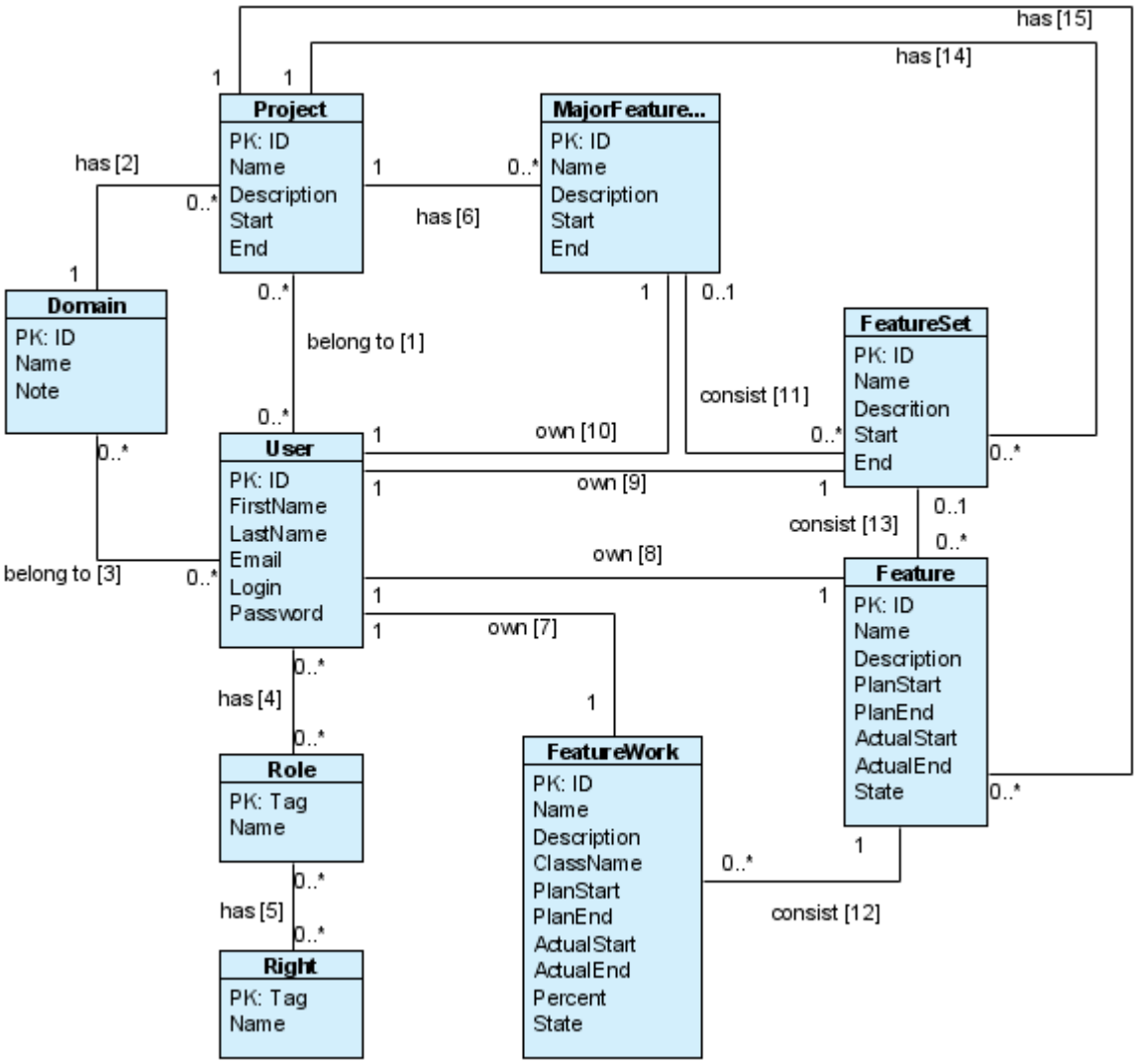
K odstínění přístupu k databázi jsou ve frameworku navrženi tzv. manažeři, kteří jako jediní znají přesnou implementaci datového modelu a jako s takovou s ní pracují. Dokáži transformovat relační data na sobě známé objekty, které podle potřeby poskytují aplikační logice k dalšímu zpracování.

7.2 Návrh databáze

V případě systému pro podporu FDD je velmi důležité zvolit správně datové struktury a metody přístupu k nim. Pro komunikaci s daty bude sloužit framework. Ten bude mapovat daná relační data do objektové podoby. Framework bude také zajišťovat veškeré dotazy nad databází. Využití frameworku pro přístup k databázi přináší nemalou výhodu pro údržbu aplikace. Pokud se totiž provedou změny v datových strukturách, tak se projeví pouze ve frameworku a není nutný zásah do aplikačního kódu. Aplikační kód tak bude zcela odstíněn od datové vrstvy a bude pracovat až s objekty předávanými frameworkem. V následující podkapitole je uveden ER diagram schématu databáze a popis jednotlivých entit.

7.2.1 ER diagram

Z ER diagramu uvedeném na obr. 7-2 je patrné, že mezi jednotlivými entitami je velmi často kardinalita 1:N nebo N:M. Tyto násobnosti u jednotlivých entit jsou z důvodu provázanosti všech údajů v metodice FDD. U jednotlivých entit jsou aktuálně uvedeny pouze základní atributy potřebné k pokrytí nejdůležitějších principů metodiky FDD. K celému datovému modelu bylo ale přistupováno s myšlenkou pozdějšího rozšíření aplikace pro potřeby společností, které by systém využívaly.



Obr. 7-2: E-R diagram

7.2.1.1 Entita User

V systému je třeba zdůraznit především entitu *User*, která představuje uživatele systému. Tato entita je centrální tabulku databáze a je provázána se všemi ostatními entitami. Tyto vazby jsou zaručují odpovědnost jednotlivých uživatelů za prvky seznamu rysů nebo příslušnost ke společnosti a jejím projektu. Zde je také pěkně vidět, jak důležitou roli hraje při metodice FDD člen vývojového týmu.

Každý uživatel je jednoznačně identifikován svým loginem, ale pro přehlednost je mu přiřazeno i identifikační číslo. Plus jsou uchovávány jeho osobní údaje.

Entita *User* může náležet k libovolnému počtu domén společností (vazba č. **3**). Pokud uživatel není v žadné doméně jedna se o uživatele na úrovni správy domén. Aktuálně je v systému dovoleno, aby uživatel patřil pouze k jedné společnosti, ale toto omezení neplatí na tabulky databáze. Zejména pro další rozšíření aplikace je umožněna příslušnost k více společnostem.

Z vazby na projekt (vazba č. **1**) vychází, že uživatel může být členem týmu v několika projektech a jeden projekt může mít více uživatelů.

7.2.1.2 Entita Domain

Entita *Domain* představuje doménu společnosti. Jsou zde uloženy základní informace o společnosti. Ke každé společnosti náleží libovolný počet uživatelů, kteří mohou v rámci společnosti mít několik různých rolí (vazba č. **4**). V doméně společnosti je možné vytvářet projekty, které jsou výchozím bodem metodiky FDD.

7.2.1.3 Entita Project

Identifikuje projekt, který je založen v rámci jedné společnosti. Projekt nemůže v systému existovat samostatně, ale musí být vždy v rámci společnost (vazba č. **3**) U projektu jsou zaznamenány milníky jako je začátek a konec a další informační údaje.

7.2.1.4 Entita Role

Entita reprezentuje role, kterých může uživatel nabývat v rámci systému, společnosti a projektu. O roli je uchováno pouze její jméno a tag, který ji jednoznačně identifikuje v rámci systému. Role má vazbu na práva v systému a jedna role může mít přiřazen libovolný počet práv (vazba č. **5**).

7.2.1.5 Entita Right

Právo, které představuje entita Right, odpovídá povolení na určitou funkcionalitu nebo stránku systému. Právo je jednoznačně identifikováno svým jménem a tagem.

7.2.1.6 Entita MajorFeatureSet

Entita MajorFeatureSet je nejvyšší položkou při vytváření seznamu rysu. Je to oblast množiny rysu. O každé oblasti je uchováváno její jméno, popis a plánované milníky. Tuto entitu vždy vlastní právě jeden uživatel systému (vazba č. **10**).

7.2.1.7 Entita FeatureSet

FeatureSet představuje entitu, která odpovídá množině rysu v metodice FDD. Tato množina je hierarchicky podřazena oblasti rysů. Entita nese informace o jménu, popisu a plánovaných milnících. Tuto entitu vždy vlastní právě jeden uživatel systému (vazba č. **9**).

7.2.1.8 Entita Feature

Entita Feature reprezentuje základní stavební jednotku metodiky FDD a to rys. V rámci rysu se uchovává informace o jeho jménu, popis a časové milníky. U rysu už jsou zaznamenávány jak plánované milníky, tak ty skutečné. Metodika definuje, že rysu musí být vždy přiřazen vlastník (vazba č. **8**).

7.2.1.9 Entita FeatureWork

FeatureWork je dílčí prací v rámci jednoho rysu. Tato dílčí práce představuje jeden úkol na implementaci rysu. Proto je nutné kromě základních údajů (jméno, popis) zaznamenat i jméno třídy, ke které se práce váže a vazbu k vlastníkovi této třídy. V rámci milníku se jedná o plánovaný začátek a konec, který určuje vlastník rysu a skutečné datum začátku a dokončení práce, které k ní přiřazuje vlastník třídy. Mimo jiné je u této entity uvedeno procentuální plnění tohoto úkolu.

7.3 Diagram tříd

V diagramu tříd jsou uvedeny třídy a jejich metody, které jsou navrženy za účelem odstínění datové vrstvy od vrstvy aplikační. Jelikož se jedná o poměrně rozsáhlý model, bude v následujícím textu rozčleněn do několika menších logicky oddělitelných částí. Celkový diagram tříd je uveden v příloze 2. Dílčí diagramy se snaží zachytit podstatné myšlenky návrhu. V diagramech je globální číslování asociačních vazeb. U ostatních vazeb jako jsou generalizace a závislosti není číslování zavedeno, protože jsou vazby dostatečně názorné a není nutný jejich popis.

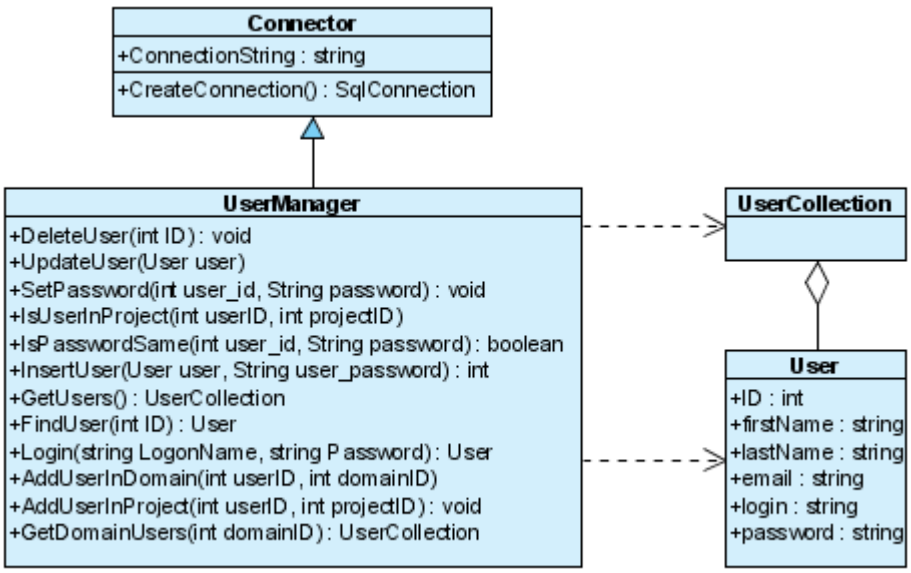
Než bude přistoupeno k popisu jednotlivých částí diagramu, je nutné vysvětlit centrální třídu, která je zobrazena ve všech diagramech. Jedná se o rodičovskou třídu *Connector*, jejímiž potomky jsou třídy *UserManager*, *SecManager*, *DomainManager*, *ProjectManager* a *FeatureWorkManager*. Tato bázová třída zprostředkovává přístup k databázi. Její atribut *ConnectionString* reprezentuje připojovací řetězec, který je nutný pro připojení k databázi. Metoda *CreateConnection* inicializuje připojení k databázi. Tato třída zabezpečuje jednotný přístup k připojení k databázi.

