UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2010 Bc. Petr Plavec

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Projekt návrhu a tvorby webového informačného systém pre správu a optimalizáciu finančných portfólií

Bc. Petr Plavec

Diplomová práce

2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr PLAVEC**
Osobní číslo:	**I08375**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Projekt návrhu a tvorby webového informačného systém pre správu a optimalizáciu finančných portfólií**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V teoretickej časti je potrebné popísať zásady projektovania informačných systémov pomocou metodiky vychádzajúcej z metodického frameworku RUP. Zo stavebných kameňov RUP je potrebné zostaviť a popísať konkrétnu metodiku, ktorá bude použitá pre tvorbu daného informačného systému pomocou zvolenej metodiky je potom potrebné realizovať vybrané prípady užitia. Realizácia bude zahŕňať modelovanie požiadaviek, analytické a návrhové modely a implementáciu. Pri realizácii bude využitý zvolený štandardný framework pre realizáciu webových informačných systémov. Ako prílohy diplomovej práce budú odovzdané rôzne artefaktv definované zvolenou metodikou

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1.Dokumentácia metodického frameworku RUP 2.Arlow Jim, Neustadt IIa UML 2 a unifikovaný proces vývoje aplikaci

Vedoucí diplomové práce: **Ing. Jaroslav Lach**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **30. října 2009**

Termín odevzdání diplomové práce: **21. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

doc. Ing. Antonín Kavička, Ph
vedoucí katedry

V Pardubicích dne 10. listopadu 2009

Prohlášení autora

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 18. 8. 2010

Bc. Petr Plavec

Poděkování

Děkuji panu Ing. Jaroslavu Lachovi za náměty a připomínky ke tvorbě této diplomové práce.

ANOTACE

Práce se v teoretické části zabývá popisem základních vlastností metodického frameworku RUP.

Cílem praktické části je vytvoření postupů a modelů, pomocí kterých se implementuje systém pro správu finančních portfolií.

KLÍČOVÁ SLOVA

RUP, UML, Microsoft Office Sharepoint Server, informační systém

TITLE

Project of Design and Creation of Web Information System for Management and Optimization of Financial Portfolios

ANNOTATION

In the theoretical part are described the basic features of the methodological framework RUP.

The result of the practical part is the creation methods and models, by which implements a system for managing financial portfolios.

KEYWORDS

RUP, UML, Microsoft Office Sharepoint Server, information system

Obsah

Seznam obrázků

1 Úvod

V současnosti funguje v oblasti informačních technologií mnoho firem, které se zabývají vývojem informačních systémů. Při realizaci složitějších systémů je nutné mít přesně stanovený postup, podle kterého se bude tvořit výsledný produkt. Jednou možností jaké nám dnes různé postupy vývoje softwaru nabízejí, je metodický framework RUP (Rational Unifed Process). Tato metodika je poskytována společností IBM a je založena na iterativním vývoji softwaru.

Cílem této diplomové práce bylo vytvořit systém na optimalizaci finančních portfolii s využitím metody RUP. Jelikož RUP je metodický framework bylo zapotřebí vybrat si z něho důležité principy pro tvorbu systému, který měl být založený na frameworku určeného pro realizaci webových informačních systémů. To by mělo usnadnit implementaci systému.

Ve druhé kapitole je popsaná metodika RUP. Je zde podrobněji rozebráno šest základních obecných praktik, podle nichž se metodika řídí. V další části jsou popsány jednotlivé fáze životního cyklu projektu. Poslední část se zabývá hlavními součástmi této metodiky, kterými jsou role, aktivita, artefakt a pracovní proces. Na závěr této kapitoly je metodika RUP celkové vyhodnocena.

V další kapitole jsem se snažil popsat obecně systém, který měl být vytvořen. V kapitole jsou hlavně stanoveny funkční a nefunkční požadavky na systém. Je zde popsáno, že jsme se snažili vytvořit systém, který by podporoval vzájemnou spolupráci všech členů týmů, na které dohlíží konkrétní manažer, který by měl přehled o tom, co který zaměstnanec udělal.

Ve čtvrté kapitole je proveden průzkum frameworků, které by byly vhodné k realizaci webového informačního systému. Po průzkumu vhodných frameworků byl na základě vlastností vybrán framework, který nejlépe splňoval naše požadavky. Nakonec byl vybrán MOSS (Microsoft Office SharePoint Server).

Pro potřeby projektu musela být vytvořena projektová dokumentace, která by byla jakýmsi vodítkem při tvorbě informačního sytému. Další kapitola se zaměřuje na vytvořené projektové dokumenty, jakými jsou vize, iterační plán, seznam rizik a vyhodnocení.

Šestá kapitola má za úkol popsat modely, které byly vytvořeny. Hlavně je kladen důraz na model případu užití a na jednotlivé případy užití, které jsou popsány v přehledných tabulkách podle metodiky RUP. V další části bylo nutné popsat strukturu všech vytvořených modelů, pomocí kterých se vytvářel koncový produkt. Při tvorbě systému bylo nutné vzít v úvahu i specifické vlastnosti MOSS a přizpůsobit mu jednotlivé modely.

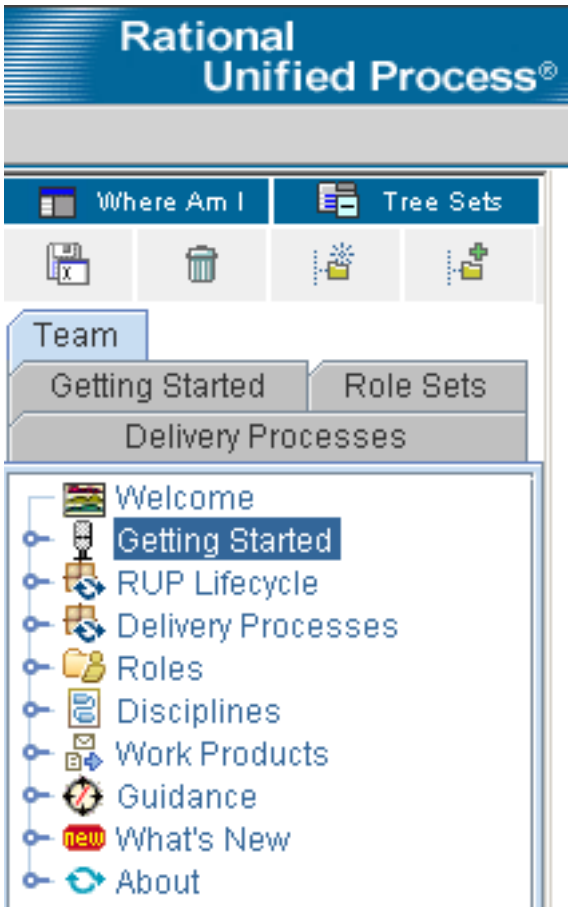
2 Metodika RUP

The Rational Unified Process (RUP) je postup projektování softwaru. Poskytuje disciplinovaný přístup k přiřazování úkolů a odpovědností v rámci organizace, která na projektu pracuje. RUP má za cíl zajistit výrobu vysoce kvalitního softwaru, který vyhovuje potřebám koncových uživatelů v rámci předvídatelného plánu a rozpočtu. Typickým znakem metodiky je její možnost konfigurace na konkrétní oblast použití. Proto se dá na metodiku pohlížet jako na procesní framework poskytující návod, jak vytvořit vhodnou kombinaci modelů a procesů vyhovující konkrétnímu projektu.

V rámci týmu vylepšuje produktivity tím, že každému členovi týmu poskytuje snadný přístup k základním znalostem, šablonám a radícím nástrojům pro všechny kritické vývojové činnosti. Tím, že všichni členové týmu mají přístup k základním znalostem, bez ohledu na to jestli pracují s požadavky, designem, testováním, řízením projektu nebo správou konfigurace, můžeme zajistit, že všichni členové týmu sdílejí společné informace, jak vyvinout kvalitní software.

RUP je vodítko pro to, jak efektivně používat Unified Modeling Language (UML). UML je standardní jazyk, který nám umožňuje jasně a přesně určit požadavky, architektury a vzory. UML byl původně vytvořen společností Rational Software. Nyní standard UML definuje standardizační skupina Object Management Group (OMG).

Pro automatizaci velké části procesů používá RUP různé nástroje, které jsou používány k vytváření a udržování různých artefaktů, zejména modelů, procesů softwarového inženýrství, mezi které patří vizuální modelování, programování, testování, atd. Možnosti a nástroje, které metodika obsahuje, jsou znázorněny na následujícím obrázku.



Obrázek 1: Hlavní nabídka v dokumentaci metodiky RUP (vlastní zdroj)

Metodika RUP je vhodná pro všechny možné vývoje softwarů. Unified Process je vhodný pro malé vývojové týmy, stejně tak jako pro velké organizace. Unified Process je založena na jednoduché a jasné procesní architektuře, která poskytuje zobecnění celé řady procesů. Metodika je variabilní a může být přizpůsobena různým situacím. Obsahuje různé nástroje, poskytující možnosti konfigurace tak, aby vyhovovala potřebám dané organizace.

Rational Unified Process uchovává několik nejlepších postupů při vývoji moderního softwaru ve formulářích, které jsou vhodné pro širokou škálu projektů a organizací. Zavádění těchto osvědčených postupů, pomocí Rational Unified Process, který slouží jako průvodce, nabízí řadu klíčových výhod pro vývojové týmy. V následující části bude popsáno šest základních nejlepších postupu v metodice RUP.

2.1 Efektivní vývoj pomocí šesti nejlepších praktik

Metodika RUP byla vytvořena na základě mnohaletých zkušeností vývojářů se SW projekty. V celé této metodice se promítají prověřené postupy a praktiky, které vedou

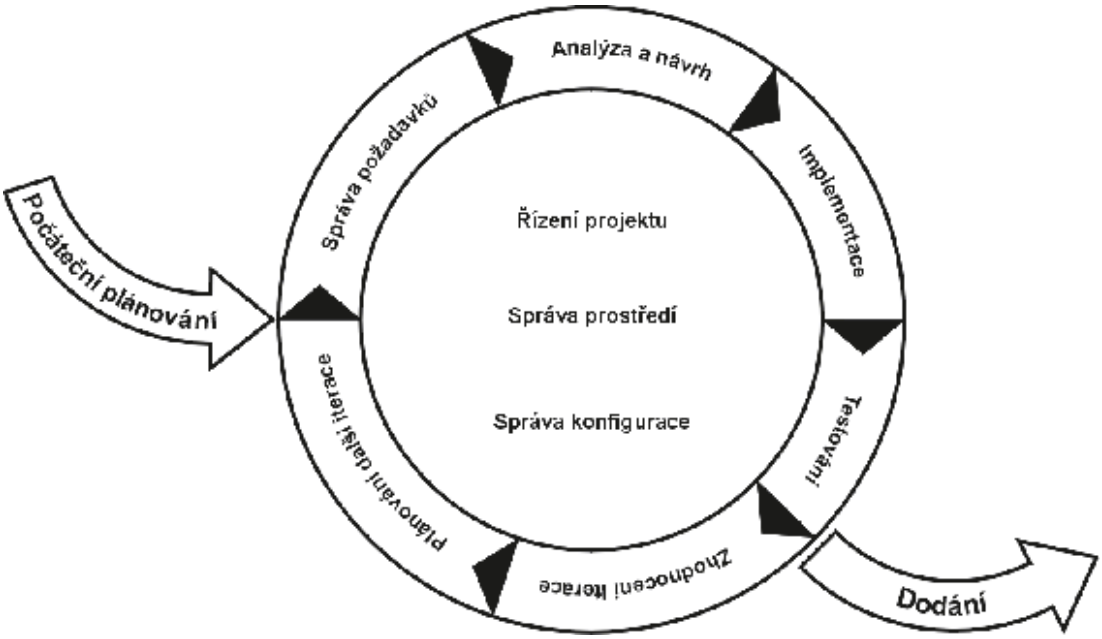
k úspěšnému vývoji systémů, zároveň zvyšují produktivitu práce, zlepšují komunikaci jak uvnitř vývojového týmu, tak mezi zadavatelem, a celkově zefektivňují celý vývojový proces. Metodika RUP se soustředí na vývoj software pomocí šesti základních obecných praktik. Tyto praktiky jsou následující:

- iterativní vývoj SW,
- správa požadavků,
- použití komponentové architektury,
- vizuální modelování SW,
- kontrolování kvality softwaru,
- řízení změn.

2.1.1 Iterativní vývoj SW

Vzhledem k dnešním sofistikovaným softwarovým systémům, je tedy nemožné, aby se postupně definoval celý problém, udělal návrh celého řešení, vybudoval software a potom otestoval výrobek na konci vývoje. Je zapotřebí iterativní přístup, který umožňuje zvýšení porozumění problému prostřednictvím po sobě následujících kroků, po kterých postupně vzniká efektivní řešení přes více iterací. Iterativní vývoj lépe odhaluje vysoká rizika v každé fázi životního cyklu, a tak významně snižuje jejich dopady.

Rizika projektu jsou identifikována zejména v počátečních fázích životního cyklu projektu, takže je možné je napadnout a reagovat na ně včas a efektivně. Rizika jsou tedy rozpoznána a ošetřena dříve, než mohou zastavit celý projekt. Vyvinuté verze jsou předány uživatelům, kteří poskytují zpětnou vazbu, čímž lze dolaďovat požadavky na systém. Vzhledem k tomu, že každá iterace končí spustitelným produktem, vývojový tým vidí podle výsledků a času stav projektu a díky kontrole je zajištěno, že projekt běží podle předem stanoveného plánu. Iterativní přístup také usnadňuje pojmout taktické změny požadavků, funkcí nebo plánu.



Obrázek 2: Schéma iterativního vývoje (Zdroj:http://objekty.vse.cz)

2.1.2 Řízení požadavků

Další praktikou využívanou RUP je řízení požadavků, která popisuje jak získávat, organizovat a dokumentovat požadované funkce a omezení; vyhodnocení změn požadavků a posouzení jejich dopadu na systém; sledování a dokumentace kompromisů a rozhodnutí.

Správa a řízení požadavků v projektu umožňují následující: komunikace probíhá formálně definovaným způsobem; požadavkům je možno určit prioritu, třídit je a sledovat; funkcionalitu a výkonnost je možné objektivně hodnotit; nekonzistence jsou jednodušeji objeveny; pomocí vhodných nástrojů je možné přehledně uchovávat všechny systémové požadavky, které se objevily v historii celého projektu, a jsou zároveň provázány s příslušnými externími dokumenty.

2.1.3 Použití komponentové architektury

Při implementaci softwaru je vhodné zvolit komponentovou architekturu, která zvyšuje produktivitu a vhodně se doplňuje s postupy iterativního vývoje. V některých iteracích tak lze podle potřeby vlastními prostředky vyvíjet nové komponenty, zajišťující unikátní funkčnosti vyvíjeného softwaru, jindy je možné uplatnit výhodné vlastnosti již realizovaných komponent.

Proces se na počátku vývoje a základních vývojových úrovní zaměřuje na robustní proveditelnou architekturu, před nasazením zdrojů pro plný vývoj. Popisuje, jak navrhnout pružnou architekturu, která je flexibilní, přívětivá ke změnám, intuitivně pochopitelná a podporuje opětovné použití softwaru.

