VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

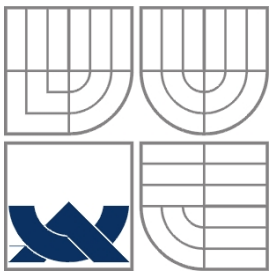
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROGRAMOVÁ PODPORA MANAGEMENTU PROJEKTŮ

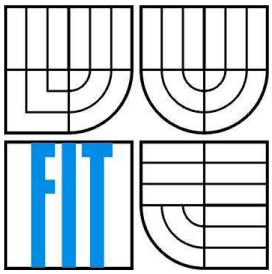
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE PETR JANDA
AUTHOR

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

PROGRAMOVÁ PODPORA MANAGEMENTU PROJEKTŮ

SOFTWARE SUPPORT FOR PROJECT MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Petr Janda
AUTHOR

VEDOUCÍ PRÁCE — Ing. Květoňová Šárka
SUPERVISOR

BRNO 2007

Zadání bakalářské práce/3961/2006/xjanda11

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Janda Petr**
Obor: Informační technologie
Téma: **Programová podpora managementu projektů**
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s procesy řízení projektů. Zaměřte se na proces plánování a realizace softwarového produktu.
2. Prostudujte současné možnosti a přístupy k tvorbě SW.
3. Specifikujte požadavky na programovou podporu projektového řízení IT projektů. Zaměřte se na časové a finanční hledisko při vývoji SW produktů.
4. Navrhněte koncepci podpory řízení IT projektů.
5. Implementujte prototyp navrženého systému ve vhodně zvoleném prostředí.
6. Prověřte funkčnost realizovaného systému. Demonstrujte na vhodně zvoleném vzorku dat.
7. Zhodnoťte dosažené výsledky se zaměřením na použitelnost v reálném prostředí. Diskutujte možnosti dalšího rozvoje realizovaného systému.

Literatura:

- Rosenau, M.D.: Řízení projektů, Computer Press, 2003, 344 s. ISBN 80-7226-218-1
- Kliem, L. R.: Project management methodology. Marcel Dekker Inc., 1997.
- Adamec F.: Řízení projektů pomocí Project 2000. Grada Publishing, 2001, 232 s. ISBN 80-7169-793-1
- Schulte, P.: Complex IT Project Management. AUERBACH PUBLICATION, 2004, ISBN 0849319323

Při obhajobě semestrální části projektu je požadováno:

- Body 1-4.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Květoňová Šárka, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Petr Janda**
Id studenta: 84347
Bytem: Družstevní 592, 793 26 Vrbno pod Pradědem
Narozen: 16. 11. 1984, Bruntál
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Programová podpora managementu projektů
Vedoucí/školitel VŠKP: Květoňová Šárka, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

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.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Tento projekt se zabývá návrhem informačního systému pro podporu projektového managementu v IT organizacích. Představuje základní aspekty řízení projektů a zejména moderní metodologii Rational Unified Process, která se orientuje na řízení projektů v dnešním dynamickém prostředí. Systém je implementován v prostředí Microsoft Visual Studia 2005 s podporou Microsoft SQL Serveru 2005 a skládá se z ASP.NET webové prezentace a DLL knihovny napsané v jazyce C#. Navržený informační systém poskytuje podporu pro plánování a umožňuje řízení průběhu projektu, nezávysle na geografické lokalizaci realizačního týmu, který může se systémem pracovat přes standardní webové prohlížeče.

Klíčová slova

Projektové řízení, Rational unified process, RUP, projekt, úkol, workflow, diagram tříd, use case diagram, UML, 3 vrstvá architektura, prezenční vrstva, aplikační vrstva, databázová vrstva, databáze, DataSet, AJAX, AJAXControlToolkit, ControlAdapters, C#, ASP.NET, Microsoft Visual Studio 2005, MSSQL

Abstract

This project describes design and implementation of project management information systém support, mainly for IT organizations. It includes informations about basic facts in project management and mainly about Rational Unified Process, the modern metodology which is well useful in today changeable work environment. System is implemented in Microsoft Visual Studio 2005 with support of Microsoft SQL Server 2005 and consists of ASP.NET presentation and of DLL library written in C# language. Information system provides support for project planing and management. System work in standard internet browsers and is very useful for geografically devided teams.