7.3.1 Uživatelská sekce

Třída *User* zastupuje tabulku *User* relační databáze. Tato třída je zavedena k mapování relačních dat na objekty. Vazby mezi třídou User a dalšími třídami budou uvedeny v souvislostech v ostatních částech diagramu v následujících kapitolách. Objekt tohoto typu může být sdružován v třídách *UserCollection* nebo existovat v samostatně. Třída *UserCollection* je agregaci instancí třídy *User*.

Pro objekt typu *User* existuje *UserManager*, který je potomkem třídy *Connector*. Jednotlivé metody této třídy poskytují na základě vstupních parametrů výsledky dotazů na databázi a především dotazy svázané s tabulkou *User*. Jednotlivé metody mohou vracet objekty typu *User* nebo *UserCollection*.

Pro přístup do databáze musí každá metoda inicializovat připojení k databází pomocí zděděné metody *CreateConnection*.

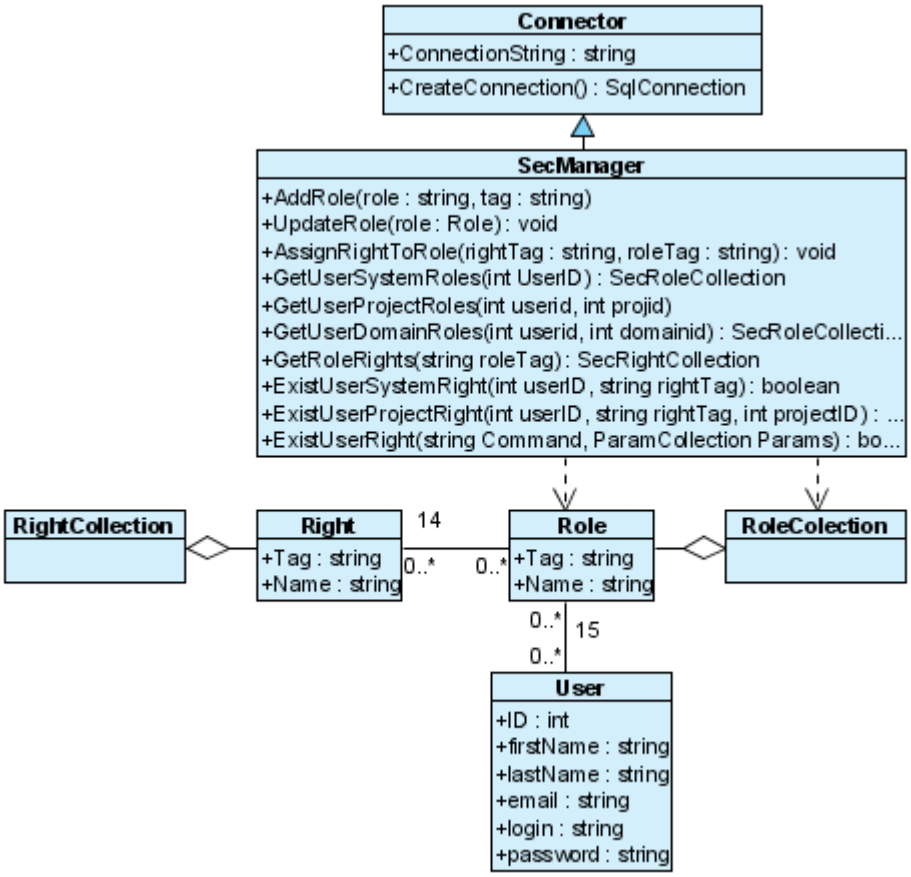


Obr. 7-3: Část diagramu tříd pro uživatele

7.3.2 Sekce bezpečnosti a autorizace

Tato část diagramu je navržena pro zajištění autorizace uživatele v systému a hlídání přístupu k jednotlivým navigačním a datovým zdrojům systému. Asociace č. **15** mezi třídou Role a třídou Right určuje agregační vazbu, v níž jedna role může obsahovat několik práv. Jak je zřejmé z diagramu, tak třída Role sdílí asociační vazbu i s třídou User. Metody třídy SecManager poskytují veškerou funkčnost pro zaručení správné autorizace v systému a správu rolí. Důležité metody jsou:

- AddRole(role string, tag string) – vytvoření nové role v databázi
- AssignRightToRole(rightTag string, roleTag string) – přiřazení práva k roli
- GetUserSystemRoles(int userID) – vrací všechny role, které má uživatel v systému
- GetUserProjectRoles(int userid, int projid) – vrací role, které má uživatel v projektu
- GetUserDomainRoles(int userid, int domainid) – vrací všechny role uživatele ve společnosti
- ExistUserSystemRight(int userID, string rightTag), ExistUserDomainRight(int userID, string rightTag, int domainID), ExistUserProjectRight(int userID, string rightTag, int projectID) – tyto metody vrací hodnotu true pokud uživatel má alespoň v jedné přiřazené roli toto právo, pokud toto právo nemá, tak vrací false



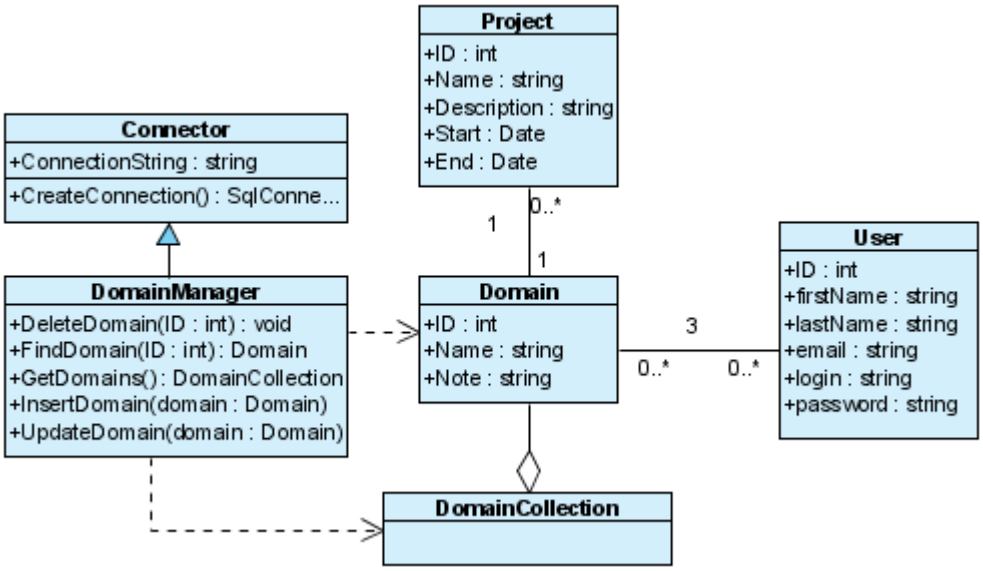
Obr. 7-4: Část diagramu tříd pro bezpečnost a autorizaci

7.3.3 Sekce domén společností

Podle této části diagramu jsou implementovány moduly domén společností. Jedná se o poměrně nenáročnou část diagramu, kdy persistentní třída *Domain* představuje společnost, jež má data uložena v tabulce *Domain*. *DomainColection* je agregací této třídy.

Třída *DomainManager* obstarává přístup k databázi a metody pro naplnění tabulky *Domain* a editaci a mazání jejích řádek. Základní metody této třídy jsou:

- InsertDomain(Domain domain) – vloží novou společnost do databáze a vytvoří pro ni nové ID
- UpdateDomain(Domain domain) – aktualizuje údaje o společnosti
- GetDomains() – vrací seznam všech společností v systému
- FindDomain(int ID) – vrací společnost podle atributu ID společnosti

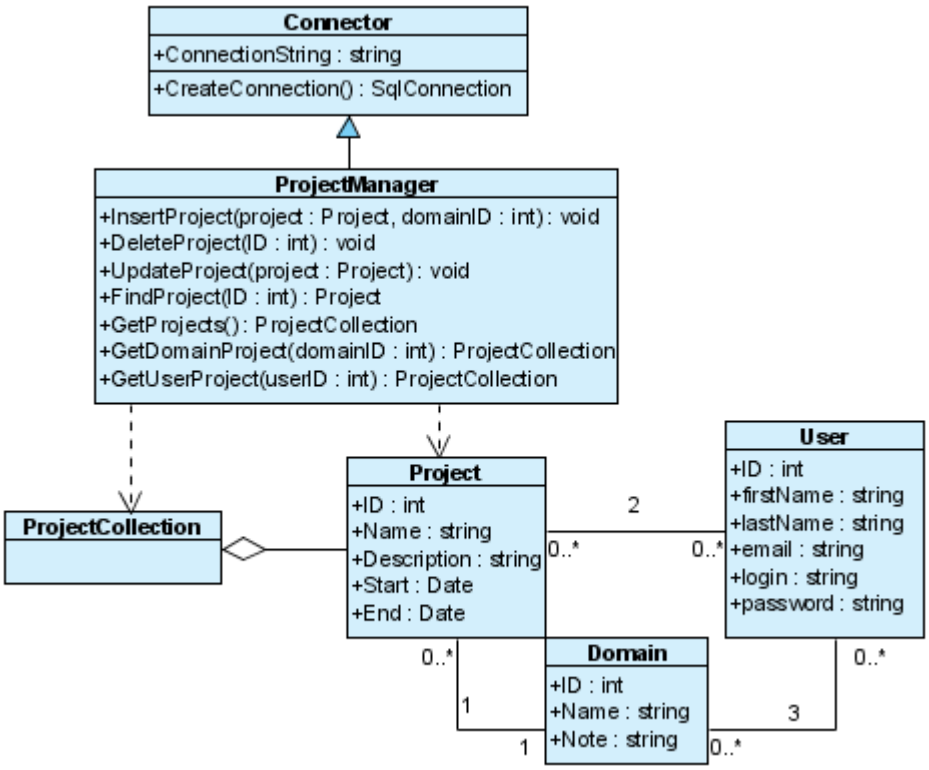


Obr. 7-5: Část diagramu pro domény společností

7.3.4 Sekce projektu

Vazby mezi jednotlivými objekty *Project*, *User* a *Domain* jsou logické a představují nutné vstupní parametry pro manager, aby dokázal vyhodnotit potřebný dotaz. Třída *ProjectManager* poskytuje základní metody pro manipulaci s objekty vyjadřující v systému entitu projekt. Seznam metod je:

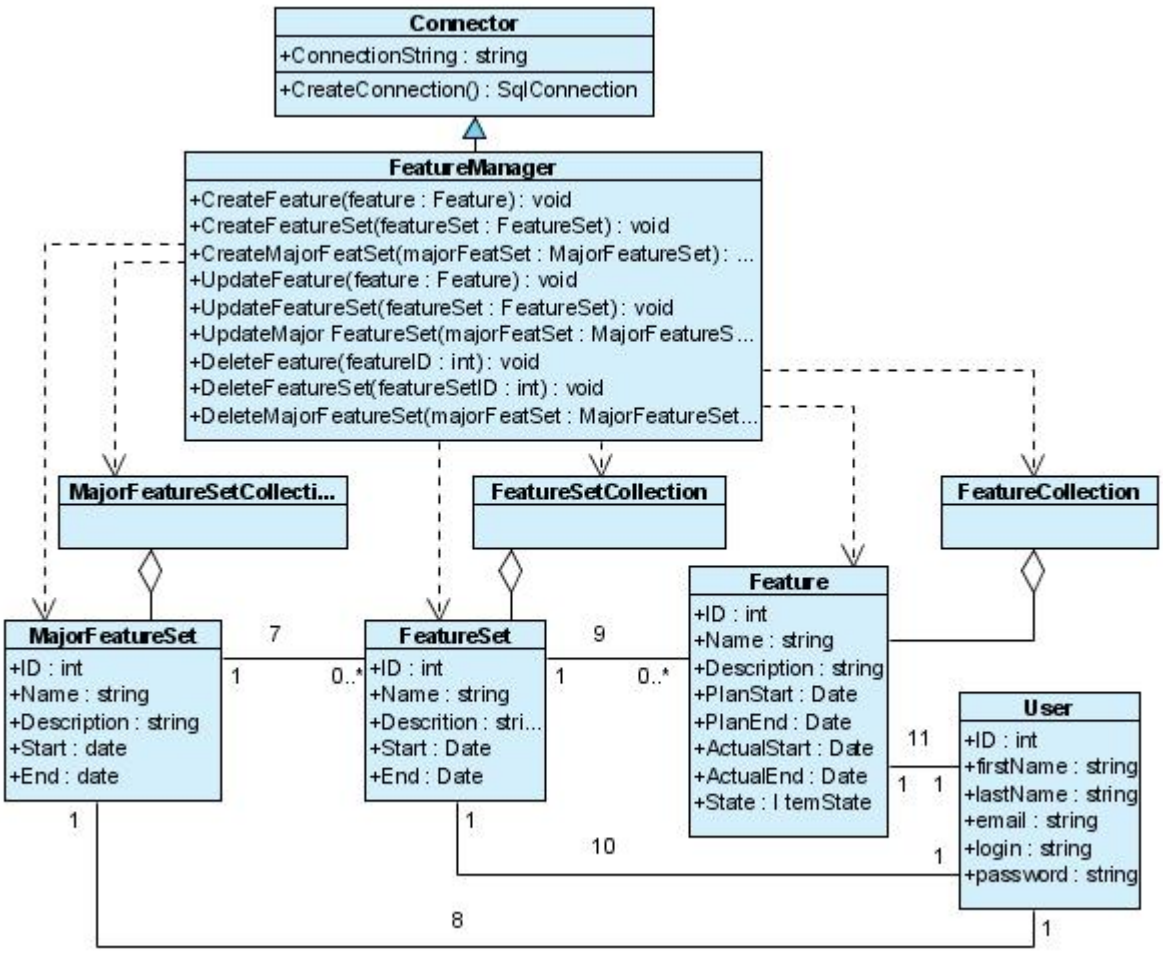
- InsertProject(Project project, int domainID) – vloží do dané společnosti nový projekt
- UpdateProject(Project project) – aktualizuje údaje o projektu
- DeleteProject(int ID) – smaže projekt s daným ID
- FindProject(int ID) – vrací projekt s daným ID
- GetDomainProjects(int domain_id) – vrací všechny projekty, které jsou v dané společnosti
- GetUserProjects(int userID) – vrací všechny projekty, na nichž spolupracuje daný uživatel



Obr. 7-6: Část diagramu tříd pro správu projektu

7.3.5 Sekce rysů

FeatureManager přestavuje jádro metodiky FDD. Z důvodu provázanosti jednotlivých entit obsluhuje nejenom rysy, ale i oblasti a množiny rysů. Schéma a vztahy mezi objekty jsou patrny z obr. 7-7. Pro jednotný přístup poskytuje FeatureManager metody, které jsou potřebné pro práci nad seznamem rysů. Není zde uveden výčet jednotlivých metod, protože jejich funkčnost je zřejmá z jejich názvu. Jsou to metody pro vytvoření, editaci nebo mazání rysu, množiny rysů a oblasti rysů.

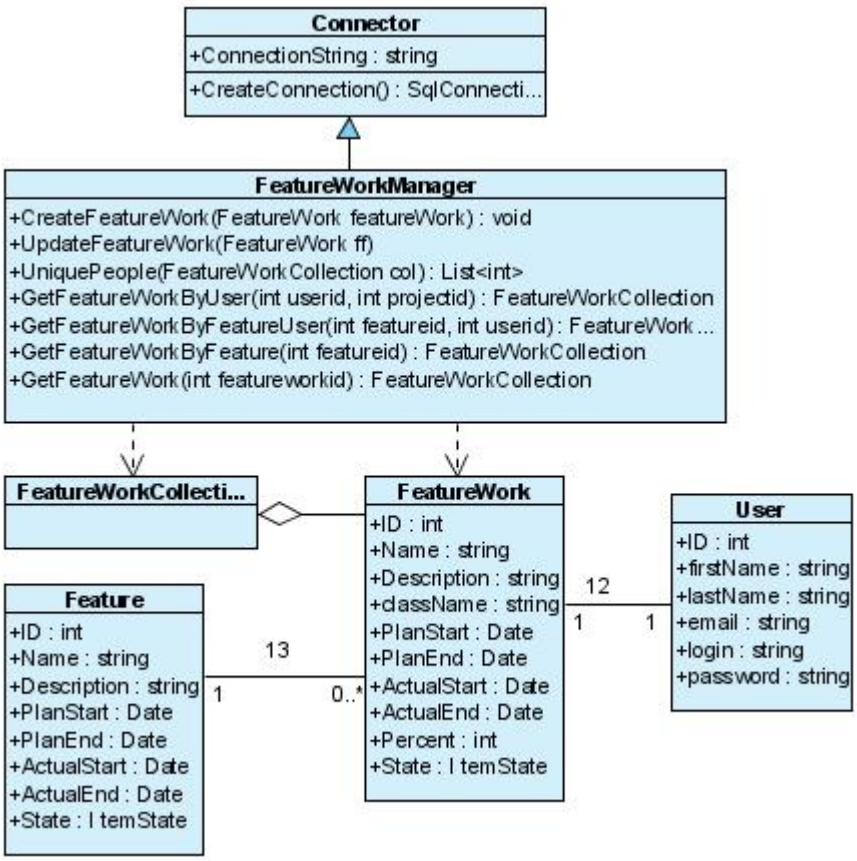


Obr. 7-7: Část diagramu tříd pro seznam rysů

7.3.6 Sekce dílčích prací na rysu

Dílčí práce má jak je vidět z obrázku č. 7-8 vztah k vlastníku a rysu, kterému náleží. Pro manipulaci s dílčími pracemi na rysu je zaveden *FeatureWorkManager*, který poskytuje tyto metody:

- CreateFeatureWork(FeatureWork featureWork) – vytvoří novou dílčí práci v určitém rysu
- UpdateFeatureWork(FeatureWork ff) – aktualizuje určitou dílčí práci v rysu
- DeleteFeatureWork(int id) – smaže určitou dílčí práci z rysu
- GetFeatureWorkByUser(int userid, int projectid) – vrací všechny dílčí práce, které uživatel obstarává v projektu. Jsou to tedy jeho práce na všech rysech v jednom projektu.
- GetFeatureWorkByFeature(int featureid) – vrací všechny dílčí práce požadovaného rysu
- GetFeatureWorkByFeatureUser(int featureid, int userid) – vrací všechny dílčí práce, na kterých uživatel pracuje v rámci jednoho rysu.
- GetFeatureWork(int featureworkid) – vrací dílčí práci s daným ID



Obr. 7-8: Část diagramu tříd pro dílčí práce rysu

Systém rolí a práv

Celý systém je navržen tak, že bude moci být v budoucnu na základě připomínek a potřeb uživatelů velmi snadno rozšiřitelný. Z tohoto důvodu není vhodné definovat uzavřenou množinu rolí, pod kterými mohou uživatelé v systému vystupovat. Navíc je nutné, aby měl jeden uživatel jiné role v rámci systému, jiné role v rámci společnosti a dalších rolí nabýval v týmu určitého projektu. Pro potřeby systému byly vytvořeny tři úrovně rolí:

- Úroveň systém
- Úroveň společnost
- Úroveň projekt

V každé úrovni platí odlišný přistup k systému rolí a práv. Jeden uživatel může být v různých úrovních vystupovat pod různými rolemi.

Na úrovni Systém je pevně definována pouze jedna role a to role administrátora systému. Tato úroveň zabezpečuje správu systému jako celku. Pokud uživatel nese roli administrátora systému není mu již pod stejnými přihlašovacími údaji dovoleno nabývat jiných rolí v dalších úrovních. Toto omezení je zavedeno z důvodu bezpečnosti, aby daný správce neměl přístup k interním datům jednotlivých společností.

Úroveň Společnost představuje interní projekty a uživatele v rámci společnosti. Jeden uživatel se zde již může vyskytovat ve více rolích. V rámci společnosti je defaultně zaveden jeden správce, který zabezpečuje personální agendu v systému společnosti. Role správce je v systému také pevně zavedena a není ji možné odebrat. Další role, které může uživatel ve společnosti zastávat je již možné dynamicky editovat a přidávat jak je uvedeno v následující odstavci.

V rámci úrovně Projekt je již umožněno definovat nové role uživatelů. Role této úrovně odpovídají rolím definovaným v metodice FDD. Defaultně jsou v systému vytvořeny základní role FDD (projektový manažer, hlavní programátor, vývojář = vlastník třídy, doménový expert a zákazník)

Dodatečné a podpůrné role pro úroveň Projekt je již možné dodefinovat na úrovni Systém. Pro každou stránku nebo ovládací prvek je v databázi vytvořeno právo. Pokud je v roli toto právo povoleno, pak je mu umožněn přístup k dané položce nebo stránce systému. Tato skupina práv je na úrovni programu a žádný uživatel ji nemůže změnit. Z tohoto důvodu je možné pro úroveň společnost a úroveň projekt definovat nové role a přiřadit jim požadovaná práva.

7.4 Systém výkazů pokroku

Metodika FDD zdůrazňuje časté podávání zpráv o pokroku ve vývoji, zasílání těchto zpráv vyššímu managementu i zákazníkům a celkově možnost monitorování vývoje projektu. Pro dodržení těchto principů musí být v systému navržen propracovaný systém výkazů (reportů), které poskytnou všem zúčastněným stranám dostatečně silné a názorné prostředky vizualizace pokroku. V systému jsou pro tyto účely navrženy čtyři typy výkazů.

7.4.1 Park diagram (Project Park Diagram)

Tento výkaz je uveden v samotné specifikaci metodiky FDD. Přistupuje k monitorování pokroku z hlediska oblastí rysů a množin rysů. Každá množina rysů představuje jeden obdélníkový prvek výkazu. V rámci tohoto prvku jsou uvedeny podřazené množiny rysu a jejich plnění. Podrobněji je struktura tohoto typu výkazu popsána v kapitole 2.7.2.

7.4.2 Graf pokroku projektu (Project development roadmap)

Výkaz, který obsahuje graf znázorňující plánované a skutečné procento dokončených rysů v časovém intervalu vývoje projektu. Na ose X tohoto grafu jsou časové údaje a na ose Y procentuální plnění rysu. Osa X má počátek v začátku projektu a konec v plánovaném datu dokončení. Osa Y má počátek v 0% a konec ve 100%. Do grafu je na jedné křivce vyneseno, kolik bylo naplánováno procent dokončených rysů k určitému datu a na druhé křivce kolik procent rysů bylo ve skutečnosti dokončeno. Jednotlivé křivky jsou barevně odlišeny.

7.4.3 Výkaz celkového pokroku (Summary progress)

Tento report poskytuje přehledné zobrazení toho, kolik rysů se v rámci každé množiny rysů vyskytuje a v jakém stavu. Jako forma tohoto výkazu byla zvolena prezentace dat v tabulkách. Každá tabulka představuje jednu oblast rysů a je nadepsána jejím jménem. Jednotlivé řádky tabulky jsou podřízené množiny rysů této oblasti. Ve sloupcích už je pak uvedeno kolik rysů se vyskytuje v daném stavu odpovídajícímu sloupci tabulky.