Rational Unified Process podporuje vývoj založený na komponentovém vývoji softwaru. Komponenty jsou jednoduché moduly, subsystémy, které splňují jasně definované funkce. RUP poskytuje systematický přístup k definování vlastní architektury pomocí nových a existujících komponent. Ty jsou shromážděny do přesně definované architektury, buď jen pro tento případ, nebo do komponentové infrastruktury, jako je například Internet, CORBA a COM, které jsou nově vznikajícím odvětvím opakovaně použitelných komponent.

Komponentová architektura a iterativní postup má za následek neustálou potřebu rozvoje architektury systému. Každá iterace vyprodukuje nějakou architekturu, která může být testovaná nebo porovnávaná s požadavky na systém. Tento postup umožňuje neustálé předcházení možných rizik při vzniku systému.

2.1.4 Vizuální modelování SW

Model je zjednodušený pohled na sytém. Modely se vytváří, aby nám usnadnily porozumění systému, který chceme vytvářet.

Tento proces ukazuje, jak vizuálním modelem softwaru zachytit strukturu a chování architektury a komponent. To nám umožní psát kód pomocí „grafických stavebních bloků“. Vizuální abstrakce pomáhá přenášet různé aspekty našeho softwaru. Například snadněji je vidět, jak prvky systému do sebe zapadají. Zajišťuje, že „stavební kameny“ jsou v souladu s kódem. Zachovává soulad mezi návrhem a jejím prováděním, a podporuje jednoznačné sdělení. Industrystandard Unified Modeling Language (UML), který vytvořil Rational Software, je základem pro úspěšné vizuální modelování. Mezi nejčastěji využívané modely UML patří modely případů užití, analytické modely, diagramy tříd, sekvenční diagramy, komponentové diagramy, atd.

2.1.5 Kontrolování kvality softwaru

Špatný výkon aplikací a nedostatečná spolehlivost jsou společné faktory, které dramaticky brzdí přijetí dnešních softwarových aplikací. Proto by měla být kvalitou přezkoumána, aby spolehlivě splňovala požadavky na funkčnost, výkon aplikace a

výkon systému. RUP nám usnadňuje činnost při plánování, navrhování, provádění realizace a vyhodnocení jednotlivých zkušebních prototypů. Hodnocení kvality je postaveno do procesu, ve všech činnostech, se zapojením všech účastníků, na základě objektivních měření a kritérií, a není prováděn samostatnou skupinou.

Kontrola kvality se dá posuzovat podle požadavků FURPS, které byly vyvinuty společností Hawlett-Packard. Tato metoda zkoumá tyto důležité faktory:

- funkčnost,
- použitelnost,
- spolehlivost,
- výkon,
- podpora.

V RUP se můžeme setkat s označením FURPS+. Toto označení nám říká, že měření kvality je doplněno o další subfaktory:

- požadavky na design,
- požadavky na implementaci,
- požadavky na rozhraní,
- fyzické požadavky.

2.1.6 Řízení změn

Při práci v týmu je důležitá vzájemná komunikace. Týmy ovšem spolupracují na stejných iteracích. Proto musí mezi týmy probíhat optimální řízení, jinak by se vývoj systému mohl stát chaotickým.

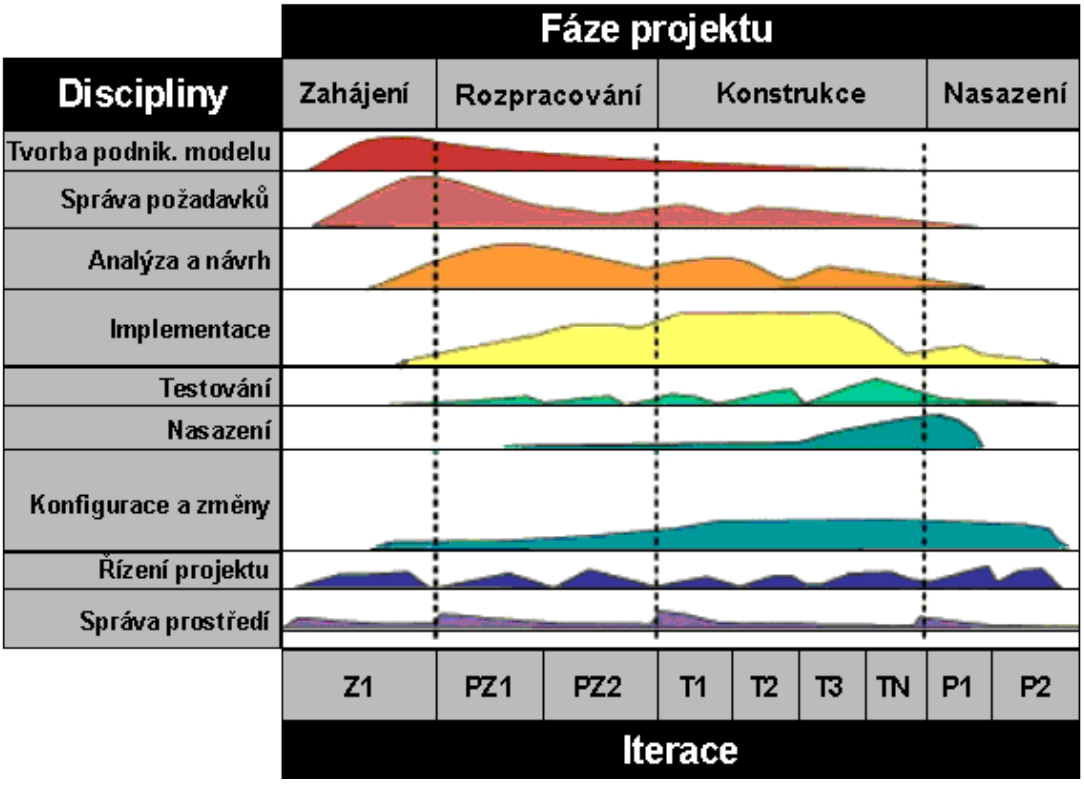
Je proto nezbytné sledovat všechny změny a vždy se ujistit, že každá změna je přijatelná, jako např. zda proběhlo otestování této změny nebo zda nekoliduje s dříve vytvořenými změnami. Tento proces tedy hlavně popisuje jak ovládat, sledovat a monitorovat změny k tomu, aby úspěšně splňovali iterativní vývoj. Také nás vede k tomu, jak vytvořit bezpečné pracovní prostředí pro každého developera tím, že je oddělený od změn provedených v ostatních pracovních skupinách. Tento proces také slouží k řízení změn všech softwarových artefaktů (např. modely, kód, dokumenty, atd.). Přínosem pro tým je pak, že může společně pracovat jako jeden celek.

2.2 Životní cyklus v RUP

V metodice RUP je projekt rozložen do čtyř základních fází. Mezi tyto čtyři fáze patří ty následující:

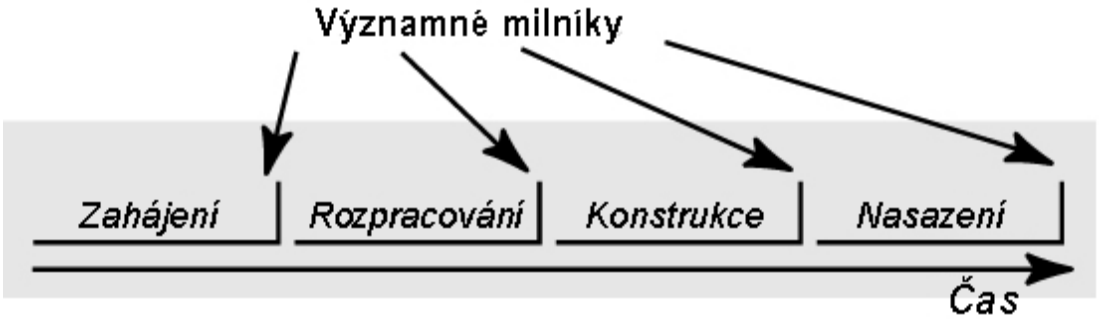
- Zahájení (Inception)
- Rozpracování (Elaboration)
- Konstrukce (Construction)
- Nasazení (Transition)

Na následujícím obrázku jsou znázorněny fáze, iterace a pracovní postupy metodiky RUP. Na pomyslné horizontální ploše je zobrazeno, jak se postupuje na projektu v jednotce času. Na vertikální ose jsou potom popsány pracovní činnosti. Pro jednotlivé pracovní činnosti je potom graficky znázorněno, v jakých fázích projektu se na té konkrétní pracovní činnosti intenzivně pracuje. Například na pracovní činnosti testování je vidět, že probíhá na konci každé iterace. Při zahájení projektu je úsilí menší než na posledních iteracích konstrukce.



Obrázek 3:Znázornění fází projektu, iterací a pracovních činností v metodice RUP (Zdroj:http://objekty.vse.cz)

V dalších podkapitolách budou jednotlivé fáze popsány podrobněji. Každá fáze musí být uzavřena podle přesně definovaného bodu (milníku). V tomto okamžiku musí být některá důležitá rozhodnutí provedena, a proto musí být klíčových cílů dosaženo.



Obrázek 4: Fáze a významné milníky v procesu (Zdroj [1])

2.2.1 Fáze zahájení (Inception)

Během zahajovací fáze, se musí navázat obchodní spolupráce pro vytvářený systém a vymezit rozsah projektu. Na to abychom toho dosáhli, musíme určit všechny vnější subjekty, se kterými bude systém komunikovat (aktéry). To znamená určit všechny případy užití a popsání několika významných skutečností. Mezi tyto skutečnosti například patři obchodní případ zahrnující kritéria úspěchu, hodnocení rizik, odhad potřebných zdrojů a fáze plánování ukazující časová data významných milníků ve tvorbě projektu.

V této fázi je nejdůležitější komunikace se zadavatelem, který by měl být seznámen s dokumenty jako je seznam rizik, vize projektu a iterační plán. Pro vývojové týmy je tato fáze velmi náročná, jelikož sestavování těchto dokumentů souvisí od dosažených zkušeností managementu, který řídí projekt, a od zkoumání všech skutečností, které se v systému objevují.

Výstupy z fáze zahájení:

- vize dokumentu: obecný pohled na jádro požadavků na systém, klíčové funkce, hlavní omezení;
- počáteční model případů užití hotový (10 – 20 %);
- počáteční projektový slovník pojmů;
- počáteční obchodní záměr, který zahrnuje obchodní souvislosti, kritéria úspěchu a finanční odhad;

- počáteční posouzení rizik;
- plán projektu ukazující fáze a iterace;
- jeden nebo několik prototypů.

Na konci fáze zahájení je první hlavní milník. Pro zahájení fáze jsou následující hodnotící kritéria:

- Zadavatel souhlasí s rozsahem definicí systému, s náklady a harmonogramem.
- Požadavky obsaženy v modelu případů užití.
- Důvěryhodnost nákladů a harmonogramu, odhady, priority, rizika a proces vývoje.
- Hloubka a rozsáhlost všech architektonických prototypů, které byly vyvinuty.
- Skutečné výdaje proti plánovaným výdajům.

2.2.2 Fáze rozpracování (Elaboration)

Účelem fáze zpracování je analýza samotného problému, základ architektonického modelu, vypracování plánu projektu a eliminace největších rizik v projektu. K dosažení těchto cílu musí být podrobný náhled na systém. Architektonická rozhodnutí mají být provedeny s pochopením funkčnosti celého systému, zejména pak jeho působností a hlavní funkční a nefunkční požadavky.

Častým tvrzením je, že tato fáze je nejdůležitější ze všech čtyř fází. Na konci této fáze je postavená architektura považovaná za úplnou a projekt prochází nejdůležitějším rozhodnutím o tom, zda se pustit do konstrukce a do fáze předání. Pro většinu systému se předpokládá, že systém bude přenosný, snadný a rychlý. Zatímco proces musí vždy zohlednit možné změny, činnosti fáze rozpracování zajistí, že architektura, plány a požadavky jsou dostatečně stabilní a rizika projektu jdou dostatečně zmírnit, takže můžeme předvídat náklady a termín dokončení vývoje projektu.

Ve fázi rozpracování by měl být sestaven prototyp architektury v jedné nebo ve více iteracích v závislosti na rozsahu, velikosti, rizika a novoty projektu. V tomto prototypu by se mělo zaměřit na rizika, která představují hlavní technický problém projektu.

Výstupy z fáze rozpracování:

- Model případů užití (alespoň z 80 % hotový) – všechny případy užití a aktéři by měli být nalezeni a hlavní případy užití by měli být popsány.
- Vedlejší požadavky, nefunkční požadavky a některé požadavky, které nejsou spojené s konkrétními případy užití, by měly být podchyceny.
- Popsána softwarová architektura
- Proveditelný architektonický prototyp
- Přepracovaný seznam rizik a přepracovaná obchodní stránka
- Vývojový plán pro celý projekt se zevrubným plánem projektu zaznamenávající iterace a hodnotící kritéria pro každou iteraci
- Aktualizovaný vývojový proces
- Předběžná uživatelská příručka

Na konci fáze rozpracování je druhá důležitá fáze projektu ukončena. V tomto okamžiku by měli být podrobně prozkoumány cíle systému, oblast nasazení, volba architektury a sestavení hlavních rizik.

Hlavní hodnotící kritéria pro fázi rozpracování zahrnují odpovědi na tyto otázky:

- Je vize a architektura stabilní?
- Pomocí spustitelného kódu, který je vytvořen v iteracích, je ověřena architektura. Tento kód zahrnuje hlavní požadavky na systém. Jsou rizika identifikována a zmírněna?
- Je plán pro fázi konstrukce dostatečně podrobný a přesný? Je založen na reálných odhadech?
- Souhlasí všechny zúčastněné strany s tím, že současné vize může být dosaženo, současný plán je proveditelný pro vývoj kompletního systému v současné navržené architektuře?
- Jsou aktuální vynaložené zdroje přijatelné vůči plánovaným výdajům?

Pokud jsou výsledky této fáze kladně vyhodnoceny, může se přestoupit do další fáze a tou je fáze konstrukce.

2.2.3 Fáze konstrukce (Construction)

Ve fázi konstrukce jsou všechny komponenty a funkce integrovány do produktu a všechny funkce jsou důkladně testovány. Fáze konstrukce je v jistém smyslu výrobní

proces, kde je kladen důraz na řízení zdrojů, na kontrolu operací optimalizující náklady, na harmonogram a na kvalitu.

Mnohé projekty jsou na tolik složité, že musí být proveden paralelní vývoj. Tyto paralelní aktivity mohou významně urychlit nasazení produktu, mohou ovšem také zvýšit složitost na řízení zdrojů a na synchronizaci pracovních postupů. Architektura bývá někdy velmi robustní, proto je důležité, aby byl plán srozumitelný. Jinými slovy, jedna z nejdůležitějších vlastností architektury je její jednoduchost při vývoji. To je jeden z důvodů, proč je kladen důraz ve fázi rozpracování na rozvážný vývoj architektury a podrobný plán. Výsledkem fáze konstrukce je produkt, který je připravený na předání do rukou koncových uživatelů.

Výsledek této iterace obsahuje minimálně:

- softwarový produkt integrovaný na odpovídající platformy,
- uživatelské příručky,
- popis aktuální verze.

Na konci fáze konstrukce je dokončený třetí důležitý milník projektu. V tomto okamžiku se musí rozhodnout, zda software, weby a uživatelé jsou připraveni na provoz, aniž by tento provoz odhalil vysoká rizika projektu. Tato verze je často nazývána jako beta verze.