Keywords

Project management, Rational unified process, RUP, project, task, workflow, class diagram, use case diagram, UML, 3 layered architecture, presentation layer, business layer, database layer, database, DataSet, AJAX, AJAXControlToolkit, ControlAdapters, C#, ASP.NET, Microsoft Visual Studio 2005, MSSQL

Programová podpora managementu projektů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Šárky Květoňové. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Janda
9.5.2007

Poděkování

Ing. Šárce Květoňové za asistenci při tvorbě systému.

Obsah

Úvod

Jen málokdo z nás si v dnešní době dovede představit život bez počítače. Jejich neustálý a rychlý vývoj stále zvětšuje jejich možnosti a softwarové systémy, které jsou nasazovány se stávají velmi komplexními.

Již v 80. letech 20. století vznikaly první metodiky pro řízení projektů v IT organizacích, zejména v těch, které se podílely na vývoji globálních produktů či ve vojenské sféře. Vývoj pokračoval dále ruku v ruce s vývojem samotných počítačů a dnes již existuje řada velmi komplexních metodik, které se vývojem softwaru zabývají. Jejich správnou aplikací se snaží jednotlivé organizace vyvíjet projekty předvídatelným způsobem s měřitelnými hledisky úspěšnosti. Jedině tak mohou dosahovat kontinuálního růstu kvality a komplexního zlepšování vývoje softwaru.

Tento dokument vznikl jako dokumentace k projektu vývoje softwaru, určeného pro podporu projektového managementu. Teoretická část této práce představuje projektový management, jeho základní principy, částí a zejména jednu z nejmodernějším metodik – *Rational unified process*, která je úspěšná i při vývoji nejsložitějších softwarových systémů. Následně se v praktické části dozvíte informace z analýzy, návrhu a implementace informačního systému, který je určen jako podpora projektového managementu i s využitím základních principů *RUP*.

1 Projektový management

Dnešní projekty, zejména v oblasti informačních technologií, jsou stále komplexnější. Dosažení stanovených cílů je tak stále těžší a vyžaduje sofistikované metody a postupy. Právě vzrůstající požadavky na vysokou úroveň a bezchybnost výsledných produktů, dodávaných v co nejkratším čase s co nejmenšími náklady daly vzniknout nové disciplíně, projektovému managementu. Vytvořením komplexního pohledu na vývoj projektu se snaží nalézt maximálně výhodné řešení. Nemá však velmi výhodné podmínky. Jedním z největších problémů kterému musí čelit je dynamické prostředí ve kterém musí pracovat. Není totiž výjimkou, že během vývoje, který může mnohdy trvat i několik let, se mohou požadavky velmi změnit. Je tak nutno zavádět složité principy a procesy, kterými se snažíme předejít problémům, které mohou být do projektu vneseny. Více o problematice správy dynamických žadavků na projekt najdete v publikaci *Agile Project Management: How to Succeed in the Face of Changing Project Requirements* [9].

Než si vysvětlíme jak takový projekt řídit musíme si říct co si pod pojmem **projekt** představit.

Jedná se o unikátní posloupnost činností jejímž výsledkem musí být splnění předem stanoveného cíle. Projekt má svůj počátek a konec a je vytvářen lidmi, mnohdy z různých profesí. V řízení projektu se prolínají 3 základní roviny, jedná se tedy o tzv. trojimperativ – rovina věcná, nákladová a časová.

Základními veličinami, které vstupují do hry při rozhodování jsou:

- podoba výsledného produktu a jeho kvalita (věcná rovina)
- náklady na jeho vytvoření (nákladová rovina)
- čas a termíny dodání (časová rovina)

Vždy je potřeba vytvořit kompromisní řešení. Extrémní upřednostňování některého z těchto aspektů může vést k velkým problémům a nebo také k ukončení projektu bez dosažení jeho cíle. Je nutné si uvědomit, že projekt musí být natolik kvalitní, aby uspokojil všechny požadavky, které jsou na produkt kladeny. Následné zvyšování kvality může způsobovat zbytečné plýtvání, nebo prodloužení projektu i když to není nutné.

V dnešním světě 21. století jdou požadavky ještě dále. Důležitost správného řízení projektů roste přímo úměrně nárokům, které by se dali shrnout třemi slovy:

- Kvalitněji
- Levněji
- Rychleji

V tvrdém souboji vítězí právě ti, kteří dokáží nejvíce splnit tyto požadavky. Dá se tak očekávat velký rozvoj disciplíny projektového managementu, která je bezpochyby klíčovým aspektem, který ovlivňuje celý vývoj od počátečních analýz až po jeho ukončení.