7.4.4 Výkaz vývojového trendu (Trend report)

Výkaz tohoto typu se vztahuje k jednotlivým milníkům vývoje. Milníky představují dny, pro které byly jsou revize na projektu. Tyto revize jsou plánovány vždy po skončení jedné iterace, která odpovídá intervalu v rámci plánu. U každé revize je uveden celkový počet rysů v projektu a procento již hotových rysů.

7.5 Návrh uživatelského rozhraní

Při návrhu uživatelského rozhraní jsou zohledněny především požadavky na logické členěni jednotlivých sekcí webové aplikace, intuitivní ovládání a názorné zobrazení dat. Proto je systém také navržen s velmi jednoduchým designem s důrazem na přehledné strukturování a ovládání aplikace. Z tohoto důvodu bylo rozhodnuto, že systém bude rozčleněn do více sekcí, které budou vždy zastřešovat určitou funkcionalitu a do každé sekce budou mít možnost vstoupit pouze určití uživatelé. Systém bude rozčleněn do těchto sekcí:

- **Správa rolí -** v této sekci jsou sdruženy veškeré ovládací prvky, které jsou poskytnuty systémovému manažerovi ke správě uživatelských rolí. Jedná se o možnost prohlížení existujících rolí, změnu jejich práv a tvorbu rolí nových.
- **Správa společností** - do sekce správy společností má přístup pouze systémový administrátor, který může definovat nové společnosti a spravovat již existující. Pro společnosti je zde umožněno vytvořit také administrátory společnosti.
- **Správa uživatelů** - tato sekce se dělí do tří dalších podúrovní podle uživatelů, kteří se mají spravovat. Uživatelé jsou rozděleni na uživatele určité společnosti, uživatele spolupracující na projektech a správce společností.
- **Profil uživatele** - sdružuje osobní údaje o přihlášeném uživateli a poskytuje možnost jejich editace
- **Správa projektů** – zde jsou soustředěny všechny projekty jedné společnosti. U každého projektu je možno přistupovat k projektovému týmu a definovat nové členy nebo přiřadit stávající členy společnosti k projektu. Pro každého člena projektového týmu je možné zadat, které role bude v rámci vývoje projektu zastávat.
- **Pracovní plocha projektu** – pro tuto sekci se musí určit projekt, se kterým se bude pracovat. Právě tady je soustředěna podpora metodiky FDD. Jsou zde koncentrovány všechny mechanismy pro vývoj vedený pomocí FDD. Tato sekce bude dále strukturována na: obecné informace o projektu, dokumentový sklad projektu, seznam rysů, plánovaní vývoje, osobní kartu (obsahuje úkoly související s vlastnictvím rysu nebo s dílčími úkoly na rysu přihlášeného uživatele) a výkazy pokroku projektu.

8 Implementace

V kapitole implementace je popsáno, jak byla aplikace vyvíjena a jaké problémy bylo nutné řešit. Jsou uvedeny jednotlivé podpůrné technologie a knihovny, které byly při realizaci aplikace využity. Kapitola nemá za cíl popsat způsob programování webové aplikace nad platformou .NET, ale spíše vytknout zajímavé okamžiky vývoje.

Při implementaci bylo využíváno principů objektově orientovaného programování a byly dodržovány moderní přístupy k tvorbě webové aplikace. Bylo důsledně dodržováno oddělení prezentačního a aplikačního kódu.

Pro implementaci bylo využito vývojové prostředí Visual Studio 2005, SQL Server Management Console a webový server IIS 6.0.

8.1 Vzhled a navigace webové aplikace

8.1.1 Unifikovaný vzhled

Při budování této aplikace byl zvolen jednotný vizuální styl. Z tohoto důvodu bylo nutné zavést standardizované formátování jednotlivých prvků aplikace a k tomu bylo využito CSS (Cascading Style Sheets). CSS poskytují řešení pro formátování webových stránek, které přesahuje hranice platforem, pracují v součinnosti s HTML[9] a jsou podporovány prakticky všemi moderními prohlížeči. Pomocí CSS byl definovaný stylový předpis (style-sheet), který definuje sadu předem určených formátovacích nastavení.

Pro potřeby unifikovaného a jednotného layoutu stránek se využívá tzv. vzorů stránek (master pages), které poskytuje ASP.NET 2.0. Vzory stránek napomáhají vytvářet jednoduché a flexibilní rozvržení, které je možné replikovat ve všech stránkách webové aplikace. Vzory stránek jsou vlastně šablony webových stránek, které definují fixní obsah a deklarují tu část webové stránky, kde má být vkládán obsah vlastní. Tento přístup poskytuje systém pro opětovné používání šablon, ale také způsob, jakým lze šablony modifikovat. Výhodou tohoto přístupu je především to, že změna v šabloně se projeví ve všech stránkách, které ji používají.

[9] HTML4 - HyperText Markup Language , značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu.

8.1.2 Navigace

Již při návrhu uživatelského rozhraní bylo provedeno rozčlenění aplikace do sekcí. Přístup k jednotlivým sekcím je v této aplikaci realizován pomocí systému záložek. Záložky jsou velmi efektivním prostředkem pro uspořádání navigace v rámci webu. Uživatele systému znají toto rozvržení z klasických desktopových aplikací a umožňuje jim intuitivní ovládání webové aplikace. Systém záložek je vytvořen pomocí již dříve zmíněného CSS formátování.

Uživateli je poskytnuta ještě doplňková navigace pomocí tzv. drobečkové navigace. Tato kontextová navigace poskytuje uživateli informaci o stránce, na které se nachází a odkud přišel. Je to v podstatě seznam odkazů na nadřazené sekce, kde jsou jednotlivé odkazy odděleny šipkou. Pro potřeby drobečkové navigace je implementována aplikační třída *Crumb*, která uchovává informace o právě jedné stránce této aplikace. Objekty typu *Crumb* jsou podle pravidel drobečkové navigace sdružovány v třídě *History*. V rámci webové stránky je pak vždy vytvořena instance třídy *History* a pro vykreslení navigace volána její metoda *DrawNavigation*.

8.2 Přístup k datové vrstvě

Vzhledem k tomu, že všechny řídící objekty frameworku je nutno před jejich použitím inicializovat správným připojovacím řetězcem k databázi, byla v aplikační vrstvě zbudována abstraktní třída *Common* poskytující instance těchto objektů inicializované a připravené k použití. Tím se sjednotí z zjednoduší používání všech objektů frameworku poskytující rozhraní *Connector*. Připojovací řetězec k databázi je uložen v konfiguračním souboru webové aplikace web.config a třída Common k němu zná cestu.

Pokud je vyžadován libovolný dotaz na databázi, jsou z aplikační vrstvy volány metody tříd frameworku. V každé metodě je nutné inicializovat připojení k databázi a otevřít spojení k databází. Pak je proveden dotaz nad databází a vzápětí spojení uzavřeno. Není tedy udržováno perzistentní připojení k databázi a tím se snižuje zátěž databázového serveru.

Pro přístup k datům je využíváno technologie ADO.NET. Jedná se o sadu tříd, které umožňují přístup ke konkrétní databázi, vykonávání SQL příkazů a získávání dat. Je využíváno databáze SQL Serveru.

8.3 Autentizace a autorizace uživatele

8.3.1 Autentizace

Celá webová aplikace je pojata jako privátní. Není možný přístup k žádné položce adresářové struktury webu bez řádné autentizace. Prostředky ASP.NET k tomuto účelu poskytují velmi mocný a praktický nástroj, který tento požadavek splnil. Zavádějí pojem (objekt) *authorize cookie*, která je uložena na straně serveru a je spojena s životností session samotné. Přítomnost takto vytvořeného objektu signalizuje, že uživatel byl řádně ověřen a přihlášen. Pokud by objekt nebyl nalezen nebo vypršela jeho platnost, je navigace automaticky přesměrována na vytvořený přihlašovací formulář.

Pro přihlášení do systému je implementován jednotný přihlašovací formulář pro všechny uživatele systému. Po zadání přihlašovacích údajů je provedena autentizace uživatele. Po korektní autentizaci je vytvořena instance třídy *User*, představující aktuálně přihlášeného uživatele, a je uložena do session. Vzhledem k tomu, že všechny operace se provádějí v kontextu aktuálně přihlášeného uživatele je znalost jeho ID a ostatních atributů nezbytná.

8.3.2 Autorizace

Autorizace probíhá při každém požadavku na zabezpečenou stránku systému. Pod pojmem autorizace je chápáno ověření přístupových práv daného uživatele pro přístup na požadovanou stránku webové aplikace nebo pro zobrazení určitých ovládacích prvků aplikace. Proto je pro každou webovou stránku vytvořena instance třídy *SecManager* a volána její metoda *ExistSystemUserRight*, *ExistDomainUserRight* nebo *ExistProjectUserRight*, podle toho zda se jedná o právo v rámci systému, společnosti nebo projektu. Pokud tyto funkce vrátí hodnotu true, pak je uživateli zpřístupněna daná položka nebo stránka. Při návratové hodnotě false se uživateli požadované prvky nezobrazí.

8.4 Sledování stavu a pokroku projektu

Jedním z mnoha kladů metodiky FDD je možnost sledovat během vývoje pokrok a stav projektu. Nejedná se pouze o hodnocení projektu jako celku, ale také plnění oblastí, množin, rysů a dílčích prací v rámci rysu. Proto je nutné u každého evidovat z kolika procent je splněn a také v jakém se nachází stavu. Každý prvek seznamu rysů nabývá právě jednoho z těchto stavů:

- Nezahájeno (Not Started)
- Probíhající práce (In Progress)
- Varování – zpoždění, problémy při řešení (Attention)
- Dokončeno (Completed)

Z analýzy stavů vyplynulo, že nestačí pouze zobrazovat stavy zadané vlastníky objektů, ale stavy musí dynamicky upravovat sám systém (framework). Každý objekt v systému s možností evidence stavu, má implementovanou vlastnost *State* a *RealState*. Vlastnost *State* představuje stav zadaný uživatelem, naproti tomu *RealState* zobrazuje reálný stav. Stavy se liší například v situaci kdy dochází ke zpoždění. Framework porovnává aktuální datum na serveru aplikace s naplánovaným koncem, a podle toho mění stav. Některé stavy jsou tedy fiktivní a nikdy nejsou uloženy do databáze. Framework se stará o správnou prezentaci aktuálního stavu. Jeho logika pro přepnutí stavu je u entit, které mají v metodice FDD alespoň jednoho následníka tato:

- Pokud jsou všichni následníci ve stavu *Dokončeno*, pak entita nabývá také tohoto stavu.
- Jestliže je alespoň jeden následník ve stavu *Probíhající práce* a žádná práce nemá stav *Varování*, pak i tato entita nabývá stavu *Probíhající práce*.
- Pokud je alespoň jeden následník ve stavu *Varování*, pak je i rys ve stavu *Varování*.
- Entita setrvává ve stavu *Nezahájeno*, pokud všichni její následníci jsou ve stavu *Nezahájeno*.

Stav pokroku je počítán zdola nahoru. Vzhledem k faktu, že entity v metodice FDD představují ve své podstatě N-nární strom a všechny jejich listy zahrnují reálnou činnost samotných programátorů, kteří jediní dokáží skutečně měnit procenta pokroku projektu, je vyhodnocení prováděno právě u nich. Ostatní nadřazené objekty tak přebírají pokrok svých následníků a počítají jejich průměr. Všechny výpočty pro jednoduchost provádí framework a aplikační vrstva pak prezentuje dosažené výsledky. Pokrok a stav entit je zobrazován jednak barevně příslušným stavem a také procentem pokroku. U procenta pokroku je pro názornost vždy zobrazena i osa s grafickým vyjádřením

8.5 Výkazy pokroku projektu

Pro sledování vývoje a poskytování zpráv o pokroku projektu klientovi je nutné, aby v systému byl implementován důmyslný systém výkazů (reportů). Základním požadavkem na výkazy byla možnost jejich exportu do formátu PDF, RTF a HTML.