Hlavní hodnotící kritéria pro fázi konstrukce zahrnují odpovědi na tyto otázky:

- Je tento produkt určený k uvolnění dostatečně připravený?
- Jsou všechny zúčastněné strany připraveny na nasazení produktu ke komunitě uživatelů?
- Jsou skutečně vynaložené zdroje oproti plánovaným výdajům ještě přijatelné?

Nikdy nemůže dojít k přechodu na fázi nasazení v případě, že bylo dosaženo neúspěchu v tomto milníku.

2.2.4 Fáze nasazení (Transition)

Účelem fáze nasazení je přechod produktu ke koncovým uživatelům. Jakmile se systém dostane ke koncovému uživateli, vznikají obvykle problémy, které vyžadují vznik nových verzí, opravy některých problémů nebo dokončení funkcí, které byly během tvorby projektu odloženy.

S předáním projektu je spojeno několik činností. Uživatelé systému provádí beta testy sloužící k validaci produktu. Někdy může nastat, že se provádí přechod z aktuálního softwaru s případnou konverzí používané databáze. Ve fázi nasazení musí taktéž dojít k zaškolení uživatelů a správců nového softwaru.

Fáze nasazení zahrnuje:

- Dosažení toho, že uživatelé budou se systémem samostatně pracovat.
- Dosažení shody všech zúčastněných stran, že aktuální verze systému odpovídá kritériím ve vizi.
- Dosažení finálního produktu rychle, ekonomicky efektivně a tak, aby byl praktický.

Tato fáze může rozsahem být velmi jednoduchá až velmi komplexní, v závislosti na typu produktu. Například, nasazení nové verze desktopové aplikace může být velmi jednoduché, oproti tomu nasazení systému, který hlídá letový vzdušný prostor, může být velmi složité.

Na konci fáze nasazení je dokončena čtvrtá část etapy projektu. Tento milník se nazývá Release Milestone (bod vydání). V tomto bodě se může rozhodnout, zda cíle byly splněny a jestli by se měl zahájit další cyklus. V některých případech se může tento milník shodovat s koncem zahajovací fáze pro další cyklus.

Hodnotící kritéria pro fázi nasazení zahrnuje odpovědi na tyto otázky:

- Je uživatel spokojený?
- Jsou skutečně vynaložené zdroje oproti plánovaným výdajům ještě přijatelné?

2.2.5 Iterace

Každá fáze v Rational Unified Process se může dále dělit na iterace. Iterace je kompletní vývojová smyčka vedoucí k uvolnění spustitelného produktu (interně nebo externě), což může být nějaká podmnožina konečného produktu ve vývoji, který roste od iterace k iteraci, aby se stal uceleným systémem.

Ve srovnání s klasickým vodopádovým procesem, iterační postup má následující výhody:

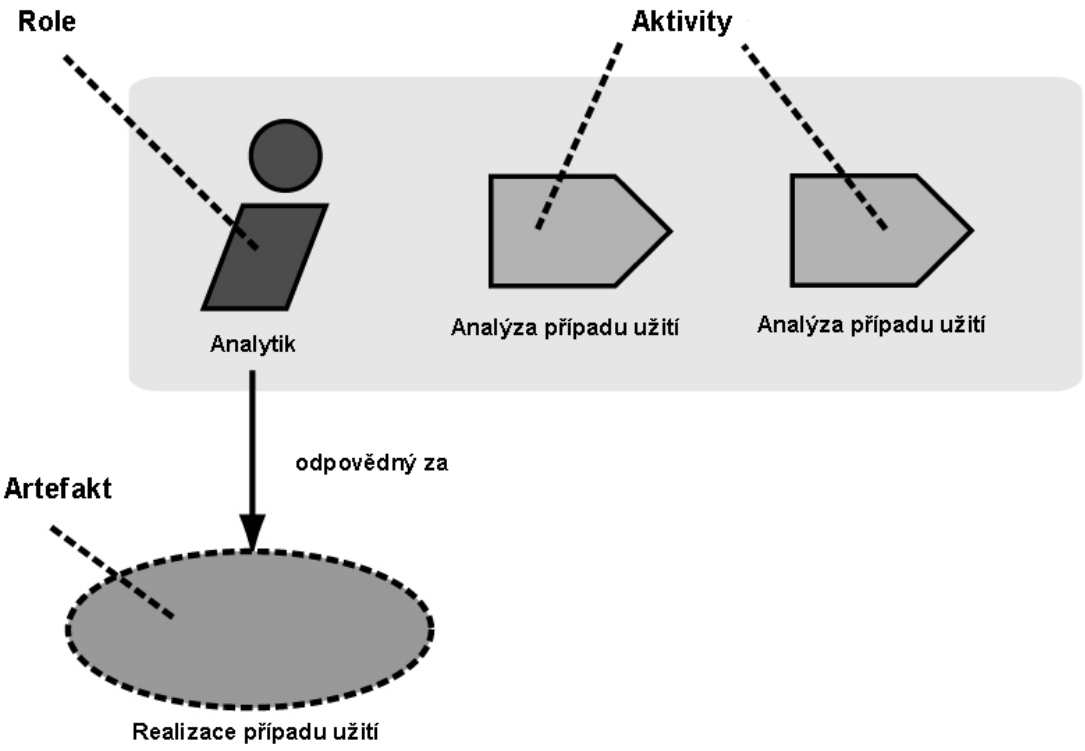
- rizika jsou zmírněny mnohem dříve;
- každá změna je lépe zvladatelná;

- vyšší míra opětovného použití;
- projektový tým se může učit během vývoje;
- celkově vyšší kvalita.

2.3 Součásti metodiky

Metodika RUP popisuje *kdo*, *co*, *jak* a *kdy* dělá. Kdo v procesu vývoje vystupuje, co má vytvořit, jak to má vytvořit a kdy to má vytvořit. Tato příslovce se vztahují ke čtyřem elementům:

- role – kdo,
- aktivita – jak,
- artefakt – co,
- pracovní proces – kdy.



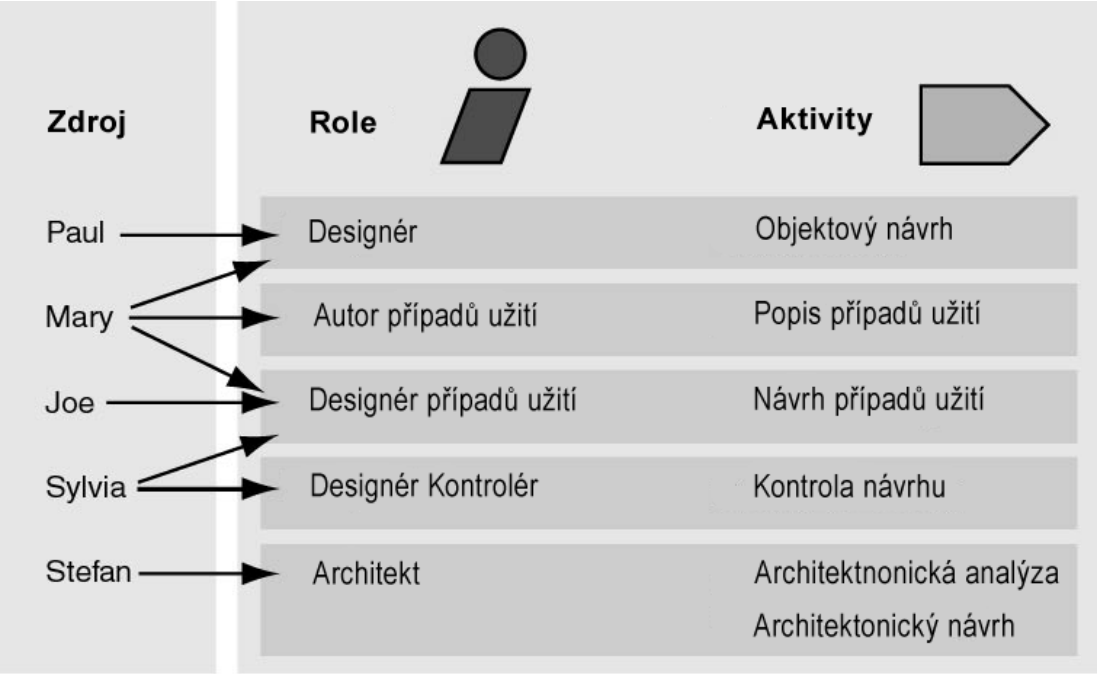
Obrázek 5: Role, aktivity a artefakt (Zdroj [1])

2.3.1 Role

Role se v metodice zavádějí, aby definovaly odpovědnosti jedince, či skupiny osob, které pracovaly na projektu jako tým. Role se přiřazuje v rámci projektu jednotlivci. Jedna osoba však může mít různé role. Pokud má osoba více rolí, je nutné definovat

odpovědnost pro každou jeho roli. Role tedy není konkrétní osoba. Role popisuje jen chování a odpovědnost.

Do rolí, které jsou zapojeny do vývojového procesu, mohou být obsazovány i osoby, které nepracují přímo ve vývojové organizaci. To jsou osoby, které přicházejí z vnějšku, jako je například zadavatel projektu nebo uživatel projektu. Osoby se obsazují do rolí podle jejich znalostí technik a metod, kterých se při vývoji projektu využívá při vykonávání různých aktivit.



Obrázek 6: Přiřazení rolí jednotlivým osobám a jejich aktivity (Zdroj [1])

2.3.2 Aktivita

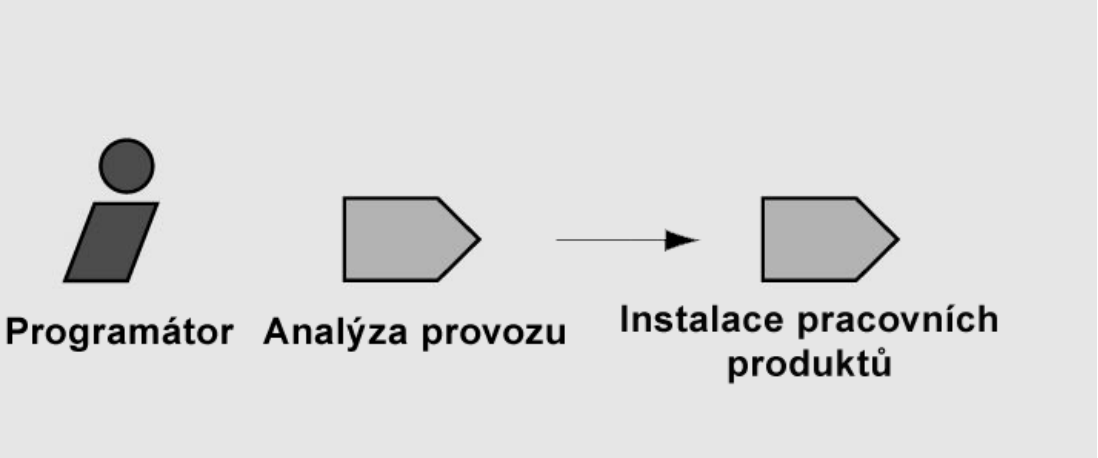
Aktivitou by se dalo rozumět, nějaká činnost nebo úkol vykonávaný nějakou osobou obsazené do definované role. Výstupy z této činnosti se nazývají artefakty. Aktivitu lze považovat za jednotku práce, která může být přidělena dané osobě.

Každá aktivita má jasný účel, obvykle vyjádřený ve smyslu vytváření nebo aktualizace některého artefaktu, jako je například model, třídy nebo iterační plán. Aktivita by měla být použita jako součást plánování a vývoje, pokud je tato činnost příliš malá, může být i zanedbána, ale je-li příliš velká, musí se rozdělit na různé podčásti, aby se dosáhlo zjednodušení této činnosti.

Příkladem aktivity může být plánování iterací. Role, která by měla vykonávat tuto aktivitu, se nazývá projektový manažer. Vstupem jsou dokumenty relevantní pro plánování, výstupem je dokument s plánem iterace.

Zde jsou uvedeny další příklady aktivit:

- hledání případů užití a aktérů (Role: Analytik);
- kontrola návrhů (Role: Designér Kontrolér);
- provedení testů výkonnosti (Role: Tester).



Obrázek 7: Grafické znázornění aktivity v pracovním procesu implementace (Zdroj [1])

2.3.3 Artefakt

Představuje ucelenou informaci nebo její část, která je vytvořena, modifikována a používána v procesu vývoje. Artefakty vznikají za celou dobu vývoje. Mají přidělenou svoji roli, která za každý vytvořený artefakt zodpovídá. Mají svůj definovaný životní cyklus, jeho délka je u různých artefaktů odlišná. Některé mohou být využity pouze jednou a některé mohou být využity při každé iteraci, kde mohou být modifikovány. U modifikace artefaktu se musí tento artefakt označit jinou verzí. Bez některých artefaktů by se vývoj projektu nedal vůbec zvládnout. Mezi takovéto artefakty se může považovat například vize, iterační plán nebo model případů užití.

Artefakty jsou produktem nebo meziproduktem vývoje SW, jsou to vstupy a výstupy aktivit. Nemusí to vždy být nějaký dokument popisující nějakou činnost. Mezi artefakty se považují i různé modely nebo části zdrojového kódu.

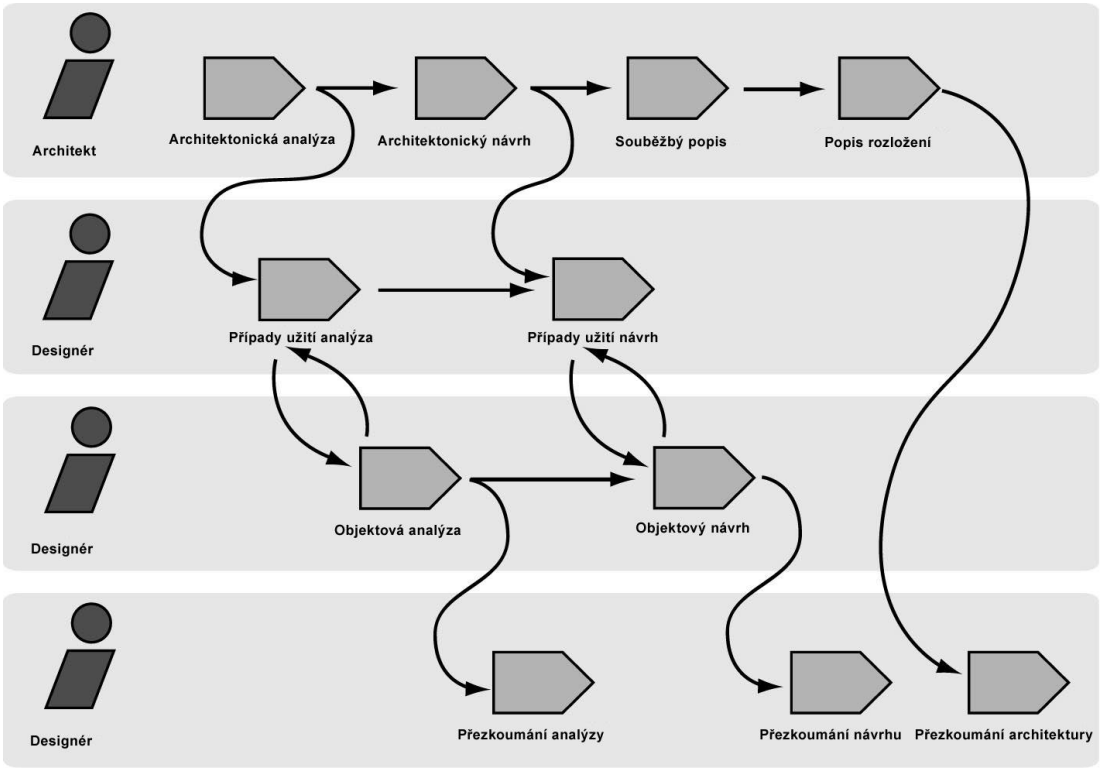
Obrázek 8: Příklad artefaktů vytvořených projektovým manažerem (vlastní zdroj)

2.3.4 Pracovní proces

Pouhé vyjmenování všech rolí, aktivit a artefaktů nevytváří proces. Potřebujeme způsob, jakým popíšeme smysluplnou posloupnost aktivit, které mají výstup a ukázat interakce mezi rolemi.