1.1 Definice a fáze managementu projektů

Pojďme si říci co projektový management je. Jedná se o kompexní postup optimalizace procesu vývoje použitím různých znalostí, technik a nástrojů. Cílem je uspokojení požadavků všech zájmových skupin, které se na projektu podílejí (Přednáška 1., [6]).

Úkolem projektového manažera je naplánování a řízení procesu vývoje, kontrolování kvality a dodržování požadavků, časového harmonogramu a naplánovaného rozpočtu.

Z pohledu životního cyklu se projekt skládá z následujících fází:

1.1.1 Analýza a definice projektu

Jedná se o fázi, ve které se snažíme zejména definovat všechny cíle projektu. Pro následný návrh a implementaci je tak nutno sestavit seznam požadavků a funkční specifikaci výsledného produktu. Nutno však připomenout, že požadavky se budou měnit i během projektu a tak v dnešních profesionálních metodikách vývoje probíhá etapa analýzy a návrhu opakovaně.

Analýza je však vždy počáteční fázi projektu a měla by odhalit možnosti a rizika projektu. Jejími základními úkony je provedení analýzy požadavků, vytvoření studie proveditelnosti (musíme zjistit zda máme možnost s dostupnými prostředky za daný čas realizovat daný projekt), funkční specifikace (jako jeden ze základních dokumentů pro návrh a implementaci), prvotní analýzy nákladů (podle nároků kladených na projekt) a stanovení cílů projektu. V této fázi je nutné zvážit všechny alternativy a možnosti jak projekt realizovat.

1.1.2 Návrh a organizování

Po analýze a definici projektu následuje fáze návrhu a organizace. Je třeba sestavit konkrétní plán implementace, jež je základním dokumentem, který slouží k řízení projektu. Vzhledem k tomu, že již máme detailnější informace o průběhu projektu můžeme také provézt přesnější odhad ceny. Z plánu projektu a návaznosti jednotlivých činností jsme schopni také stanovit předpokládanou délku a termín dokončení.

Součástí návrhu je také definice požadavků na pracovní sílu, kterou je potřeba do projektu začlenit. Nezbytnou organizační aktivitou je tedy sestavení týmu. Jedná se o složitý úkol projektového manažera, jelikož musí pečlivě volit pracovníky, kteří svou kvalifikací co nejlépe vyhovují ke splnění jednotlivých úkolů v rámci projektu. Je potřeba volit pracovníky, od kterých se předpokládá správné plnění úkolů v rámci jejich kvalifikace, není však dobrou volbou vybírat

pracovníky, kteří evidentně svými dovednosti a kvalifikací překračují požadavky, které na ně budou kladeny. Mohlo by tak dojít k neúměrnému prodražení projektu a zbytečnému plýtvání nevyužitím jejich potenciálu.

Samotný návrh projektu je prováděn rozdělením projektu do dílčích celků. Projekt je rozdělen do etap. Každá etapa by měla mít jako svůj výsledek vytvoření části výsledného projektu. Může se jednat například o vývoj některé z klíčových komponent. Výsledek jedné etapy je tak ověřitelný a testovatelný a může být předáván jako vstup etapy, která bude následovat. Ukončení etapy a kontrola jejího výsledku je jedním z důležitých milníků a její součástí je také kontrola a porovnávání aktuální časové a finanční náročnosti s plánem projektu.

Jednotlivé etapy jdou členit dále do několika kroků. Jedná se o jednotku jejímž výsledkem je určitý výsledek, který slouží k realizaci cíle dané etapy. Může se jednat například o vytvoření datového modelu nebo definici databázových tabulek.

V rámci jednoho kroku může dále proběhnout několik činností, které jsou elementární jednotkou při plánování projektu.

Samozřejmostí je, že projekt není již od prvopočátku plánován na jednotlivé činnosti. Počáteční návrh projektu musí nutně obsahovat plánování alespoň na úrovni etap projektu s jasně definovanými vstupy a výstupy. Během projektu dále dochází ke zpřesňování plánu projektu až na úroveň činností v jednotlivých krocích. Plánování na úrovni činností je však velmi závislé na aktuálním průběhu projektu a musí být flexibilní. Je tedy nemožné ještě před započetím implementace naplánovat projekt na nejnižší možnou úroveň. Takový plán by byl obtížně proveditelný, jelikož by neumožňoval reagovat i na sebemenší změny či problémy které mohou v průběhu vývoje nastat.