Generování různých formátů výstupu představuje vždy v systémech nemalý problém. Musí být dosaženo co nejjednotnějšího vzhledu i formátování dosažených výsledků ve všech zvolených formátech. Jako první se jistě nabízí řešení prezentace dat ve formě XML[10] a jejich transformace

[10] XML – eXtensible Markup Language je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat.

pomocí XSL, XSL-fo[11] do příslušných formátů. Toto řešení nabízí veškerou potřebnou funkcionalitu, a bylo by dosaženo jistě dobrého výsledku. Nicméně tvorba šablon XSL-fo pro transformaci je značně složitá a pracná, proto byla pro potřeby této diplomové práce zvolena Open Source knihovna iTextSharp (GPL), které poskytuje unifikovaný framework pro export a formátování dat do zvolených formátů. Tato knihovna vznikla v jazyce Java a až následně byla portována do prostředí .NET.

Použití knihovny je velmi jednoduché a intuitivní. Poskytuje bázový objekt *Document*, do kterého jsou vkládány formátovací objekty jako tabulka, obrázek, textové pole, a další. Typ formátu se volí vytvořením instance objektu, který je potomkem bázové třídy *Document* a tato konkrétní třída v sobě zahrnuje potřebné formátování výstupů.

S vědomím tohoto unifikovaného přístupu ke generování byla v systému zavedena třída *Report*. Je to bázová třída pro všechny výkazy v systému. Svým potomkům poskytuje virtuální metodu *Generate*, která má jako parametr právě instanci objektu *Document*, do které bude generování probíhat. Jednotlivé třídy, které jsou potomky třídy *Report*, tedy neznají formát, do kterého generují data. O volbu formátu se stará bázová třída a tyto třídy mají v režii pouze generování dat bez starosti o výstupní formát. Toto řešení je velice příjemné na práci a poměrně rychle je dosaženo potřebných výsledků.

Výsledné výkazy jsou vždy při požadavku uloženy na serveru do temporary adresáře a poté poskytnuty uživateli ke stažení. Výjimku tvoří online zobrazení výkazů ve formě HTML. Z takto vytvořeného vykazu je vyjmutou vše mezi tagy body a to pak vloženo do generované stránky.

8.5.1 Grafy

Pro potřeby grafických výstupů ve formě grafu byla zvolena Open Source knihovna zedGraph. Její použití podléhá licenci GPL a je plně využitelná v tomto projektu. Je využita např. při generování výkazu sledování pokroku. V něm je užito spojnicového grafu , který obsahuje dvě křivky, z nichž jedna nese informace o plánovaném plnění rysů a druhá jak byly rysy plněny ve skutečnosti. Na ose X jsou časové milníky projektu a na ose Y procentuální plnění rysu. Do grafu je pak vždy na jedné křivce vyneseno, kolik procent hotových rysů bylo pro daný datum naplánováno a na druhé křivce, kolik bylo rysů splněno ve skutečnosti. Takto je možné sledovat zda křivka plnění je pod křivkou plánu, či naopak plán předbíhá.

[11] XSL-fo - eXtensible Stylesheet Language - Formatting Objects e značkovací jazyk na bázi XML pro formátování dokumentů. XSL-FO je součástí XSL, sady W3C technologií určených pro transformaci a formátování XML dat.

9 Zhodnocení a další směr vývoje systému

Z důvodu získání objektivního názoru na kvalitu realizovaného informačního systému byl systém poskytnut nezainteresovaným jedincům k testování. Testovací skupina byla seznámena s principy metodiky FDD. Každý jedinec pak byl požádán o vedení malého fiktivního projektu. Z testování vyplynulo, že navržené uživatelské prostředí je dostatečně intuitivní a poskytuje pružnost a rychlost při navigaci.

Zvolené principy pokrývají celou šíři metodiky FDD pro vedení vývoje softwarového projektu. Poskytují pohled na vývoj jak ze strany jedince, tak ze strany celého projektového nebo rysového týmu. Podle metodiky jsou kopírovány role a potřeby jednotlivých uživatelů v systému. Ten jim poskytuje přesně takové pravomoci, jaké jsou uvedeny v metodice. Přesto nejsou vývojové týmy pevně svázány pouze metodikou a systém umožňuje přizpůsobení se jejich požadavkům.

Pro prezentaci pokroku je v systému dostupná paleta výkazů, které mohou být poskytnuty vyššímu managementu, zákazníkům anebo mohou sloužit samotným vývojářům jako motivace k plnění termínů a odvádění kvalitní práce. Tyto výkazy jsou dostupné ve všech formátech a tak na jejich základě mohou být sestavovány prezentace stavu projektů nebo být archivovány pro pozdější provedení auditů.

V budoucnu by mohl být systém rozšířen o větší podporu managementu časového rozvrhu členů týmu. Tato funkcionalita by byla přínosná především vlastníkům rysů, kteří by měli lepší prostředky pro výběr toho, který rys má být v rámci jaké třídy realizován, aby nedocházelo k situacím, že se musí čekat na nějakého vlastníka třídy s realizací dílčí práce v rámci rysu. Jistě by bylo také výhodné, aby měl v sobě systém podporu sestavování celkového modelu pomocí UML. Pro tuto možnost by bylo nutné specifikovat, jak by byla v systému uchovávána třída a jak by jí byl přiřazen vlastník.

Pro podporu verzí vznikajících v rámci každé iterace by bylo jistě užitečné začlenit do systému také systém SVN[12] pro správu zdrojových kódu. Tak by se zachovala integrita mezi jednotlivými verzemi systému a jim odpovídajícími výkazy.

Při vývoji každého produktu vznikají chyby a bylo by přínosné, kdyby byl v systému integrován nějaký systém pro jejich správu např. Bugzilla. Bugzilla je systém pro sledování chyb, požadavků na rozšíření a jiných úkolů.

[12] SVN - Subversion je software, který patří do kategorie SCM (Source Code Management) aplikací. Tyto aplikace pomáhají programátorovi udržet přehlednost projektu, na kterém pracuje.

10 Závěr

Cílem této diplomové práce bylo vytvořit informační systém pro podporu vývoje pomocí metodiky FDD. Pro implementaci systému bylo nutné nastudovat principy a myšlenky metody FDD, která si získává velkou popularitu mezi společnostmi zabývajícími se tvorbou softwaru. Tyto teoretické poznatky jsou uvedeny v úvodu práce. Zájemci pro užití a nasazení této agilní metodiky tak naleznou odpovídající údaje o metodice v celé její šíři a také možnosti jejího užití. Pro výsledný systém bylo důležité vytvořit adekvátní návrh, který by reflektoval potřeby všech účastníků vývojového týmu a přitom poskytoval možnost jednoduchého rozšíření aplikace. Právě návrh byl stěžejním prvkem celé práce. V návrhu jsou zahrnuty nejdůležitější aspekty metodiky, které pokrývají všechny kroky vývoje softwaru podle FDD.

Aktuálně existuje několik desktopových aplikací podporujících vývoj podle metodiky FDD, ale byla nalezena pouze jedna webová aplikace tohoto typu (nezahrnuje všechny kroky metodiky). Přitom právě metodika FDD vyžaduje přístup k systému mnoha externími spolupracovníky společnosti, ať už se jedná o zákazníka, doménového experta nebo externího vývojáře. Proto je pro systém tohoto typu vhodná právě webová aplikace, která umožňuje spouštět systém z libovolného počítače připojeného k síti internet pouze se znalostí přihlašovacích údajů. Tato skutečnost ukazuje přínos realizovaného systému, který zaplní volné místo na trhu aplikací pro podporu metodiky FDD a zároveň poskytne členům vývojového týmu možnost řídit své projekty a sledovat jejich pokrok prakticky nepřetržitě.

Literatura

[1] Coad, P.: *Java Modeling In Color With UML (Kapitola 6)*, Dostupné z www: http://www.petercoad.com/download/bookpdfs/jmcuch06.pdf

[2] Abrahamsson, Pekka; Salo, Outi; Ronkainen, Jussi; Warsta, Juhani: *Agile software development methods. Review and analysis*, Dostupné z www: www.inf.vtt.fi/pdf/publications/2002/P478.pdf

[3] Články o FDD dostupné z www: http://www.featuredrivendevelopment.com

[4] Arlow, J. Neustadt I.: *Uml a unifikovaný proces vývoje aplikací.* 1. vyd. Brno: Computer Press, 2003. 387 s. ISBN 80-7226-947-X

[5] Kadlec, V.: *Agilní programování*. 1. vyd. Brno: Computer Press, 2004. 278 s. ISBN 80-251-0342-0

[6] Jeff de Luca.: *The Latest FDD Processes*, Dostupné z www: http://www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf

[7] Sochor, J.: *Feature Driven Development,* Dostupné z www: http://www.fi.muni.cz/~sochor/PA103/Slajdy/MetodaFDD.pdf

[8] Amber, S.: *Průzkum tvrdí: Agilní programování funguje*. Software Developer: časopis o vývoji software a vývojářích, říjen 2006, roč. 1, č. 1, s. 29-30.

[9] Buchalcevová, A.: *Metodiky vývoje a údržby informačních systémů.* 1.vyd. Praha: Grada Publishing, 2005. 163 s. ISBN 80-247-1075-7

[10] Amber, S.: *I vy můžete pracovat agilně!*. Software Developer: časopis o vývoji software a vývojářích, únor 2007, roč. 2, č. 1, s. 12-13.

[11] Appleton, B.: *FDD is Football-Driven Development* , Dostupné z www: http://www.agilemanagement.net/Articles/Papers/FDDisFootball.html

[12] Palmer, S. Felsing, J.: *A Practical Guide to Feature-Driven Development(Kapitola 3)*, Dostupné z www: http://www.petercoad.com/download/bookpdfs/pg2fddch03.pdf

[13] MacDonald, M. Szpuszta M.: *ASP.NET 2.0 a C# tvorba profesionálních webových stánek* 1. vyd. Brno: Zoner software, 2006. 1376 s. ISBN 80-86815-38-2

Příloha 1

Uživatelská příručka

Přihlášení do systému

Na úvodní stránce informačního systému jsou umístěny základní informace o této aplikaci a také přihlašovací formulář. Pro přihlášení do systému je nutné zadat login a heslo uživatele. Poté proběhne přihlášení a uživatel je přesměrován na stránky systému na než má oprávnění.

Správa rolí

Pro správu rolí uživatelů musí mít uživatel práva systémového administrátora. Na této stránce viz. Obr.1 jsou role, které jsou dostupné v systému a má možnost zobrazení jejich detailu, jejich editace a případně i mazání. Správce má možnost také vytvořit novou roli. Pro vytvoření nové role je nutné zadat jméno nové role a identifikátor, který slouží k jednoznačné identifikaci role v systému. Tento identifikátor by měl být psán velkými písmeny neoddělenými mezerou o maximálním počtu deseti znaků. Při tvorbě nové role jsou také definovány práva, která bude mít uživatel s touto rolí. Tyto role jsou globálně viditelné v systému.

Obr. 1: Správa rolí

Domény společnosti

Pod touto záložkou má systémový administrátor možnost definovat nové společnosti nebo editovat a mazat stávající. Systémový správce nepředstavuje supersprávce, který by měl přístup ke všem členům a datům společnosti, ale má právo definovat v rámci společnosti nové správce společnosti a editovat stávající. Daní doménoví správci pak zabezpečují veškerou správu dané společnosti.