Pracovní proces je posloupnost aktivit vykonávaných rolemi, které produkují viditelný výstup. Každý pracovní proces se tedy skládá z několika aktivit. Z hlediska UML může být pracovní proces vyjádřen sekvenčním diagramem, diagramem vzájemné spolupráce nebo diagramem aktivit.

Je třeba si uvědomit, že ne vždy je možné nebo praktické zachytit v modelech všechny závislosti mezi aktivitami. Často jsou dvě aktivity těsně propojeny jinak, než jak je zachyceno na modelu. Lidé nejsou stroje a pracovní postup nelze vykládat doslovně jako program pro lidi, který má být přesně dodržován a prováděn mechanicky.



Obrázek 9: Ukázka pracovního procesu (Zdroj [1])

V metodice RUP je devět hlavních pracovních procesů, které seskupují všechny role a aktivity do logických skupin.

Hlavní pracovní procesy jsou rozděleny do šesti základních „inženýrských“ pracovních procesů:

- obchodní modelování (Business modeling workflow),
- specifikace požadavků (Requirements workflow),
- analýza a návrh (Analysis and Design workflow),
- implementace (Implementation workflow),
- testování (Test workflow),
- nasazení (Deployment).

Dále jsou pracovní procesy rozděleny na tři „podpůrné“:

- konfigurační a změnové řízení (Configuration and Change Management workflow),
- projektové řízení (Project Management workflow),
- prostředí (Environment workflow).

I když názvy šesti „inženýrských“ pracovních procesů mohou připomínat sekvenční fáze v tradičním vodopádovém životním cyklu, měli bychom mít na paměti, že jednotlivé fáze v iteracích jsou rozdílné a pracovní procesy jsou v průběhu životního cyklu stále přehodnocovány. Skutečné kompletní projekty využívají v praxi těchto devět hlavních pracovních procesů, které jsou opakovaně používány s různou intenzitou a důrazem v každé iteraci.

Obchodní modelování

Jedním z hlavních problémů při vývoji SW je nedokonalá komunikace mezi obchodním inženýrstvím a softwarovým inženýrstvím. Může to nakonec vést k tomu, že výstupy z obchodního inženýrství nejsou správně používány a vyvinutý software bude pro zákazníka nepoužitelný. RUP se snaží poskytnout vhodné nástroje a procesy oběma stranám tak, aby si navzájem vyhověly. Hlavní snahou je, aby existovala souvislost mezi obchodními a softwarovými modely.

V obchodním modelování jsou obchodní procesy dokumentovány za použití takzvaných obchodních případů užití. To zajišťuje společné porozumění mezi všemi zúčastněnými stranami o tom, co z podnikových procesů musí být podporována v rámci organizace.

Specifikace požadavků

Cílem pracovního procesu specifikace požadavků je popsat, co by měl systém dělat a umožňuje vývojářům systému a zadavatelům se lépe dohodnout na popisu tohoto systému. K dosažení tohoto cíle je zapotřebí zjistit, utřídit a zdokumentovat přehled všech požadovaných funkcí a omezení. Je nutné sledovat a zaznamenávat kompromisy a všechny rozhodnutí, které se udělaly.

Vize je dokument, který poskytuje kompletní vizi o vyvíjeném systému a vytváří podporu pro kontrakty mezi financujícími orgány a vývojovou organizací. V tomto dokumentu jsou identifikovány případy užití, což představuje chování systému. Je třeba také zjistit aktéry představující uživatele nebo jiný systém, který s vyvíjeným systémem bude spolupracovat. Každý případ užití je detailně popsán. Tento popis definuje, jak systém spolupracuje s aktérem a co dělá.

Analýza a návrh

Cílem analýzy je ukázat jak bude systém realizován ve fázi konstrukce. Chceme tedy vytvořit systém, který by měl vykonávat úkoly a funkce specifikované modelem pří-

padů užití, měl by splňovat všechny předem definované požadavky a měl by být silně strukturovaný (snadno ho změnit v okamžiku, kdy se provedou změny jeho funkčních požadavků).

Výsledkem analýzy a návrhu je návrhový model popřípadě analytický model. Návrhový model slouží jako abstrakce zdrojového kódu, to znamená, že návrhový model je plán, jak bude zdrojový kód strukturován a napsán.

Návrhový model se skládá z návrhových tříd strukturovaných do návrhových balíčků a subsystémů, které mají dobře definované rozhraní. Dále obsahuje popisy toho, jak objekty návrhových tříd spolupracují.

Návrhové činnosti jsou změřeny na architekturu. Výroba a validace architektury je hlavní náplní počátečních návrhových iterací. Architektura je reprezentována několika pohledy. Tyto pohledy zachycují hlavní konstrukční rozhodnutí. V podstatě jsou architektonické pohledy abstrakce nebo zjednodušení celé konstrukce, ve které jsou důležité vlastnosti více zviditelněny tím, že se neklade takový důraz na detaily. Architektura je důležitým nástrojem nejen pro vývoj dobrých návrhových modelů, ale také pro zvýšení kvality veškerých modelů postavených během vývoje systému.

Implementace

Cílem implementace je definovat uspořádání kódu s ohledem na implementaci subsystémů poskládaných do vrstev. Dalším cílem je implementace tříd a objektů s ohledem na komponenty. Při implementaci se musí provádět test vyvinutých komponent a jednotek. V poslední části implementace je nutné, aby se výsledky jednotlivých vývojových týmů integrovali do výsledného spustitelného systému.

Vyvíjený systém se implementuje pomocí komponent. RUP popisuje, jak opětovně použít už existující komponenty nebo jak implementovat nové komponenty s dobře vymezenou odpovědností. To zaručuje, že se systém snadněji udržuje a zvyšuje to i jeho opětovné použití.

Testování

Účelem testování je ověřit verifikaci mezi objekty a správnost integrace všech komponent. Dalším účelem je ověření, zda všechny požadavky na systém byly provedeny správně. Posledním účelem tohoto procesu je zajistit chyby, které mohli v systému vzniknout, před nasazením.

RUP používá iterativní přístup, což znamená, že testování se provádí v průběhu projektu. Tento postup nám umožňuje najít potencionální chyby co nejdříve, takže nám to radikálně snižuje náklady na stanovení chyby. Testy jsou prováděny ve třech oblastech: spolehlivost, funkcionalita, aplikační a systémová výkonnost. Pro každé z těchto měřítek kvality proces popisuje, jak provádět v životním cyklu plánování, projektování, implementace, provedení a vyhodnocení.

Strategie pro to, kdy a jak zautomatizovat testy jsou v RUP popsány. Automatické testy jsou zvláště důležité při použití iterativního přístupu. Umožňují provádět regresní testování na konci každé iterace, stejně tak jako u každé nové verze produktu.

Nasazení

Cílem pracovního procesu nasazení je úspěšně vyrobený produkt uvolnit a dodat software ke koncovým uživatelům. Tento proces zahrnuje širokou škálu aktivit jako je balíčkování softwaru, distribuce, instalace a poskytování pomoci a asistence uživatelům. V mnoha případech tato činnost zahrnuje i plánování a provádění beta testů, migrace existujícího softwaru nebo dat a formální přijetí softwaru zákazníkem.

Přestože proces nasazení je většinou soustředěn do fáze nasazení (transition phase), je potřeba mnoho jeho aktivit zařadit do dřívějších fází, hlavně na konec fáze konstrukce (construction phase).

Projektové řízení

Projektové řízení je zodpovědné za dodržování stanovených cílů, řídí rizika a překonává omezení tak, aby výrobek úspěšně splňoval všechny potřeby zákazníků a uživatelů. Skutečnost, že velmi málo produktů je úspěšných ukazuje na to, že projektové řízení je velmi složitý proces. Cílem projektového řízení je zjednodušit úkoly. Metodika RUP poskytuje různé návody k projektovému řízení. Je třeba si uvědomit, že se nejedná o zaručený recept vedoucí k úspěchu, ale o návody, které zpravidla významně zvyšují šance na úspěšný vývoj a následné dodání systému.

Projektové řízení v metodice RUP zahrnuje: základní rámec pro řízení projektu, návody a rady pro plánování projektu, obsazování rolí, denní řízení a kontrolu projektu, rámec pro řízení a správu rizik. To vše není recept na úspěch, ale představuje to obecný přístup k řízení projektu, který výrazně zlepší pravděpodobnost úspěšného dodání softwaru.

Konfigurační a změnové řízení

Tento pracovní proces popisuje, jak řídit velké množství artefaktů, které jsou vyráběny mnoho lidmi pracujících na společném projektu. Kontrola pomáhá vyhnout se nákladným zmatkům a zajišťuje, aby výsledné artefakty nebyly ve vzájemném konfliktu kvůli některým z následujících typů problémů:

- Současný update – pokud dva nebo více lidí pracují odděleně na stejném artefaktu, poslední z nich může udělat změny, které přemažou práci ostatních lidí.
- Omezená sdělení – když je opravovaný problém v artefaktu rozdělen mezi více vývojářů, může nastat, že nebudou informováni o změně.
- Více verzí – většina velkých programů je vyvíjena ve vývojových verzích. Jednu verzi může používat zákazník, zatímco druhá verze je teprve testována a třetí je zatím ve vývoji. Pokud se vyskytnou problémy v jedné verzi, je potřeba provést opravy u všech ostatních verzí. Nebezpečí záměny může vést k nákladné opravě a přepracování, proto je potřeba změny důkladně sledovat a kontrolovat.

Konfigurační změnové řízení popisuje, jak můžeme pomocí pomocných nástrojů zachovávat stav o tom, proč, kdy a kým byl změněn jakýkoliv artefakt. Tento pracovní proces také zahrnuje požadavek na změnu řízení, to je jak vytvářet zprávy o chybách a jak chybná data využít pro další vývoj.

Prostředí

Účelem pracovního procesu prostředí je poskytnout vývojové prostředí, potřebné nástroje a procesy, které jsou nutné pro podporu rozvoje vývojového týmu. Zaměřuje své aktivity na konfiguraci procesů v rámci projektu. Také se snaží vytvořit pokyny pro podporu projektu. Krok za krokem popisuje jak implementovat procesy v organizaci.

Pracovní proces prostředí poskytuje mnohé vývojové nástroje, které obsahují pokyny, šablony a nástroje jak přizpůsobit procesy.

2.4 Vyhodnocení metodiky

Metodika RUP je založena na třech základních principech. Těmito principy jsou role, aktivita a artefakt. Díky tomu máme lepší přehled o tom jaká osoba, které byla

přiřazena určitá role, vykonávala určitou činnost a za jaký výsledný artefakt zodpovídá. Taktéž je jasně vidět, ke které činnosti se vztahuje jaký artefakt.

Jelikož RUP je metodický framework, tak nám poskytuje návod, jak postupovat při tvorbě projektu tak, že si z něho vezmeme jen to, co potřebuje pro konkrétní projekt. Nástroje metodiky jsou dodávány jako webové stránky, takže mohou být součástí firemního intranetu. Díky tomu zajišťuje informace všem zaměstnancům v reálném čase. V těchto nástrojích můžeme najít velké množství šablon, které nám ulehčí vytvářet dokumenty, modely atd.

Metodika vznikla na základě mnohaletých zkušeností předních světových odborníků zabývajících se softwarovým inženýrstvím. Díky tomu vychází z potřeb používaných na reálných projektech a plně v sobě implementuje všech šest základních obecných praktik vývoje popsaných v kapitole 2.1

Navigace v metodice je velmi intuitivní, ale velmi obsáhlá. Každý proces odkazuje na mnoho dalších míst. Každá činnost či artefakt mají stanovené kroky, které je potřeba udělat před zahájením. Může se snadno stát, že se člověk v tak velkém množství potřebných kroků může jednoduše ztratit.

Metodika je velmi mocný nástroj v případě, že se vhodně používá a lidé, kteří s ní pracují, mají dobrý přehled, jak se v ní pohybovat. Pro mnoho firem se může stát dobrým vodítkem, jak produkovat kvalitní software.

3 Popis systému

Vzhledem k tomu, že trh je zaplněn množstvím nabídek a orientovat se v nich je pro běžného občana velice obtížné, musí vyhledávat služby finančních poradců. Finanční poradce musí analyzovat současné smlouvy, které klient uzavřel, a navrhnout klientovi výhodnější řešení zhodnocení jeho peněz. Optimalizovat stávající programy není zrovna jednoduché a proto systém, který bude navržen, by měl usnadnit práci finančním poradcům a manažerům dohlížejících na jejich činnost.

3.1 Finanční služby

Finanční služby jsou poskytovány v následujících odvětvích: hypoteční úvěry, úvěry a meziúvěry ze stavebního spoření, spotřebitelské půjčky, splátkový prodej, leasingy,

životní pojištění, penzijní připojištění, stavební spoření, investování na kapitálovém trhu a bankovnictví.

3.2 Optimalizace finančních programů

Optimalizací finančních programů je označován proces, při kterém se analyzují dostupné finanční možnosti klienta a porovnávají se s existujícími programy tak, aby nabídka pro klienta byla co nejvýhodnější. Po této analýze dochází k vytvoření návrhu změn dosavadních finančních programů tak, aby klientovy potřeby a cíle byly naplněny. Tato analýza je natolik informačně náročná, že vytvoření aplikace by se mnohonásobně ulehčilo práci finančního poradce.

3.3 Kolaborace v týmu

V rámci týmu, který poskytuje finanční poradenství, je potřeba sdílet mezi sebou informace o finančních portfoliích. Manažer týmu musí mít přehled nad svými podřízenými, u kterých kontroluje správnost vypracovaných finančních portfolií. Jednotné úložiště uzavřených smluv urychlí vyhledávání a orientaci již vytvořených finančních portfolií pro klienty.

3.4 Požadavky na aplikaci

- Funkční požadavky
    - Optimalizace stávajících finančních programů (vytvoření finančního portfolia klienta)
    - Vytvoření tiskového výstupu finančního portfolia s využitím statistických analýz a grafů
    - Zprostředkování finančních portfolií v rámci týmu
- Systémové požadavky
    - Webová aplikace přístupná z veřejného internetu
    - Diferencovaný přístup finančních poradců
    - Centrální uložení dat
- Obecné požadavky
    - Zamezení opakovaného pořizování stejných dat

4 Způsob implementace

Pro implementaci informačního systému jsme hledali vhodný framework, pomocí něhož by se systém snáze implementoval. Na základě znalostí a zkušeností byl základním požadavkem na framework programovací jazyk, jakým byl framework naprogramován, proto jsme vybírali z těchto programovacích jazyků PHP a ASP.NET.