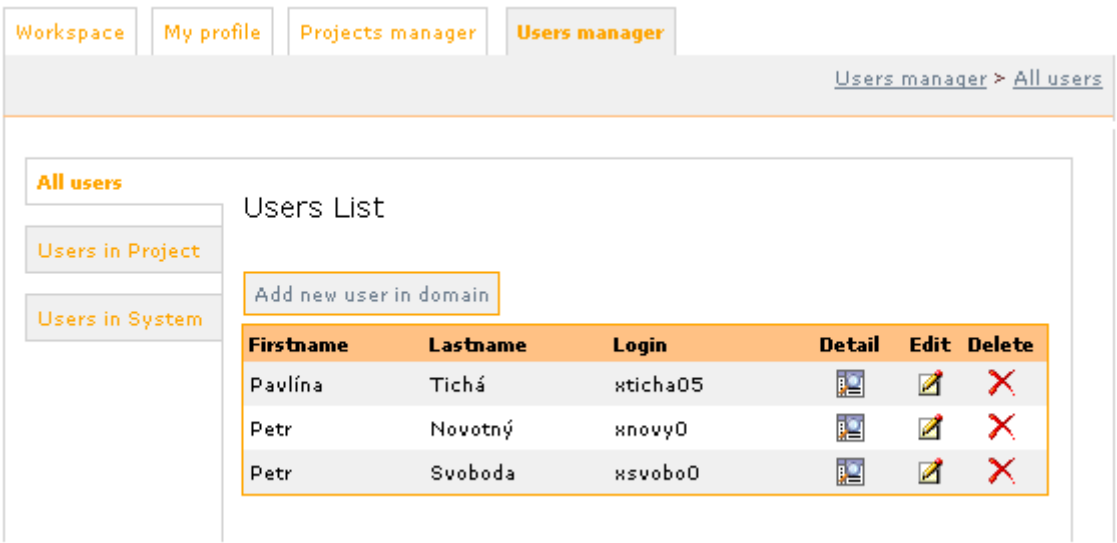
Profil uživatele

Každý uživatel má přístupnou záložku se svými osobními údaji. Je zobrazeno jakých rolí nabývá ve společnosti a na jakých projektech aktuálně spolupracuje. Má také možnost měnit heslo pro přihlášení do systému.

Správa uživatelů

Tato záložka je zobrazena všem uživatelům systému, kteří na ni mají práva. V rámci této stránky jsou dostupné tři stupně správy.

- Správa systémových administrátorů – zde jsou soustředěni všichni systémoví administrátoři
- Správa členů společnost – tato položka se zobrazí tomu uživateli, který je přihlášen jako člen nějaké společnosti a má práva správy členu společnosti. V rámci této sekce má potom možnost definovat nové členy společnosti a definovat jim práva ve společnosti. Tyto práva se vztahují globálně ke společnosti a nedědí se do projektu, na kterých daní uživatelé spolupracují.
- Správa projektových týmů – zde je nejprve nutné vybrat projekt a pak v rámci projektu mají uživatelé s právy projektového manažera možnost tvořit a editovat tým pro řešení projektu. Práv definovaných v této záložce nabývají uživatelé pouze v rámci projektu



Obr. 2: Správa uživatelů

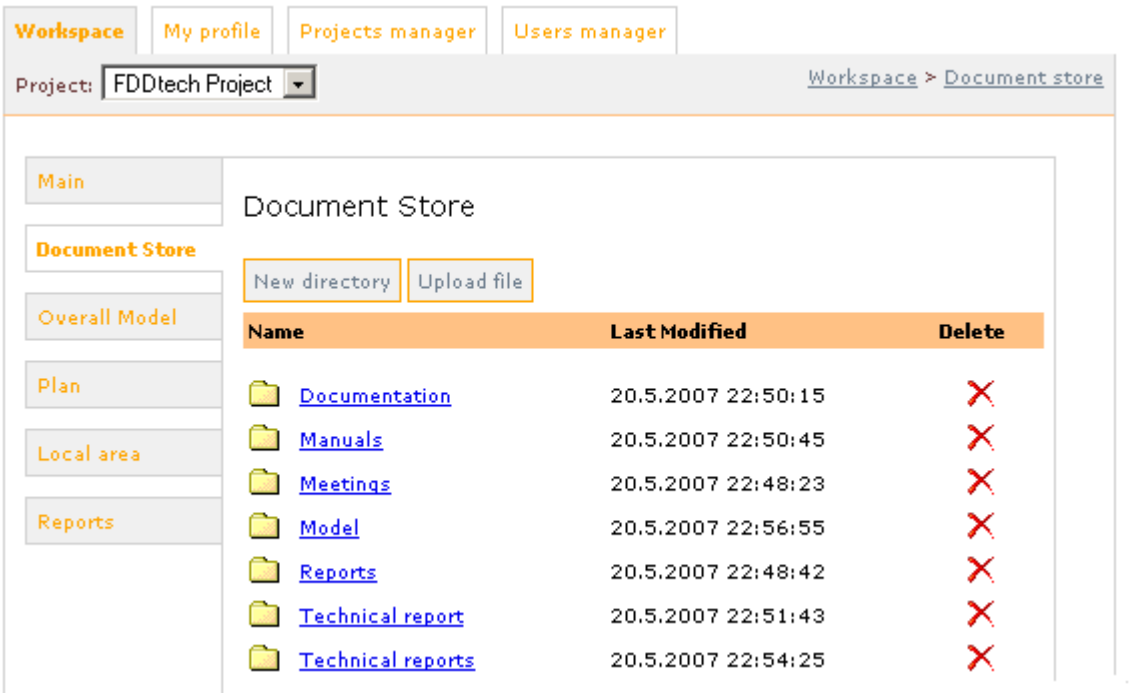
Správa projektů

Do této záložky mají přístup pouze uživatelé, kteří mají práva projektového manažera. Mohou zde založit nový projekt a definovat v jakém časovém období má být realizován. U každého projektu pak mají možnost editovat jeho údaje a také spravovat členy projektového týmu.

Pro každý projekt je možné definovat nové členy, kteří dosud nejsou členy společnosti, nebo přiřadit k projektu již existující členy společnosti. Pro každého člena takto vzniklého týmu může projektový manažer definovat jiné role jedinečné v rámci projektu.

Pracovní plocha projektu

Tato záložka se zobrazí pouze tomu uživateli, který je členem některého projektového týmu v rámci společnosti, ve které je přihlášen. První je nutné v záhlaví stránky zvolit, pod jakým projektem se bude pracovat. Po volbě projektu je již možné využívat prostředků dostupných pro podporu vývoje pomocí metodiky FDD.



Obr. 3: Náhled na pracovní plochu projektu

Informace o projektu

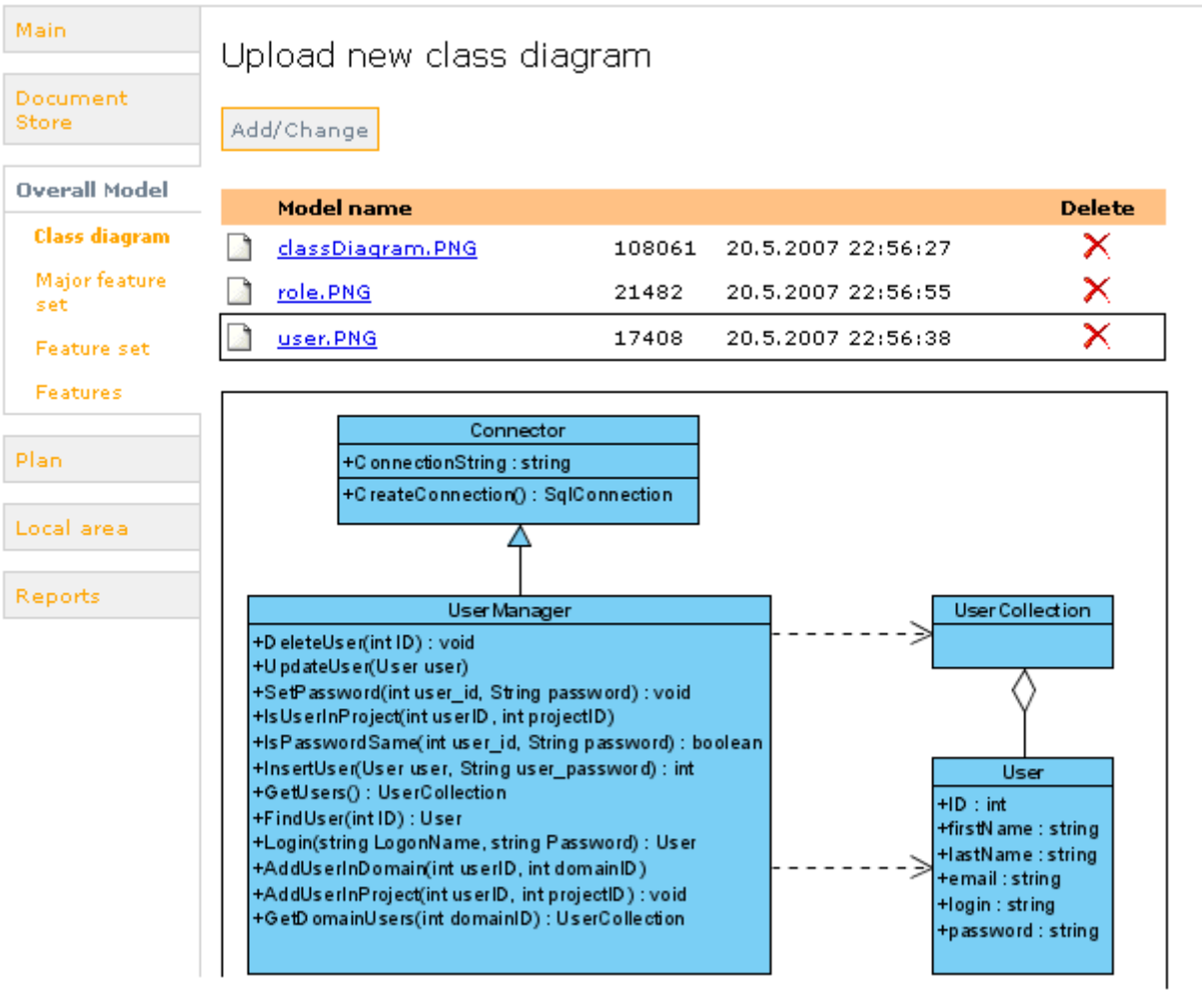
Zde jsou uvedeny pouze informační údaje týkající se projektu. Jeho název, plánované milníky a počet oblastí rysů, množin rysů a jednotlivých rysů. Je zde také stručný přehled v jakých pozicích jsou v týmu zastoupeni konkrétní lidé.

Dokumentový sklad

Na úrovni dokumentového skladu jsou definovány různá práva. Uživatel může podle přidělených práv sklad pouze prohlížet, vytvářet a mazat adresáře nebo nahrávat a mazat soubory. V dokumentovém skladu jsou soustředěny všechny dokumenty vztahující se k projektu. Uživatel může založit adresář s vysvětlujícím názvem např. Schůzky, Finální zprávy, Manuály ... a do něj pak nahrávat požadované soubory.

Celkový model

Zde lze podle prvního kroku metodiky FDD poskytnout vývojářům k nahlédnutí celkový model systému. Je umožněno nahrát více modelů a mezi nimi listovat. Po vybrání modelu je pak zobrazen vždy jeho náhled.



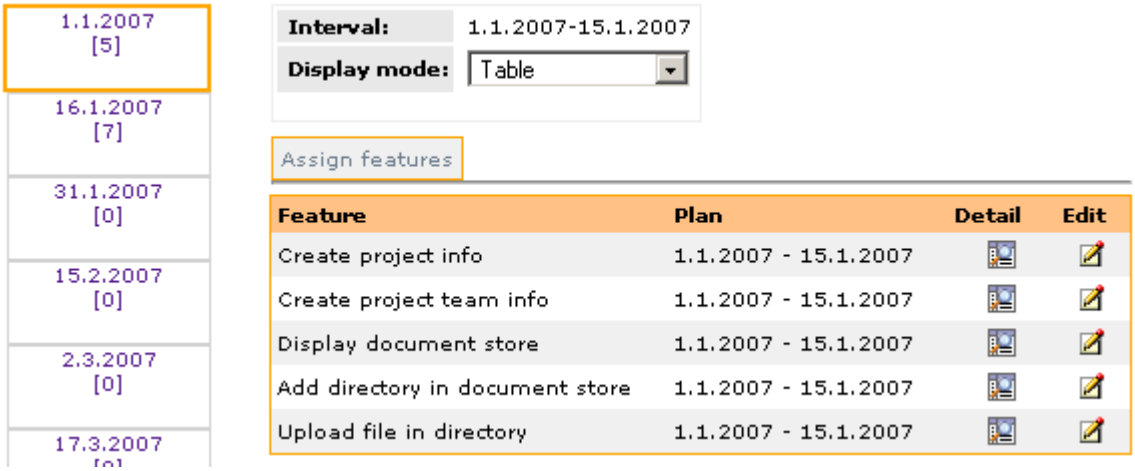
Obr. 4: Celkový model – diagram tříd

Seznam rysů

Seznam rysů je v metodice FDD strukturován do tří úrovní. V záložce celkového modelu je pak možno tvořit všechny úrovně. Na úrovni oblastí rysů je možno vytvořit danou oblast a pak v rámci ní vytvořit množinu rysů a v dalším zanoření jednotlivé rysy. U každého je možné specifikovat vlastníka, který ponese odpovědnost. Je zde umožněna editace a také mazání.

Plán

Pokud nebyl projektu přiřazen už při jeho založení začátek a konec je nutné v prvním kroku definovat právě tyto milníky. Po správném zadání začátku a konce projektu se zobrazí plánovaní projektu s časovou osou rozdělenou na dvoutýdenní intervaly. Data konců těchto intervalů pak určují datum provedení revize na projektu a kontrolu plnění plánu.



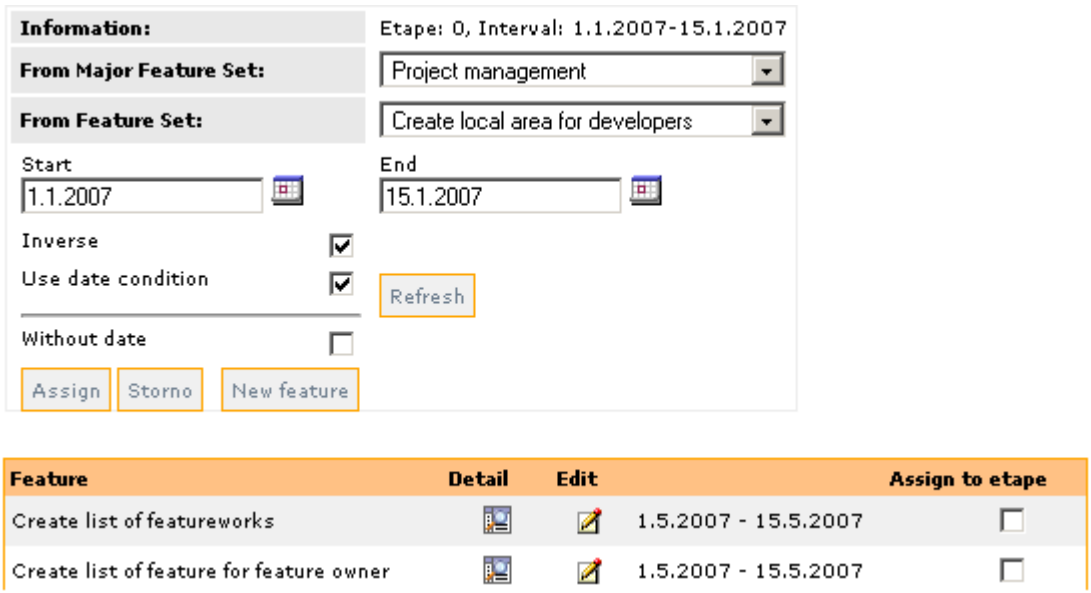
Obr. 5: Náhled na plán projektu

Obr. 5 ukazuje časovou osu projektu. Vždy je vybrán právě jeden interval a pro něj je možné zvolit rysy, které se mají v tomto časovém období realizovat. Pro plánování je nutné zvolit položku Přiřadit rysy. Následně je zobrazena nová stránka viz Obr. 6. pro pomoc s plánováním. V záhlaví je uvedena informace o jakou etapu se jedná a pak pomocné filtry pro nalezení potřebných rysů. Je možné vyhledat rysy na základě:

- Příslušnosti k oblasti rysu
- Příslušnosti k množině rysů
- Časové příslušnosti
- Bez zadaných milníků

Po nastavení vlastností filtrování rysů je nutné potvrdit filtrování a poté se zobrazí vyhovující rysy. U těch je barevně odlišena jejich příslušnost. Tmavě zelená barva určuje, že jsou zcela obsaženy v dané etapě, světle zelená to, že do etapy zasahují, ale vyskytují se i v rámci etapy jiné. Pokud rys není nijak zvýrazněn, tak není členem této etapy vůbec. U každého rysu je zobrazeno zaškrtávací políčko, které umožňuje přiřadit rys k dané etapě. Po zatrhnutí tohoto tlačítka a zmáčknutí tlačítka

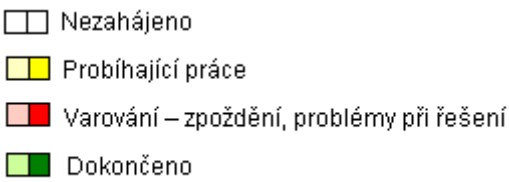
Přiřadit je rys přiřazen k etapě a jeho plánovaný začátek a konec je nastaven na začátek a konec etapy. Takto je možné naplánovat přibližnou realizaci rysů. V rámci etapy má pak možnost každý vlastník rysů vybírat, který rys se bude kdy realizovat.



Obr. 6: Stránka pro přiřazení rysů k intervalům

Seznam dílčích prací

Každý vývojář má v lokální oblasti přístup k přehlednému seznamu svých úkolů. Má zde zobrazeno, jaké práce má v rámci rysu vykonat. Na úvodní straně je zobrazen výčet všech dílčích prací vývojáře ve všech rysech daného projektu. Jednotlivé práce jsou barevně odlišeny a význam jednotlivých barev je uveden na obr. 7. Pokud je zvolena v seznamu určitá práce, pak je zobrazen její detail pod tabulkou s výčtem. U každé práce vývojář nastavuje z kolika procent už je práce hotova, kdy na ní začal pracovat a kdy s práci dokončil. U každé práce je také odkaz na rysový tým.



Obr. 7: Barvy odpovídající stavům

Vedení v rysovém týmu

Pod tuto záložku mají přístup pouze vlastníci určitých rysů, kteří představují vrchní programátory v metodice FDD. Je zde zobrazen seznam rysů, za které nesou odpovědnost. Ukáždého rysu je zobrazeno z kolika procent je hotový a kdy je naplánovaný jeho začátek a konec.

Feature name	Progress	Complete	Plan	Works	People
Create project info		100%	1.5.2007 - 15.5.2007	1	1
Create project team info		100%	1.5.2007 - 15.5.2007	1	1
Display documert store		100%	1.5.2007 - 15.5.2007	1	1
Add directory in document store		0%	10.5.2007 - 15.5.2007	0	0
Upoad file in directory		40%	1.5.2007 - 15.5.2007	1	1
Deete directory from document store		50%	1.5.2007 - 22.5.2007	1	1
Deete file from document store		60%	16.5.2007 - 30.5.2007	1	1

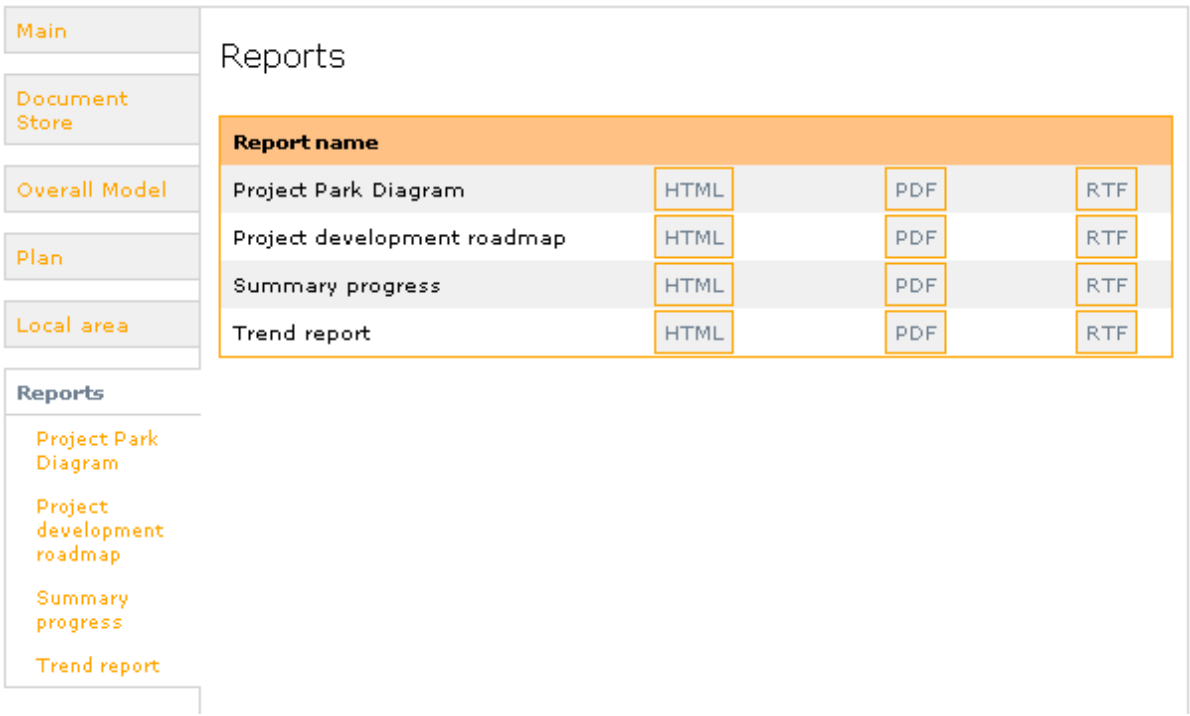
Obr. 8: Seznam rysů jednoho vlastníka

Pokud uživatel jeden rys vybere, tak se zobrazí ve spodní části stránky detailní informace o tomto rysu. Vlastník může daný rys editovat, plánovat jeho dílčí práce, zobrazit jeho tým a jaký je pokrok na rysu.

- **Editace rysu** – V rámci této položky je možné změnit údaje o rysu. Především se jedná o jeho pevné naplánování a aktualizace skutečného začátku a konce.
- **Plánování dílčích prací** – Zde má vlastník rysu právo nastavit, co potřebuje pro splnění daného rysu udělat a od jakého vlastníka rysu. Na základě celkového modelu vybere, jakou třídu nebo její část potřebuje implementovat, určí, kdo je vlastníkem této třídy a zadá, v jakém časovém intervalu má být práce realizována. Poté je tato práce přiřazena konkrétnímu vývojáři a zobrazí se v grafickém znázornění prací.
- **Rysový tým** – Zde jsou jednotliví členové rysového týmu. Je zde odkaz na jejich profily a na detailní seznam jejich dílčích prací na rysu.
- **Pokrok rysu** – Je zde zobrazena osa a na ní procentuální plnění daného rysu. Pod touto informací je výčet všech prací na rysu a jejich procentuální plnění. Vlastník rysu zde získává informace o pokroku na rysu a může sledovat, zda nedochází ke zpoždění.