4.1 Kohana

Tento framework je jistou verzí CodeIgniteru, napsanou čistě pro PHP 5. Dalším rozdílem je, že jej nevyvíjí firma, jak je tomu u CodeIgniteru, ale je vyvíjen komunitou.

Dokumentace je zpracována velmi přehledně a díky rozdělení na tematické celky v ní lehce najdeme požadované informace. Chybějící API reference se dá lehce vygenerovat pomocí vestavěného modulu Kodoc.

Díky své jednoduchosti a výborné dokumentaci je Kohana ideální pro začínající uživatele. Není to jen jednoduchý framework, ale rovněž může sloužit k vytváření složitých systámů. Po přidání několika komponent jej, díky jeho rychlosti a nízkým paměťovým nárokům můžeme využít i pro velké projekty.

Základní vlastnosti:

- Kohanu lze rozšířit přidáním knihoven či modulů. Každý modul může mít své knihovny, pohledy, controllery a tak dále.
- Je volitelně dodávaná se čtyřmi moduly, které doplňují Kohanu např. o jednoduchý generátor formulářů nebo o autentizaci uživatelů.

4.2 ZEND

Zend Framework (ZF) je asi nejrozšířenějším PHP frameworkem. Tento framework je od společnosti Zend Technologies Ltd.

Dokumentace je u Zendu velmi propracovaná a snadno se v ní orientuje. Je také doplněna velkým množstvím příkladů. Na stránkách Zendu nalezneme i různé video tutoriály a prezentace.

Tento framework je celosvětově velmi rozšířený a komunita kolem něj je obrovská. Na internetu tak existují velká fóra v různých jazycích, jež jsou naplněna řešenými problémy.

Ačkoli je Zend velmi populární a nabízí mnoho zajímavých možností, rychlosti přístupu do databáze nejsou příliš rychlé.

Základní vlastnosti:

- všechny komponenty jsou plně objektově orientované a vyhovují direktivě E_STRICT,
- modulární architektura typu „užij-co-potřebuješ“ minimalizuje křížové závislosti mezi komponentami,
- rozšiřitelná implementace MVC s podporou layoutů a šablonovacím systémem,
- podpora pro multi-databázové systémy zahrnuje MySQL, Oracle, IBM DB2, MSSQL Server, PostgreSQL, SQLite a Informix Dynamix Server,
- kompozice e-mailu a schopnost jej odeslat / přijmout skrze mbox, Maildir, POP3 nebo IMAP4.

4.3 DotNetNuke

DotNetNuke je open source verze systému pro správu obsahu (Content Management System) vhodný pro vytváření a nasazování webových projektů jako jsou například komerční webové stránky, intranety a extranety a online publikační systémy a portály. Je to také projekt, jehož cílem je vyvíjet software na základě zapojení komunity a vzájemného sdílení znalostí.

DotNetNuke je poskytován jako open-source software sířený pod licenčním ujednáním typu BSD. Obecně tato licence dává právo získat software bez jakýchkoliv poplatků. Kromě toho dává tato licence uživatelům právo nakládat s tímto software libovolně a to jak v případě komerčního tak nekomerčního využití, a to s jedinou podmínkou, aby uživatelé napomáhali rozvoji komunity kolem projektu DotNetNuke.

DotNetNuke je postaven na platformě Microsoft ASP.NET (VB.NET) a lze jej velmi snadno instalovat a spravovat. Díky rostoucí komunitě a dedikované skupině

vývojářů a profesionálů, kteří jsou základní vývojářskou skupinou DotNetNuke, je podpora pro DotNetNuke okamžitě dostupná.

DotNetNuke je systém pro správu obsahu. Ideální na vytváření a nasazování projektů jako komerčních webových stránek, firemní intranety a extranety a online publikační servery a portály.

Základní vlastnosti:

- Byl navržen tak, aby uživatelům zjednodušil správu jejich vlastních webových projektů.
- Podporuje více portálů a webových projektů v jedné instalaci.
- Je vybaven vestavěnými nástroji, které poskytují bohatou nabídku funkčností.
- Je podporovaný jeho hlavním vývojovým týmem a také mezinárodní komunitou uživatelů.
- Může být připraven k provozu během několika minut.
- Obsahuje lokalizační prvky umožňující administrátorům snadno překládat jejich projekty a portály do libovolného jazyka.

4.4 ASP.NET MVC

ASP.NET MVC je volně dostupný framework podporovaný firmou Microsoft pro vytváření webových aplikací, které využívají model-view-controller. ASP.NET MVC je založen na ASP.NET. Tento framework se po stažení nainstaluje přímo do Visual Studia, kde nám přibude možnost vytváření ASP.NET MVC projektů. Díky tomu je práce s tímto frameworkem velmi jednoduchá.

Základní vlastnosti:

- zajišťuje úplnou kontrolu nad vlastními HTML značkami,
- umožňuje bohaté AJAX integrace,
- intuitivní webové stránky,
- přehledné oddělení webových aplikací, což umožní jednodušší úpravy v budoucnosti
- jednoduchá testovatelnost

4.5 Microsoft Office SharePoint Server

Produkt Microsoft Office SharePoint Server poskytuje jednotné, integrované místo, kde zaměstnanci mohou efektivně spolupracovat s dalšími členy týmu, vyhledávat materiály organizace, odborníky nebo podnikové informace, spravovat obsah a pracovní postupy a využívat přehledu o podniku k přijímání kvalitnějších rozhodnutí. Microsoft SharePoint Server může být ve firmě nasazen různými způsoby, které jsou znázorněny na obrázku 10.

Obrázek 11: Možnosti nasazení Microsoft Sharepoint v rámci firmy. (Zdroj: www.microsoft.com)

Základní vlastnosti:

- Umožňují týmům účinně *spolupracovat*, podílet se na vytváření a publikování dokumentů.
- Je možné vytvářet *portály*. Osobní web pro sdílení informací a úpravu uživatelského rozhraní a obsahu podnikového webu na základě profilu uživatele.
- Rychlé a snadné *vyhledávání* osob, znalostí a obsahu v obchodních aplikacích.
- Umožňuje *vytvářet* a *spravovat* dokumenty, záznamy a webový obsah.

- Je možné vytvářet *pracovní postupy* a elektronické formuláře za účelem automatizace a zjednodušení podnikových procesů.
- Informační pracovníci mohou snadno přistupovat k nepostradatelným *obchodním informacím*, analyzovat a prohlížet data a publikovat sestavy, což jim umožňuje přijímat kvalitnější rozhodnutí.

4.6 Volba frameworku pro implementaci

Podle požadavků zadavatele, který kladl důraz na vzájemnou spolupráci zaměstnanců, které má ve svém týmu, jsme vybírali nejvhodnější framework. Po prozkoumání základních vlastností jednotlivých frameworků jsme dospěli k závěru, že nejvhodnější variantou pro implementaci našeho systému bude Microsoft SharePoint Server od společnosti Microsoft. Tvorba systému v ostatních prozkoumaných frameworcích by byla velmi programátorsky a časově náročná.

Využití Microsoft Office Sharepoint Server (MOSS) v našem případě spočívá ve spolupráci jednotlivých zaměstnanců a ve správě podnikového obsahu. Výběr toho systému by nám měl usnadnit propojení dat mezi zaměstnanci zvláště v případě, kdy manažer potřebuje dohlížet na smlouvy, které vytvořili zaměstnanci patřící do jeho týmu.

Nevýhodou tohoto systému je nutnost, že musí běžet na serverovém operačním systému od Microsoftu, jako je například Windows Server 2003 nebo Windows Server 2008. Pro implementaci informačního systému jsme tedy museli nainstalovat server.

4.7 Tvorba systému v SharePointu

Při vývoji našeho systému jsme se snažili co nejvíce využít vlastností MOSS tak, abychom nemuseli nic programovat. Při nutnosti doprogramovávat si komponenty pro MOSS by se nám zvýšili náklady na vlastní údržbu systému, které mohou vzniknout příchodem novějších verzí MOSS. Může se stát, že s příchodem novějších verzí nemusí naše vlastní doprogramované komponenty fungovat. Další skutečností, která by mohla nastat, je, že se ve frameworku může vyskytnout nějaká chyba. Za odstranění této chyby by byl potom zodpovědný dodavatel frameworku.

MOSS obsahuje několik základních možností vývoje aplikace. Mezi tyto možnosti patří standardní využití komponent a nástrojů již zabudovaných v MOSS. Pomocí těchto komponent a nástrojů můžeme vytvářet aplikace bez nutnosti psaní vlastních

kódů. Vývoj aplikací je pak založen na seznamech a relacích mezi nimi. Seznamy jsou podobné tabulkám, které jsou v prostředí SQL serverů. Každý seznam je množina sloupců, kde každý sloupec má přiřazený svůj datový typ, který je součástí (MOSS), ale MOSS si umožňuje nadefinovat vlastní datový typ. Mezi seznamy můžeme tvořit relace. Relaci vytvoříme tak, že jako datový typ jednoho ze sloupců seznamu vybereme odkaz na položku v jiném seznamu.

Při postupu této implementace jsme ovšem narazili na problém při výpisu finančních portfolií. Při výpisu jsme potřebovali více dat z jednoho listu, což MOSS standardním řešením neumožňoval. Proto jsme museli si tento výpis naprogramovat pomocí tzv. WebPart, což jsou komponenty, které se vkládají do aplikace.

5 Projektová dokumentace

Při postupu pomocí metodiky RUP se vede podrobná projektová dokumentace, která je jakýmsi vodítkem při tvorbě projektu. Dokumenty, které jsme během tvorby systému vytvářeli, jsou popsány v následujících kapitolách.

5.1 Vize

Tento dokument slouží jako podklad při realizaci samotného projektu. Popisuje samotný problém, který je realizován, charakteristiku uživatelů a zadavatelů a přehled produktu, kde jsou popsány požadované funkce systému.

5.1.1 Nastavení

Problém čeho	Při vytváření finančního programu je finanční poradce nucen složitě zpracovávat klientovi smlouvy tak, aby dosáhl optimálního řešení. Další problém je dohled nad podřízenými v rámci jednoho týmu.
Ovlivňuje	Finanční poradci zpracovávající finanční programy. Vedoucí týmů dohlížející nad prací své skupiny
Dopad problému	Nízký komfort při vytváření finančního portfolia. Špatný dohled nad členy týmu.
Výhody úspěšného vyřešení	Zvýšení komfortu práce finančního poradce. Zjednodušení kontroly vytvořených portfolií v rámci týmů.

Tabulka 1: Přehled problému

Pro	Finanční poradci a vedoucí týmů
Kdo	Nedostatek podobných systémů pro finanční poradce
Název produktu	Konzultat
Který	Zjednodušení práce finančního poradce při vytváření finančního plánu, který musí být přehledný a srozumitelný klientovi
Na rozdíl od	Na trhu v současné době neexistuje podobný produkt
Náš produkt	Usnadní práci v týmu a zjednoduší práci finančního poradce.

Tabulka 2: Pozice produktu na trhu

5.1.2 Charakteristika uživatelů

Název	Popis	Zodpovědnost	Zadavatel
Finanční poradce	Správa finančních portfolií	• Správa vlastních klientů • Správa finančního portfolia • Vytváření finančního programu	
Vedoucí týmu	Dohled nad prací své skupiny	• Kontrola vytvořených programů členů týmu • Koordinuje práci • Stejné funkce jako finanční poradce	

Tabulka 3: Uživatelé - souhrn

5.1.3 Produkt – přehled

Produkt - perspektiva

- Tento produkt je zcela uzavřený a nezávislý. S návazností na jiné systémy se nepočítá.

Předpoklady a závislosti

- Předpokladem nasazení produktu je operační systém Windows Server 2003 s Microsoft Office SharePoint 2007. Produkt pak bude možné nasadit a po konfiguraci jej uvést do funkčního stavu.

Funkce
Zaregistruj finančního poradce
Registruj platby
Spravuj aktuality
Pošli upozornění o ukončení licence
Nastavuj finanční údaje
Spravuj klienty
Vyhledej klienta
Přihlášení
Generuj seznam finančních poradců
Spravuj finanční portfolia
Zadání životního pojištění
Vyhledej finanční portfólio
Vytvoř finanční program
Spravuj tým

Tabulka 4: Nejdůležitější funkce

5.2 Iterační plán

Účelem tohoto dokumentu je vytvořit podklad, na jehož základě se bude možné orientovat v tom, jaký je současný stav projektu, dále na čem se aktuálně pracuje a jaký je plán. Určuje také konkrétní odpovědnost za jednotlivé zde popsané činnosti.

Lze tak určit jaká je pravděpodobnost, že se podaří dodržet požadovaný časový termín jednotlivých fází projektu a také konečný termín, do kdy má být projekt plně funkční.

Iterace		**Začátek**	**Konec**
Zahájení	I1	28. 10. 2009	4. 1. 2010
Rozpracování	E1	5. 1. 2010	26. 2. 2010
	E2	1. 3. 2010	30. 4. 2010
	E3	3. 5. 2010	13. 5. 2010
	E4	13. 5. 2010	20. 5. 2010
Konstrukce	C1	21. 5. 2010	30. 6. 2010
	C2	1. 7. 2010	19. 8. 2010
Předání	T1	20. 8. 2010	

Tabulka 5: Plán iterací

Iterace	**Cíl**
I1	Vize, Iterační plán, Seznam rizik, Vyhodnocení
E1	Nainstalování a nastavení serveru, Vyzkoušení použitelnosti navrženého frameworku
E2	případy užití, analytický model s vlastními stereotypy
E3	Vytvoření modelu architektury MOSS
E4	Výpis listů jen pomocí MOSS
C1	Uživatelské účty, skupiny, finanční programy
C2	Vytvoření vlastní webpart Finanční portfolia
T1	Sepsání

Tabulka 6: Cíle iterací

5.3 Seznam rizik

Tento dokument upozorňuje na možná rizika, která mohou nastat při realizaci projektu. Dále vytváří plán předcházení rizik a možné alternativy pokud dojde k naplnění některého rizika.

5.3.1 Časové riziko: nedodržení termínu

Velikost rizika	Velká
Popis	Při vytváření projektu se může stát, že se některá iterace nepodaří dodělat v termínu, který je stanovený iteračním plánem
Dopady	Nedokončení projektu
Ukazatele	Harmonogram iterací uvedených v iteračním plánu
Strategie předcházení rizika	Kontrola dodržování harmonogramu

Tabulka 7: Časové riziko: Nedodržení termínu

5.3.2 Technické riziko: Nezprovoznění serveru

Velikost rizika	Malá
Popis	Může nastat, že škola nezajistí server, který by byl přístupný ve veřejné síti.
Dopady	Projekt by se realizoval na lokálním počítači
Alternativní plán	Použití virtuálního operačního systému na vlastním počítači. Práce na systému by se musela realizovat pomocí backupů, které by si předávali členové realizačního týmu. Odpadla by tím však možnost paralelního vývoje aplikace.