Výkazy pokroku projektu (reporty)

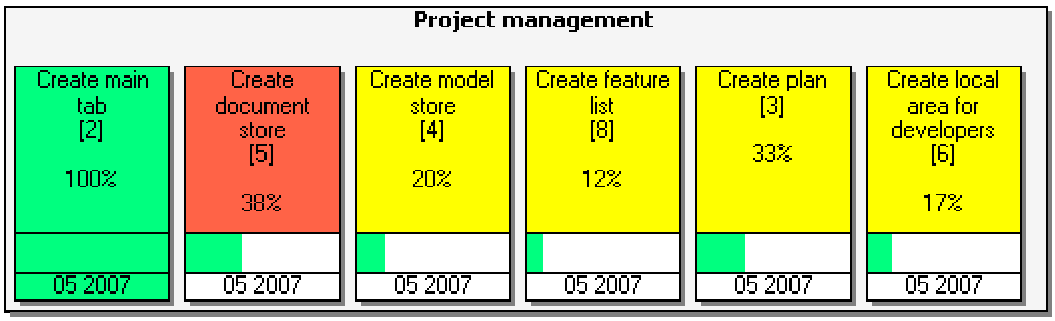
Na této úrovni jsou zobrazovány pro management projektu a koncového zákazníka zprávy o pokroku na projektu. Uživatel může zvolit v jakém formátu chce vybraný výkaz stáhnout nebo zobrazit. Podporované formáty jsou HTML, PDF a RTF.



Obr. 9: Zobrazení stránky se všemi reporty ve všech formátech

1. Park diagram

Tento výkaz je názorným prezentováním stavu a pokroku realizace jednotlivých oblastí rysů a jim podřízených množin rysů. Pro každou oblast rysu je zobrazen samostatný box s jejím jménem. V tomto boxu jsou pak uvedeny všechny množiny rysů, které obsahuje. Podrobný popis je uveden na obr. 10.



Obr. 10.: Park diagram

2. Graf pokroku projektu

Výkaz, který obsahuje graf znázorňující plánované a skutečné procento dokončených rysů v časovém intervalu vývoje projektu. Červená křivka vykresluje, kolik bylo na určitý datum naplánováno rysů a zelená křivka, kolik rysů bylo hotovo ve skutečnosti.

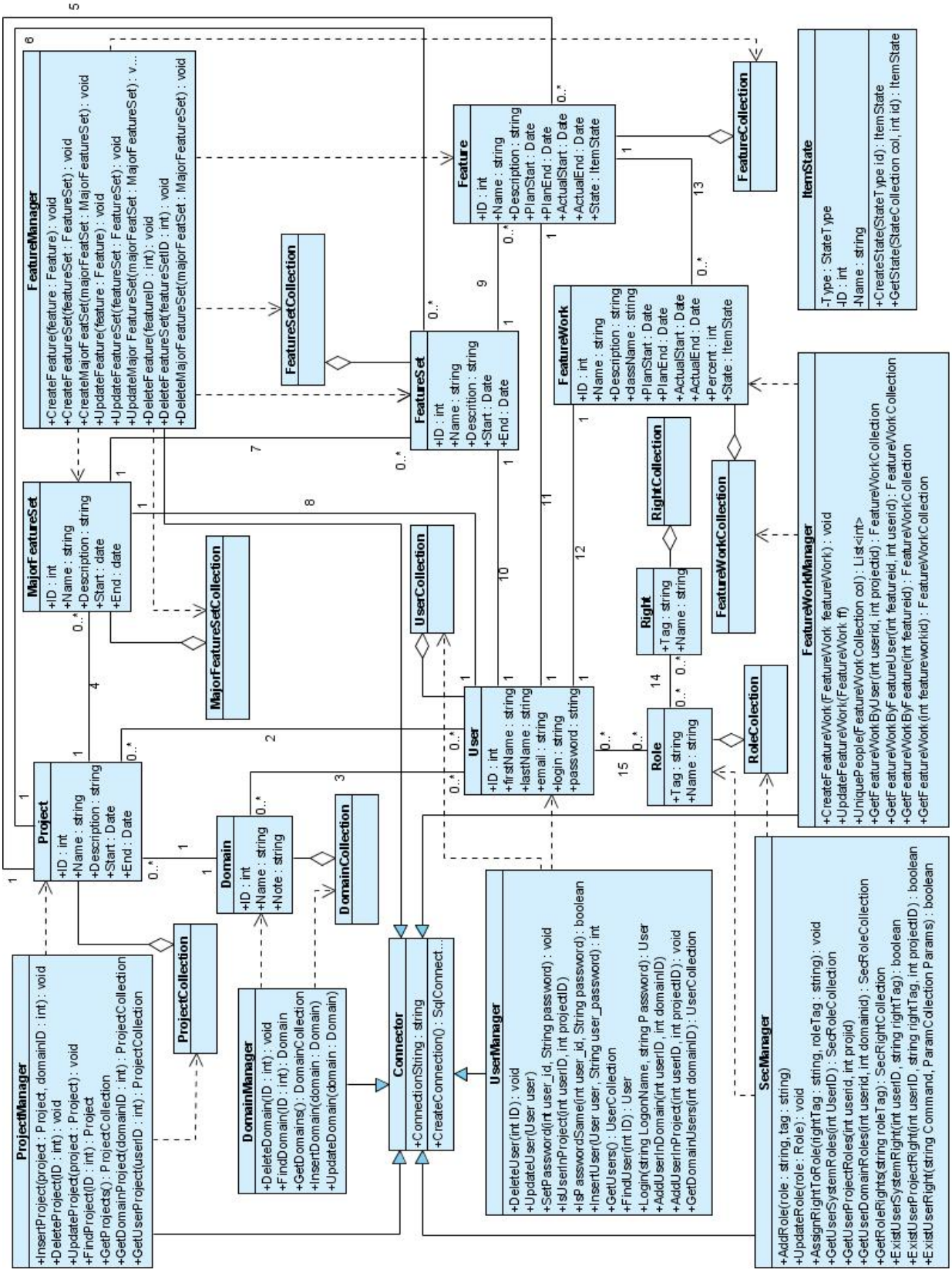
3. Report celkového pokroku

Tento výkaz poskytuje přehledné zobrazení, kolik rysů se v rámci každé množiny rysů vyskytuje v jakém stavu. Každá tabulka představuje jednu oblast rysů a je i nadepsána jejím jménem. Jednotlivé řádky tabulky jsou podřízené množiny rysů této oblasti. Ve sloupcích už je pak uvedeno kolik rysů se vyskytuje v daném stavu odpovídajícímu sloupci tabulky.

4. Report vývojového trendu

Tento výkaz vychází z celkového plánu projektu a váže se k plánovaným revizím. Pro datum každé revize je uvedeno, kolik je hotových rysů, na kolika se pracuje, kolik rysů ještě zbývá a případně kolik rysů má nějaké zpoždění.

Příloha 2


ProjectManager
+InsertProject(project : Project, domainID : int) : void
+DeleteProject(ID : int) : void
+UpdateProject(project : Project) : void
+FindProject(ID : int) : Project
+GetProjects() : ProjectCollection
+GetDomainProject(domainID : int) : ProjectCollection
+GetUserProject(userID : int) : ProjectCollection
Project
+ID : int
+Name : string
+Description : string
+Start : Date
+End : Date
ProjectCollection
DomainManager
+DeleteDomain(ID : int) : void
+FindDomain(ID : int) : Domain
+GetDomains() : DomainCollection
+InsertDomain(domain : Domain)
+UpdateDomain(domain : Domain)
Domain
+ID : int
+Name : string
+Note : string
DomainCollection
Connector
+ConnectionString : string
+CreateConnection() : SqlConnect...
UserManager
+DeleteUser(int ID) : void
+UpdateUser(User user)
+SetPassword(int user_id, String password) : void
+IsUserInProject(int userID, int projectID)
+IsPasswordSame(int user_id, String password) : boolean
+InsertUser(User user, String user_password) : int
+GetUsers() : UserCollection
+FindUser(int ID) : User
+Login(string LogonName, string Password) : User
+AddUserInDomain(int userID, int domainID)
+AddUserInProject(int userID, int projectID) : void
+GetDomainUsers(int domainID) : UserCollection
User
+ID : int
+firstName : string
+lastName : string
+email : string
+login : string
+password : string
UserCollection
Role
+Tag : string
+Name : string
RoleCollection
Right
+Tag : string
+Name : string
RightCollection
FeatureWork
+ID : int
+Name : string
+Description : string
+dassName : string
+PlanStart : Date
+PlanEnd : Date
+ActualStart : Date
+ActualEnd : Date
+Percent : int
+State : ItemState
FeatureWorkCollection
FeatureWorkManager
+CreateFeatureWork(FeatureWork featureWork) : void
+UpdateFeatureWork(FeatureWork ff)
+UniquePeople(FeatureWorkCollection col) : List<int>
+GetFeatureWorkByUser(int userid, int projectid) : FeatureWorkCollection
+GetFeatureWorkByFeatureUser(int featureid, int userid) : FeatureWorkCollection
+GetFeatureWorkByFeature(int featureid) : FeatureWorkCollection
+GetFeatureWork(int featureworkid) : FeatureWorkCollection
SecManager
+AddRole(role : string, tag : string)
+UpdateRole(role : Role) : void
+AssignRightToRole(rightTag : string, roleTag : string) : void
+GetUserSystemRoles(int UserID) : SecRoleCollection
+GetUserProjectRoles(int userid, int projid)
+GetUserDomainRoles(int userid, int domainid) : SecRoleCollection
+GetRoleRights(string roleTag) : SecRightCollection
+ExistUserSystemRight(int userID, string rightTag) : boolean
+ExistUserProjectRight(int userID, string rightTag, int projectID) : boolean
+ExistUserRight(string Command, ParamCollection Params) : boolean
FeatureManager
+CreateFeature(feature : Feature) : void
+CreateFeatureSet(featureSet : FeatureSet) : void
+CreateMajorFeatSet(majorFeatSet : MajorFeatureSet) : void
+UpdateFeature(feature : Feature) : void
+UpdateFeatureSet(featureSet : FeatureSet) : void
+UpdateMajor FeatureSet(majorFeatSet : MajorFeatureSet) : v...
+DeleteFeature(featureID : int) : void
+DeleteFeatureSet(featureSetID : int) : void
+DeleteMajorFeatureSet(majorFeatSet : MajorFeatureSet) : void
Feature
+ID : int
+Name : string
+Description : string
+PlanStart : Date
+PlanEnd : Date
+ActualStart : Date
+ActualEnd : Date
+State : ItemState
FeatureCollection
FeatureSet
+ID : int
+Name : string
+Description : string
+Start : Date
+End : Date
FeatureSetCollection
MajorFeatureSet
+ID : int
+Name : string
+Description : string
+Start : date
+End : date
MajorFeatureSetCollection
ItemState
-Type : StateType
-ID : int
-Name : string
+CreateState(StateType id) : ItemState
+GetState(StateCollection col, int id) : ItemState

Příloha 3

Popis nasazení systému

Za účelem jednoduché a nenáročné nasazení systému v praxi byly vytvořeny dva instalační balíčky. Tvorbu těchto instalačních balíčků zabezpečuje pro vyvíjené projekty Microsoft Visual Studio 2005. Jeden balíček je vytvořen pro databázovou část a druhý pro část webovou.

Pro správnou funkčnost databázové části je nutné nainstalovat MS SQL Server nebo jeho bezplatně šiřitelnou verzi SQL Server Express. Po spuštění instalačního balíčku databáze je uživatel vyzván k zadání názvu a cesty SQL serveru a názvu databáze. Po zadání těchto údajů instalační proces provede vytvoření databáze a importuje do ní databázové schéma systému a všechny výchozí data.

Instalátor pro webovou část si vyžádá přítomnost IIS, .NET 2.0 a Windows Installer 3.1. Následně je uživatel vyzván k zadání názvu webu a portu, na kterém má být spuštěna webová aplikace.

Posledním krokem je nakonfigurovat správně *connectionString* v aplikačním souboru web.config. Do atributu *connectionString* je nutné zadat platnou konexi na vytvořenou databázi.

Příloha 4

Obsah CD

Přiložený disk CD-ROM obsahuje následující adresáře:

- doc - technická zpráva diplomové práce
- bin - obsahuje datový instalační balíček a webový instalační balíček
- src – zdrojové kódy
- lib - podpůrné knihovny jako itextsharp a zedgraph
- util – spustitelný soubor pro web deployment
- db – databáze

Podrobnější informace jsou uvedeny v souboru *readme.txt* v kořenovém adresáři tohoto CD.