Tabulka 8: Technické riziko: Nezprovoznění serveru

5.3.3 Technické riziko: Nepoužitelnost MOSS

Velikost rizika	Malá
Popis	Aplikace by měla být realizovaná prostřednictvím Microsoft Office Sharepoint. V iteraci E1 by se mělo otestovat, zda je MOS pro tvorbu aplikace vhodný.
Dopady	Řešení prostřednictvím MOS by nebylo možné a přešlo by se na alternativní plán
Ukazatele	V iteraci E1 se nepodaří vytvořit případ užití – správa klientů
Strategie předcházení rizika	Kontrola dodržování harmonogramu
Alternativní plán	Projekt by se realizoval pomocí nějakého jiného frameworku

Tabulka 9: Technické riziko: Nepoužitelnost Microsoft Office Sharepoint Server

5.3.4 Technické riziko: Nutnost programování vlastních webpart

Velikost rizika	Malá
Popis	Pokud by se nepovedl udělat výpis listů podle požadavků zadavatel. Musel by se výpis programovat pomocí vlastních webpart.
Dopady	Zvýšila by se náročnost na údržbu systému.
Ukazatele	Výpis listů není podle potřeb zákazníka
Strategie předcházení rizika	Důkladné nastudování možností MOOS
Alternativní plán	Programování vlastních webpart

Tabulka 10: Technické riziko: Nutnost programování vlastních webpart

5.4 Vyhodnocení

Dokument má za cíl popsat stav projektu k datu jeho vytvoření. Cílem jednotlivých verzí dokumentu je především srovnání situace projektu k datu jeho vytvoření. Tento dokument byl vytvořen jako vzorový pro další dokumenty vyhodnocení k určitému datu.

5.4.1 Aktuální rizika

Riziko	Popis	Alternativní plán	Stav
Nedodržení termínů	Při vytváření projektu se může stát, že se některá iterace nepodaří dodělat v termínu, který je stanovený iteračním plánem	Není	Nastalo
Nezprovoznění serveru	Může nastat, že škola nezajistí server, který by byl přístupný ve veřejné síti.	Použití virtuálního operačního systému na vlastním počítači. Práce na systému bude realizovaná pomocí backupů. Odpadá tím možnost paralelního vývoje aplikace	Nastalo
Nepoužitelnost Microsoft Office Sharepoint Server	Aplikace by měla být realizovaná prostřednictvím Microsoft Office Sharepoint. V iteraci E1 by se mělo otestovat, zda je MOSS pro tvorbu aplikace vhodný.	Projekt by se realizoval pomocí některého jiného frameworku.	Nenastalo
Technické riziko: Nutnost programování vlastních webpart	Pokud by se nepovedl udělat výpis listů podle požadavků zadavatel. Musel by se výpis programovat pomocí vlastních webpart.	Programování vlastních webpart	Nastalo

Tabulka 11: Rizika k určitému datu

5.4.2 Stav projektu

Jelikož se muselo přejít na alternativní plán, tak projekt skončil v iteraci E4.

5.4.3 Naplnění rozsahu projektu nebo produktu

Přechodem na alternativní plán se projekt zpozdil a momentálně stav projektu neodpovídá iteračnímu plánu.

5.4.4 Aktuální činnosti a jejich stav

Iterace	Činnost	Stav
I1	Vize, Iterační plán, Seznam rizik, Vyhodnocení, Vyzkoušení použitelnosti navrženého frameworku	Splněno
E1	Nainstalování a nastavení serveru, Ověření použitelnosti MOSS	Splněno
E2	Případy užití	Splněno
E3	Vytvoření modelu architektury MOSS	Splněno
E4	Výpis listů jen pomocí MOSS	Nesplněno (přechod na alternativní plán)

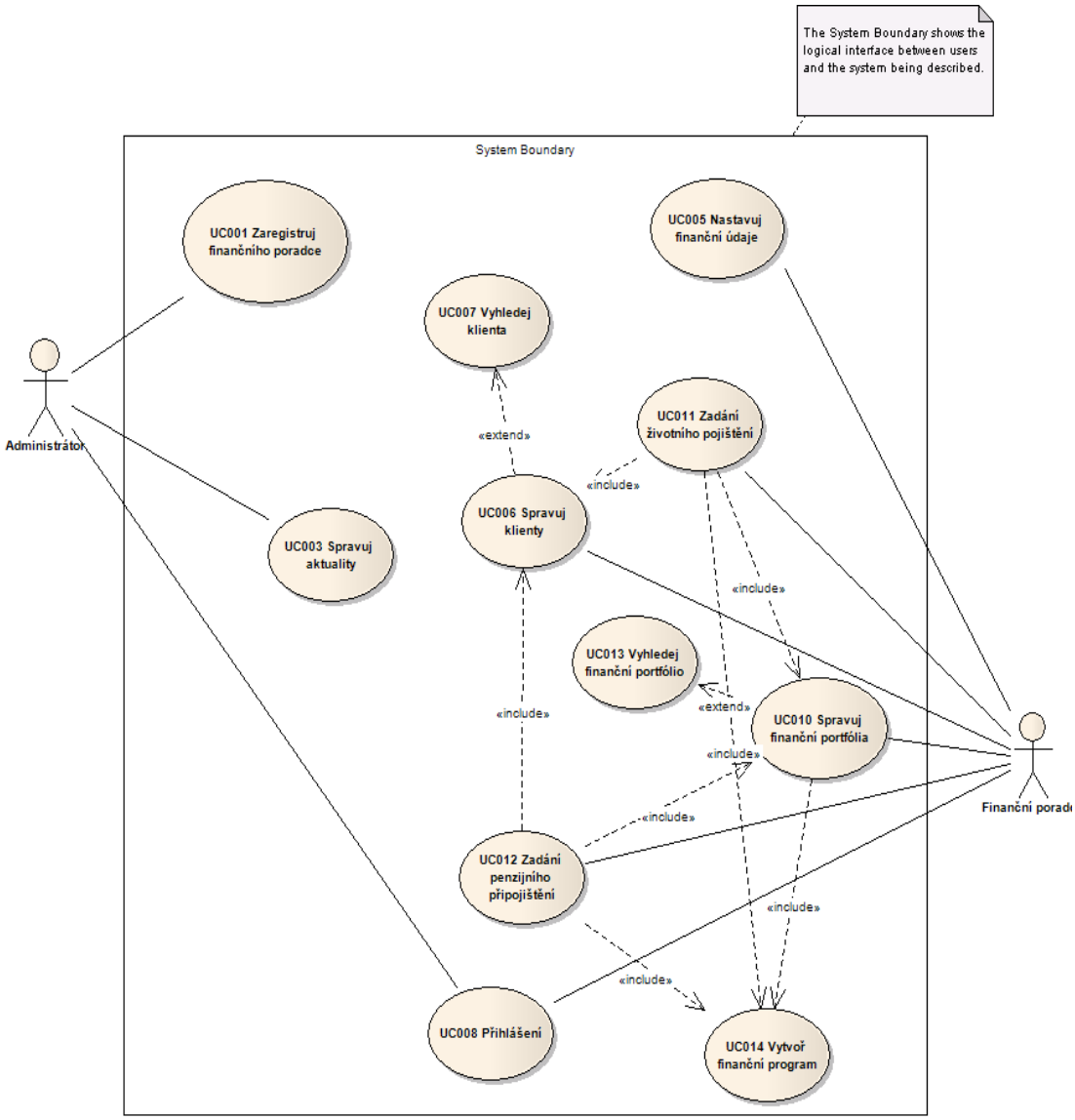
Tabulka 12: Činnosti a jejich stav

6 Modelování

Následující kapitola se soustředí na popis všech modelů, které během tvorby projektu vznikly podle definovaných případů užití. Podrobněji jsou zde popsány případy užití, struktura modelů a specifikace při modelováni v MOSS.

6.1 Případy užití

Podle požadavků zadavatele se jako první model vytváří model případů užití. V tomto modelu se hledají aktéři, kteří v systému vystupují a jejich případy užití. V našem systému máme dva aktéry, kterými jsou administrátor, a druhým aktérem je finanční poradce. Model případů užití je znázorněn na obrázku 2, kde si jednotlivé případy užití popíšeme podrobněji v následujících podkapitolách.



Obrázek 12: Model případů užití (vlastní zdroj)

6.1.1 UC001 Zaregistruj finančního poradce

Use-Case ID	UC001	
Hlavní aktér	Administrátor	
Vedlejší aktéři	Žádní	
Stručný popis	Administrátor zaregistruje nového finančního poradce do systému	
Vstupní podmínky	Administrátor je přihlášený do systému	
Trigger	Administrátor zvolí funkci Registrace nového finančního poradce	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	Administrátor vyplní registrační údaje o finančním poradci	Systém zaregistruje nového finančního poradce
Alternativní scénář		
Výstupní podmínky		

Tabulka 13: Případ užití UC001 Zaregistruj finančního poradce

6.1.2 UC003 Spravuj aktuality

Use-Case ID	UC002	
Hlavní aktér	Administrátor	
Vedlejší aktéři	Žádní	
Stručný popis	Administrátor přidá, upraví nebo odstraní aktualitu.	
Vstupní podmínky	Administrátor je přihlášený do systému	
Trigger	Administrátor zvolí funkci Správa aktualit	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	viz. stručný popis	zobrazení aktuality
Alternativní scénář		
Výstupní podmínky		

Tabulka 14: Případ užití UC003 Spravuj aktuality

6.1.3 UC005 Nastavuj finanční údaje

Use-Case ID	UC005	
Hlavní aktér	Finanční poradce	
Vedlejší aktéři	Žádní	
Stručný popis	Finanční poradce nastaví finanční údaje (úroková míra státu…), které budou předvyplněny v editačních polích při zadávání nového finančního programu nebo budou pracovat s finančně-matematickými vzorci.	
Vstupní podmínky	Finanční poradce je přihlášený do systému	
Trigger	Finanční poradce zvolí funkci Na stav finanční údaje	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	1. Finanční poradce vyplní finanční údaje.	
	2. Finanční poradce stiskne tlačítko Ulož finanční údaje.	Systém uloží finanční údaje pro daného finančního poradce.
Alternativní scénář		
Výstupní podmínky		

Tabulka 15: Případ užití UC005 Nastavuj finanční údaje

6.1.4 UC006 Spravuj klienty

Use-Case ID	UC006	
Hlavní aktér	Finanční poradce	
Vedlejší aktéři	Žádní	
Stručný popis	Finanční poradce přidá, upraví nebo odstraní klienta.	
Vstupní podmínky	Finanční poradce je přihlášený do systému	
Trigger	Finanční poradce zvolí funkci Přidat klienta	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	1. Finanční poradce vyplní informace o klientovi.	
	2. Finanční poradce stiskne tlačítko Ulož klienta.	Pokud klient neexistuje, systém uloží klienta.
Alternativní scénář		
Výstupní podmínky		

Tabulka 16: Případ užití UC006 Spravuj klienty

6.1.5 UC007 Vyhledej klienta

Use-Case ID	UC007	
Hlavní aktér	Finanční poradce	
Vedlejší aktéři	Žádní	
Stručný popis	Finanční poradce vyhledá klienta.	
Vstupní podmínky	Finanční poradce je přihlášený do systému	
Trigger	Finanční poradce zvolí funkci Vyhledat klienta	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	1. Finanční poradce vyplní informace o klientovi.	
	2. Finanční poradce stiskne tlačítko Vyhledat klienta.	Systém vypíše nalezené klienty, kteří se shodují s vyplněnými informacemi.
Alternativní scénář		
Výstupní podmínky		

Tabulka 17: Případ užití UC007 Vyhledej klienta

6.1.6 UC008 Přihlášení

Use-Case ID	UC008	
Hlavní aktér	Administrátor, Finanční poradce	
Vedlejší aktéři	Žádní	
Stručný popis	Podle uživatelského jména a hesla se přihlásí do systému buď administrátor, nebo finanční poradce.	
Vstupní podmínky		
Trigger	Administrátor nebo Finanční poradce zvolí funkci Přihlášení do systému	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
Alternativní scénář		

Tabulka 18: Případ užití UC008 Přihlášení

6.1.7 UC010 Spravuj finanční portfolia

Use-Case ID	UC010	
Hlavní aktér	Finanční poradce	
Vedlejší aktéři	Žádní	
Stručný popis	Finanční poradce přidá, upraví nebo odstraní finanční portfólium	
Vstupní podmínky	Finanční poradce je přihlášený do systému	
Trigger	Finanční poradce zvolí funkci Přidat finanční portfólium	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	1. Finanční poradce vyplní informace o finančním portfóliu (název)	
	2. Finanční poradce stiskne tlačítko Ulož finanční portfólium.	Systém uloží finanční portfólium (identifikace podle názvu, data vytvoření).
Alternativní scénář		
Výstupní podmínky		

Tabulka 19: Případ užití UC010 Spravuj finanční portfolia

6.1.8 UC011 Zadání životního pojištění

Use-Case ID	UC011	
Hlavní aktér	Finanční poradce	
Vedlejší aktéři	Žádní	
Stručný popis	Finanční poradce zadá nový finanční program – životní pojištění.	
Vstupní podmínky	Finanční poradce je přihlášený do systému.	
Trigger	Finanční poradce zvolí funkci Zadání životního pojištění	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	Krok 1: Finanční poradce vyhledá klienta.	Systém přiřadí životnímu pojištění klienta.
	Krok 2: Finanční poradce vyhledá finanční portfólium.	Systém přiřadí životnímu pojištění finanční portfólium.
	Krok 3: Finanční poradce zaregistruje životní pojištění.	Proběhne výpočet zbývajících položek životního pojištění pomocí finančně-matematických vzorců. Systém přidá životní pojištění.
Alternativní scénář	**Alternativní scénář 1:** Klient má více než jedno životní pojištění: Krok 3 se v tomto případě opakuje (finanční poradce registruje více životních pojištění).	
Výstupní podmínky	Životní pojištění je zaregistrováno.	

Tabulka 20: Případ užití UC011 Zadání životního pojištění

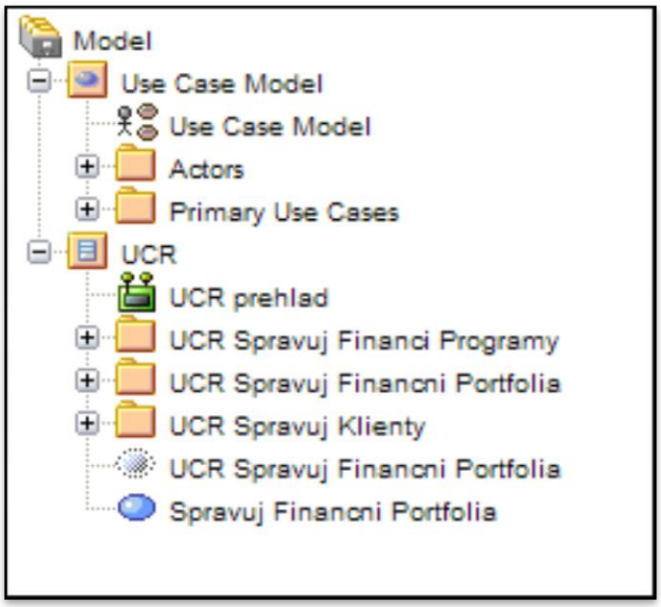
6.1.9 UC012 Zadání penzijního připojištění

Use-Case ID	UC012	
Hlavní aktér	Finanční poradce	
Vedlejší aktéři	Žádní	
Stručný popis	Finanční poradce zadá nový finanční program – penzijní připojištění.	
Vstupní podmínky	Finanční poradce je přihlášený do systému.	
Trigger	Finanční poradce zvolí funkci Zadání penzijního připojištění	
Hlavní scénář	**Chování aktéra**	**Odpověď systému**
	Krok 1: Finanční poradce vyhledá klienta.	Systém přiřadí penzijnímu připojištění klienta.
	Krok 2: Finanční poradce vyhledá finanční portfólium.	Systém přiřadí penzijnímu připojištění finanční portfólium.
	Krok 3: Finanční poradce zaregistruje penzijní připojištění.	Proběhne výpočet zbývajících položek penzijnímu připojištění pomocí finančně-matematických vzorců. Systém přidá penzijní připojištění.
Alternativní scénář	**Alternativní scénář 1:** Klient má více než jedno penzijní připojištění: Krok 3 se v tomto případě opakuje (finanční poradce registruje více penzijních připojištění).	
Výstupní podmínky	Penzijní připojištění je zaregistrováno.	

Tabulka 21: Případ užití UC012 Zadání penzijního připojištění

6.2 Struktura modelů

Při modelování v programu Enterprise Architect jsme si nejprve museli zvolit strukturu modelů. Modely jsou rozděleny do jednotlivých balíčků podle případu užití. Na následujícím obrázku je znázorněna tato struktura.

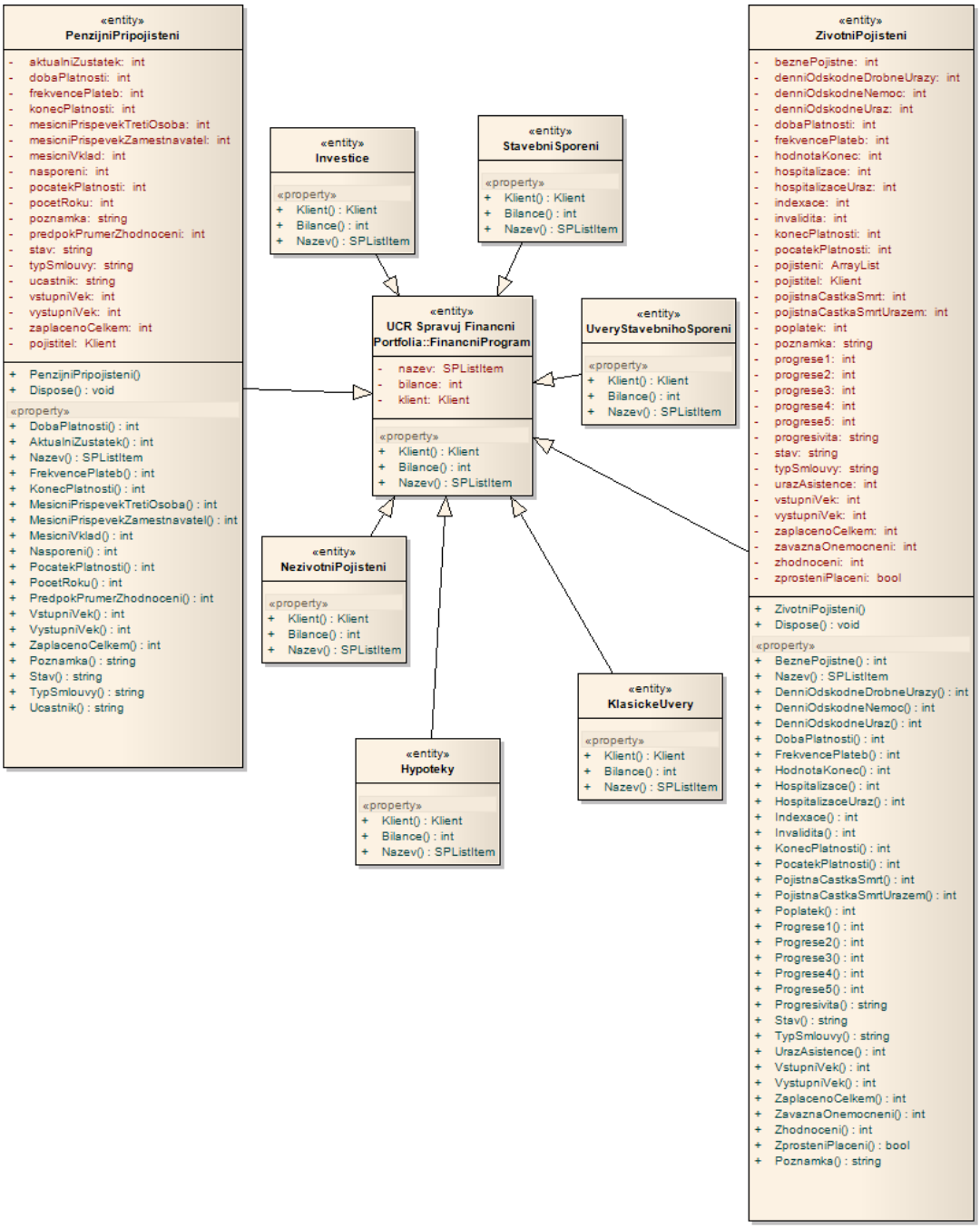


Obrázek 13: Struktura modelů (vlastní zdroj)

První balíček je model případů užití, který se dělí na další 2 balíčky. V prvním s názvem *Actors* máme uložený aktéry, kteří vystupují v systému. V druhém balíčku, který se jmenuje *Primary Use Cases*, jsou všechny případy užití a model, který nám určuje závislosti mezi aktéry a jednotlivými případy užití.

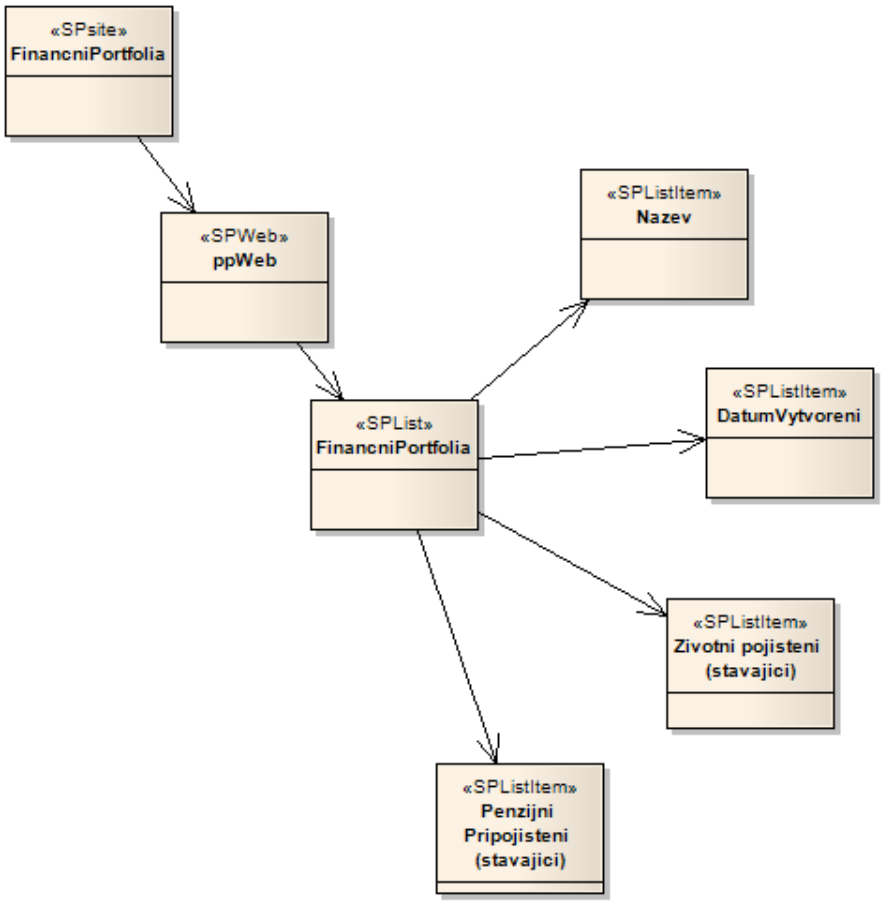
Po vytvoření modelu případu užití bylo nutné tento model zrealizovat. Dalším balíčkem, který byl vytvořen je UCR (Use Case Realization). Pro realizaci byly vybrány 3 případy užití a ty jsou v balíčcích *UCR Spravuj Financni Programy, UCR Spravuj Financni Portfolia a UCR Spravuj Financni klienty.*

První balíček obsahuje všechny programy, které finanční poradce může uzavírat se svým klientem. V modelu je využita generalizace obecné třídy *FinancniProgram,* od které ostatní třídy, které představují konkrétní finanční programy, dědí atributy *nazev, bilance* a *klient*. Pro implementaci v našem systému byly zvoleny dva finanční programy, které se uzavírají nejčastěji, a ty byly namodelovány podrobněji. Tyto třídy se jmenují *PenzijniPripojisteni* a *ZivotniPojisteni.* Na následujícím obrázku je tento model znázorněn.



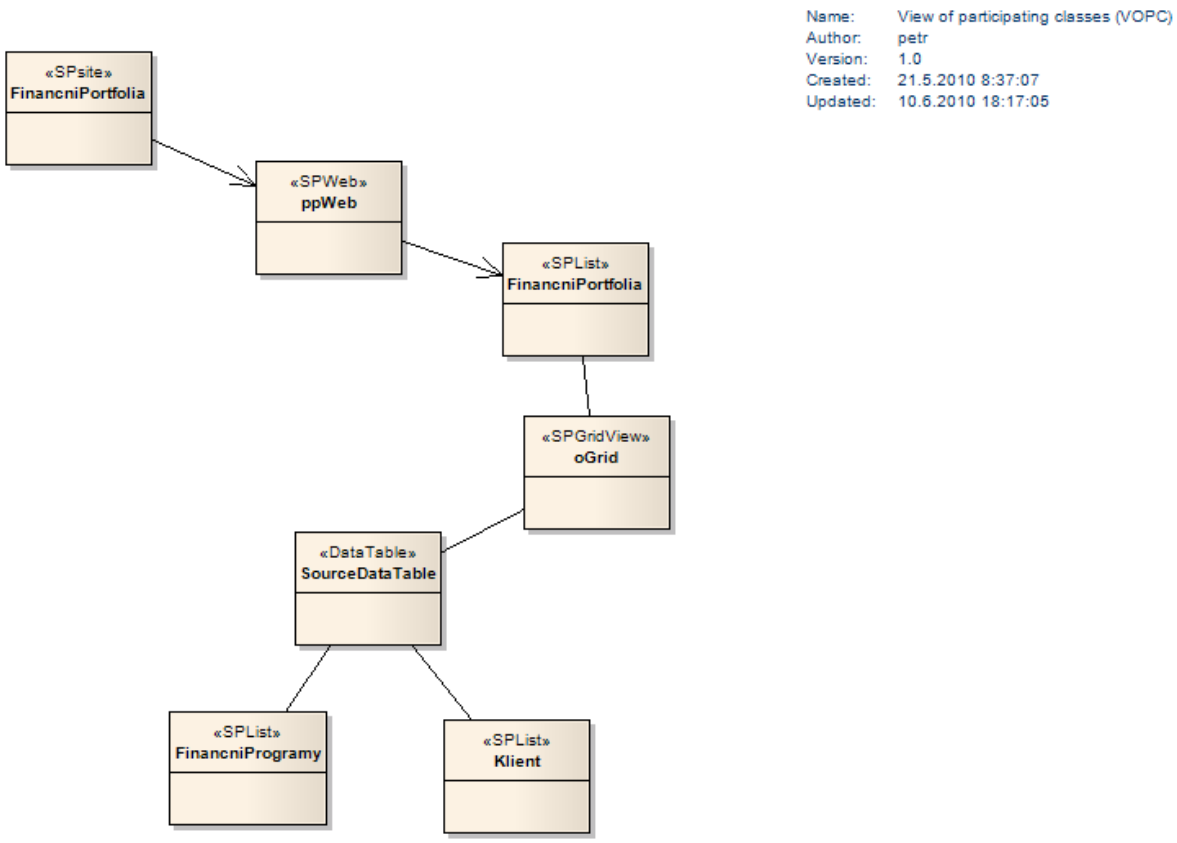
Obrázek 14: Model UCR Spravuj Financni Programy (vlastní zdroj)

Druhý balíček, který realizuje případ užití *UC010 Spravuj finanční portfolia*, obsahuje 3 základní modely. První model sloužící k výpisu dat na webové stránce, se nazývá *Spravuj Financni Porfolia WebPageView*. Cílem tohoto modelu bylo vytvořit přehlednou formu vypisovaných dat do objektu třídy DataGridView, který je standardní součástí .NET frameworku. Tento model je znázorněný na obrázku 15.



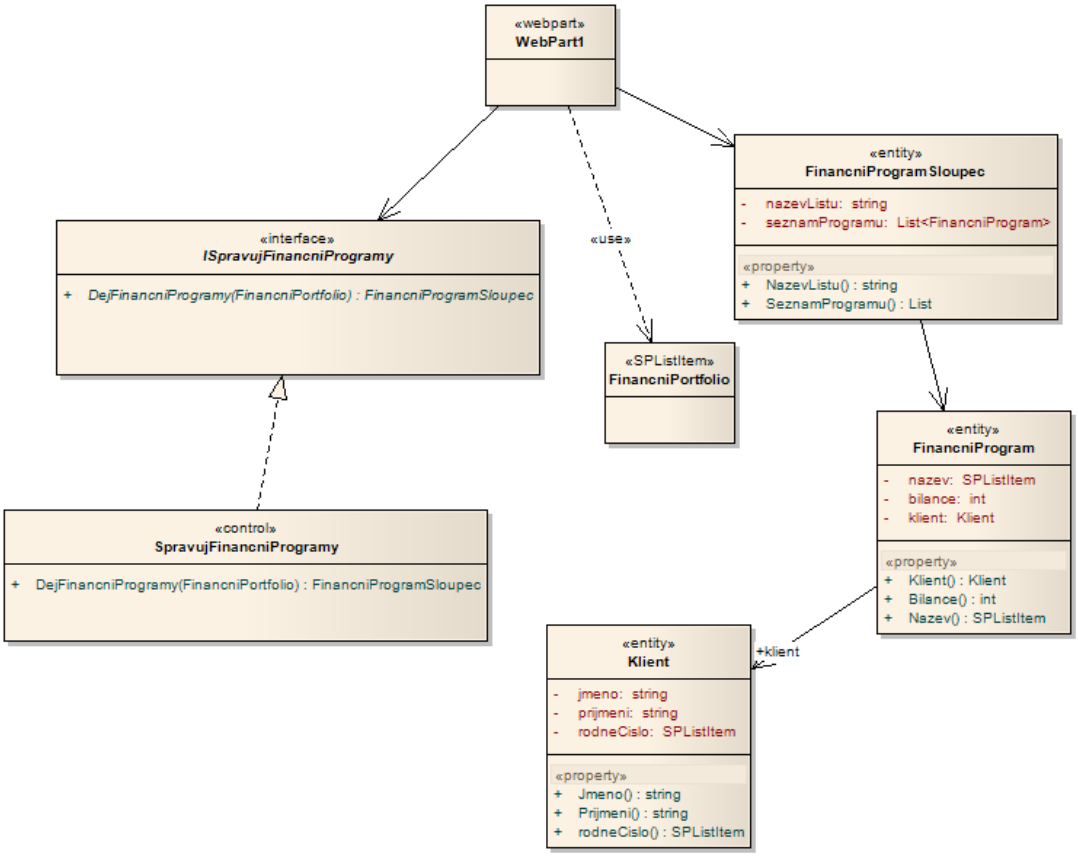
Obrázek 15: Model Spravuj Financni Porfolia WebPageView (vlastní zdroj)

Další model, který se nachází v balíčku *UCR Spravuj Financni Portfolia*, se nazývá *View of participating classes.* Tento model byl vytvořen, aby demonstroval, s jakými objekty se v MOSS pracuje a jakým způsobem tyto objekty používáme v našem systému. Tento model nám taktéž znázorňuje propojení objektů, které jsou součástí Sharepointu a objekty, které jsou standardně používány v .NET frameworku. Návrh tohoto modelu nám v tomto případě znázorňuje výpisy dat do standardní komponenty ASP.NET *DataGridView*. Tento model je znázorněn na následujícím obrázku.



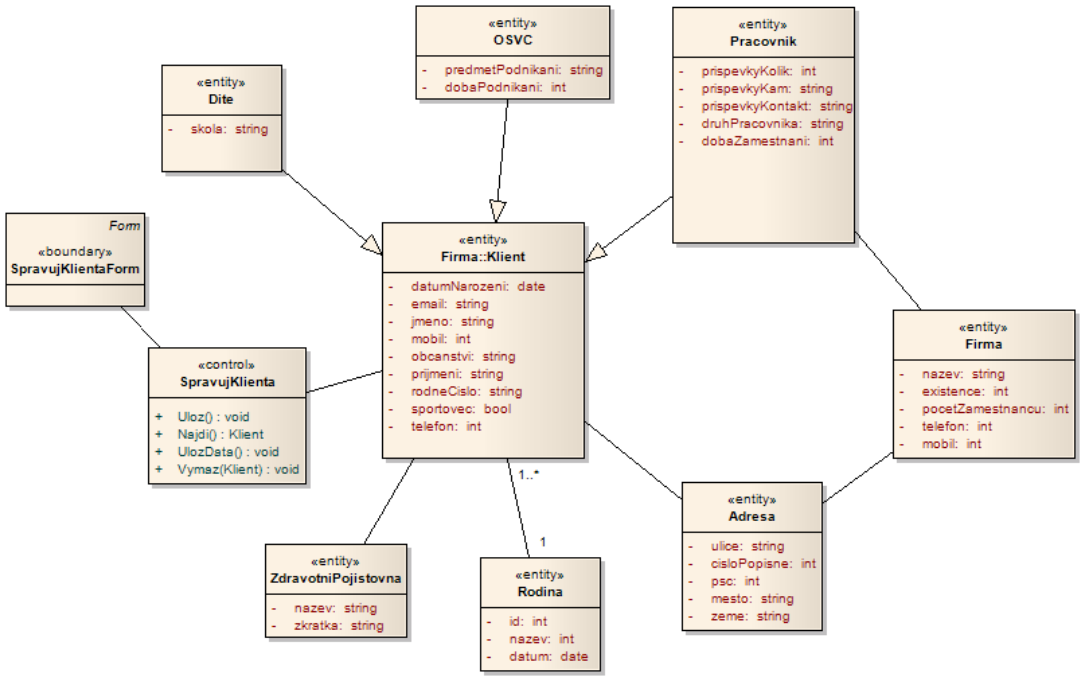
Obrázek 16: Model View of participating classes (vlastní zdroj)

Poslední model, který je v balíčku obsažen, se jmenuje *UCR Spravuj Financni Portfolia.* Tento model byl vytvořen pro implementaci vlastní webparty. V tomto modelu jsme se snažili zpřehlednit kód tím, že jsme u modelu chtěli oddělit datovou část od funkční logiky. Takto zvolená architektura se nazývá MVC (Model-view-controller). Třída, která má stereotyp *control*, se nazývá *SpravujFinancniProgramy*. Tato třída má jednu jedinou metodu nazvanou *DejFinancniProgramy.* Tato metoda vrací objekt třídy *FinancniProgramSloupec,* která uchovává název listu, z kterého se čtou všechny potřebné údaje uložené V MOSS, a seznam všech finančních programů obsažené v portfoliu. Tento model je znázorněn na následujícím obrázku.



Obrázek 17: Model UCR Spravuj Financni Portfolia (vlastní zdroj)

Poslední balíček, který je obsažen v balíčku UCR, realizuje případ užití *UC006 Spravuj klienty*. Pro tento případ byl vytvořený obecný analytický model znázorňující propojení jednotlivých tříd. Tento model je na následujícím obrázku.



Obrázek 18: Analytický model UCR Spravuj klienty (vlastní zdroj)

6.3 Modelování v SharePointu

Microsoft SharePoint nabízí vysoce strukturovaný objektový model, který usnadňuje přístup k objektům, které reprezentují různé aspekty webu *SharePoint Web*. Od nejvyšších úrovní objektů, můžeme přejít hierarchií objektů a získat objekt, který chceme použít v kódu.

Objekty MOSS jsou tedy seřazeny hierarchicky, kde na nejvyšším stupni hierarchie využíváme objekt SPSite. Objekt SPSite představuje soubor logicky souvisejících objektů SPWeb. Takovýto soubor se nazývá kolekce.

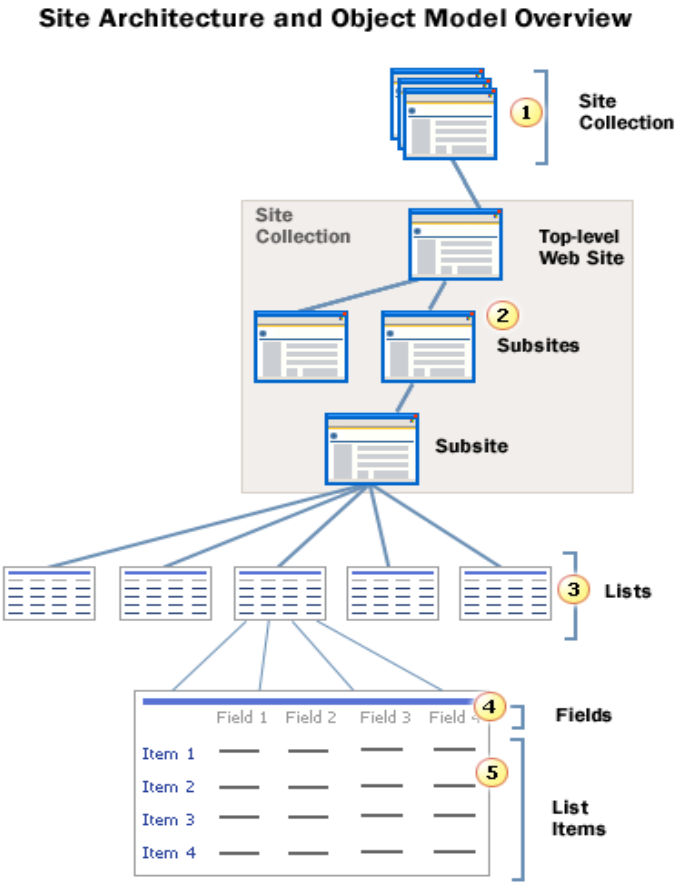
Každá kolekce webů obsahuje libovolné množství objektů SPWeb a každý objekt má členské funkce, které mohou být použity pro správu webu, včetně jeho schémat a šablon, stejně jako přístup ke složkám a souborům. Vlastnosti webu vrací objekt SPWebCollection, který reprezentuje všechny podřízené weby určitého webu. List vrací objekt SPListCollection, který reprezentuje všechny seznamy webu.

Pro správu seznamů nebo pro přístup k položkám seznamu se používá objekt SPList. Objekt SPListItem obsahuje jeden řádek v seznamu. Vlastnost *Fields* vrací SPFieldCollection, který reprezentuje všechna pole nebo sloupce v seznamu. Vlastnost *Items* vrací objekt SPListItemCollection, který reprezentuje všechny řádky nebo položky v seznamu.

Každý objekt SPField má členy obsahující nastavení pro pole.

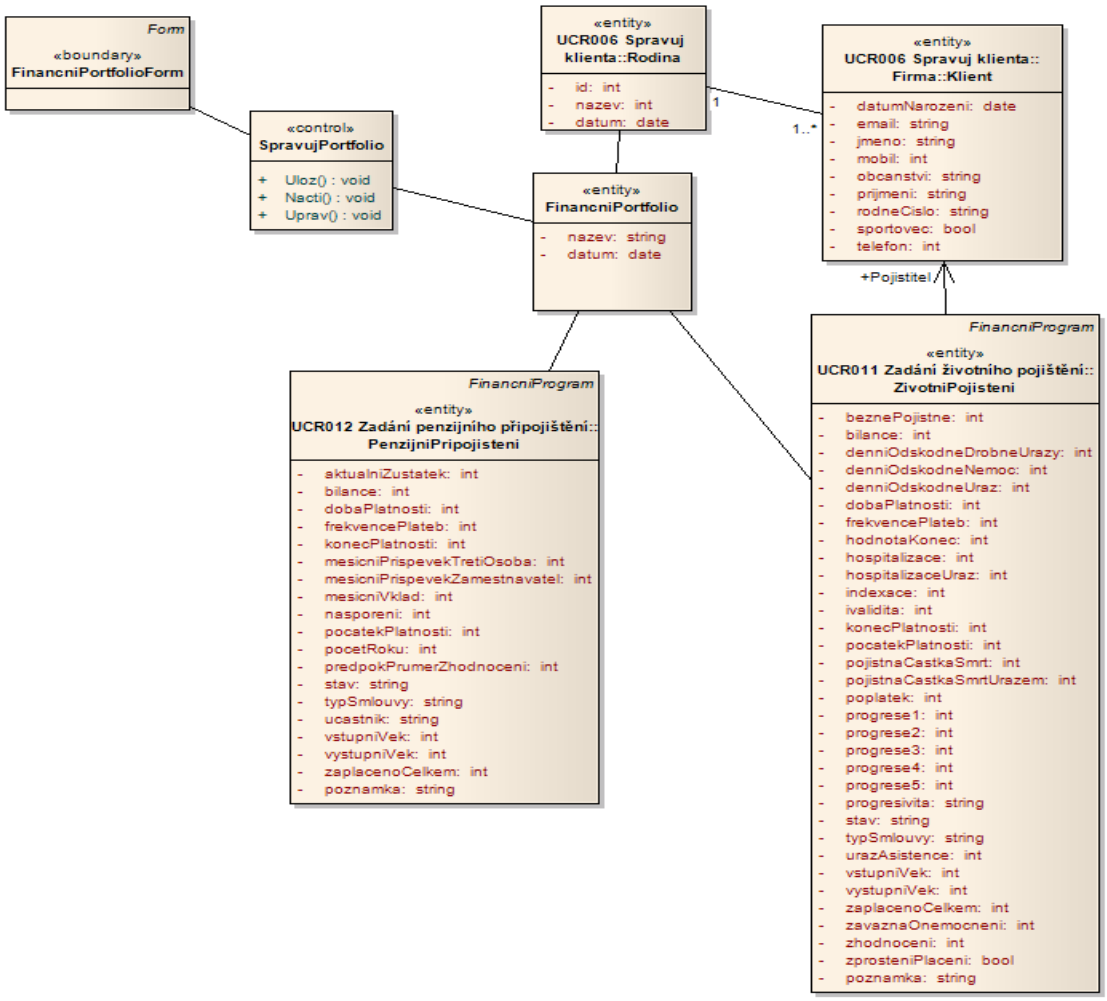
Každý SPListItem reprezentuje jednotlivý řádek seznamu.

Podrobnější ukázka hierarchické struktury je znázorněna na obrázku 19.



Obrázek 19: Znázornění webové architektury ve vztahu ke kolekcím a objektům (Zdroj: http://msdn.microsoft.com)

Při dodržení takto znázorněné struktury bylo zapotřebí vytvořit model, který by reprezentoval případy užití. Pro realizaci jsme si zvolili případ užití s nejvyšším stupněm rizika, což spočívalo v tom, že pokud jsme nechtěli programovat vlastní webparty, museli jsme najít standardní řešení pomocí rozhraní přímo v Sharepointu. Případ užití, který jsme si pro implementaci vybrali, se jmenuje *UCR010 Spravuj finanční portfolio.* Na 20. obrázku je znázorněn obecný analytický model tohoto případu užití.



Obrázek 20: Analytický model případu užití UCR010 Spravuj finanční portfolio (vlastní zdroj)

7 Závěr

Cílem této diplomové práce bylo vytvořit postup pro tvorbu informačního systému za použití nějakého standardního frameworku. Pro vytvoření postupu jsem vycházel z metodického framworku RUP, který je podrobněji popsán v teoretické části této práce.

V praktické části jsem se zaměřil na vývoj informačního systému pro správu finančních portfolií. Tento systém by měl ulehčit práci finančním poradcům tím, že jim automaticky vypočítává rozdíl mezi stávajícími a optimalizovanými finančními programy. Dalším požadavkem na systém bylo, aby finanční poradci mohli mezi sebou spolupracovat v rámci jednoho týmu, na který by dohlížel manažer této skupiny.

Nejdříve bylo potřeba vybrat framework, pomocí něhož by se informační systém implementoval. Po prozkoumání různých frameworků, jsme se rozhodli postavit systém na Microsoft Office Sharepoint Serveru (MOSS), jenž umožňuje vytvářet systémy pro vzájemnou spolupráci.

Po prozkoumání možností MOSS byly pomocí RUP stanovené postupy vývoje informačního systému a vytvořené modely, které museli splňovat specifické vlastnosti MOSS.

Přínosem této diplomové práce je praktická ukázka, jak se systematicky řídit při vývoji informačního systému podle metodického frameworku RUP.

8 Zdroje

[1] *IBM* [online]. 2005 [cit. 2010-08-18]. Rational Unified Process: Best practices for software development teams. Dostupné z WWW: <http://www.ibm.com/developerworks/rational/library/253.html>.

[2] JULINEK, Pavel. *Použití RUP pro malé SW projekty*. [s.l.], 2008. 76 s. Diplomová práce. Masarykova univerzita.

[3] ALDORF, Filip. *Metodika RUP* [online]. 2005 [cit. 2007-04-11]. Dostupný z WWW: <http://objekty.vse.cz/Objekty/RUP>.

[4] *Capturing Architectural Requirements* [online]. 2005 [cit. 2010-08-18]. Capturing Architectural Requirements. Dostupné z WWW: <http://www.ibm.com/developerworks/rational/library/4706.html>.

[5] *ZEND framework* [online]. 2010 [cit. 2010-08-18]. Dostupné z WWW: <http://framework.zend.com/>.

[6] *Kohana: The Swift PHP Framework* [online]. 2010 [cit. 2010-08-18]. Dostupné z WWW: <http://kohanaframework.org/>.

[7] *DotNetNuke* [online]. 2010 [cit. 2010-08-18]. Dostupné z WWW: <http://www.dotnetnuke.cz/>.

[8] *ASP.NET MVC* [online]. 2010 [cit. 2010-08-18]. Dostupné z WWW: <http://www.asp.net/mvc>.

[9] *Office SharePoint Server* [online]. 2010 [cit. 2010-07-18]. Co je Microsoft Office SharePoint Server?. Dostupné z WWW: <http://www.microsoft.com/cze/sharepoint/produkt/podrobnosti.aspx>.

[10] *MSDN* [online]. 2010 [cit. 2010-07-21]. Dostupné z WWW: <http://msdn.microsoft.com/cs-cz/default.aspx>.

[11] Rational Method Composer [online]. [s.l.] : [s.n.], 2006 [cit. 2010-08-18]. Dostupné z WWW: <http://www-01.ibm.com/software/awdtools/rmc/>.