1.1.3 Implementace

Máme-li hotovou analýzu a návrh projektu, můžeme začít se samotnou implementací. V této fázi začíná vznikat konkrétní podoba realizace projektu. Veškeré aktivity a procesy probíhají podle plánu, který byl sestaven v předchozí etapě. Pro projektového manažera nastává v této fázi důležitá nová role, kterou je řízení a kontrola projektu. Je nutné, aby byl vývoj projektu monitorován a v případě odchylek od jeho plánu patřičně řízen. Vždy je nutné kontrolovat hlediska, která jsou pro vývoj projektu důležitá.

Jedním z kontrolovaných hledisek musí být čas. Vývoj projektu by měl probíhat podle naplánovaného harmonogramu a v případě odchylek by měl projektový manažer patřičně reagovat. Objeví-li se totiž v projektu náznaky jeho možného zpoždění, je potřeba se zamyslet proč tomu tak je. Příčin může být několik z nichž některé jsou velmi závažné. Uveďme si několik případů:

- **Přetížení pracovníka** – špatným naplánováním a hlavně přidělením zdrojů k projektu může dojít k přetížení pracovníka, který tak není schopen zvládat všechny role a dostát požadavkům, které jsou na něj kladeny. Jedním z možných řešení tak je zapojit do těchto

úkolů dalšího pracovníka, který je schopen realizovat alespoň část z těchto úkolů. Situace je o to složitější, jedná-li se o tzv. kritické úkoly, na jejichž dokončení závisí několik dalších a jež mohou nejvíce negativně ovlivnit harmonogram projektu

- **Špatný návrh** – je také možné, že během návrhu se zapomnělo na některé klíčové aspekty, které mohou ovlivnit průběh projektu, nebo např. jedné z jeho etap. Je možné, že odhadovaný čas a náklady na jeho realizaci se mohou lišit, avšak to jak negativně tak i pozitivně, samozřejmě že pro vývoj projektu je důležité odstranit hlavně vlivy negativní. Proto dojde-li k zjištění, že některá část projektu je náročnější než se ze začátku zdála. Je nutné opět reagovat a přizpůsobit plán projektu.
- **Nízké nasazení pracovníka nebo jeho špatná kvalifikace** – v průběhu projektu se může ukázat, že některý z členů týmu nebyl na svou pozici vybrán dobře. Může se také stát, že své úkoly neplní dobře a přistupuje k projektu lehkovážně. V tomto případě je opět nutný zásah projektového manažera, který musí sjednat nápravu, v některých případech až nahrazením pracovníka.
- **A další** …

Vidíme, že během implementace může nastat spousta problémů. Je však důležité si uvědomit, že spousta z nich má příčinu již v některé z předchozích fází. Je také nutností, aby při analýze a návrhu došlo k odstranění většiny nejasností a možných problémů, jelikož jejich následné odstraňování v průběhu samotné implementace je vždy mnohem náročnější a to jak z hlediska časového tak z hlediska nákladů potřebných na provedení a zapracování úprav. Nebývá však výjimkou, že špatná analýza a návrh mohou zavést tým při implementaci do slepé uličky a tak i k možnému selhání projektu. Kvalita samotné implementace se tedy velmi odvíjí od kvality návrhu.

1.1.4 Kontrola a testování

Máme-li dokončenu implementaci samotného projektu nebo jedné z jeho etap, můžeme přikročit k další fázi, která se nazývá testování. Cílem testování je kontrola právě vytvořeného díla. Vždy je nutné otestovat jeho správnou funkčnost předtím, než bude použit pro vývoj v dalších etapách. Předchází se tak problémům, které se mohou projevit až velmi pozdě, kdy při vývoji v posledních etapách je najednou zjištěno, že její výsledky nebudou kompatibilní s komponentou, která je ve skutečnosti jádrem systému v důsledku jejího nedostatečného otestování. Takový přístup může projektového manažera a celý tým dovést do obrovských problémů, jelikož bude potřeba projekt přepracovat, což je vždy spojeno s dodatečnou prací a náklady. Provádění testů je tedy velmi důležitou součástí projektu. Je potřeba této fázi přikládat dostatečnou pozornost a také si na ni vyčlenit potřebné prostředky a čas. Je chybou projektového manažera, pokud se pod nátlakem, například z důvodů úspory času nechá dotlačit do situace, kdy razantně snižuje množství času určené pro testování, nebo dokonce tuto fázi úplně vynechává. S vidinou aktuální úspory času se však chová velmi krátkozrace, jelikož ušetřený čas může vést k problémům, které mohou znamenat významné zvýšení nákladů později během projektu.

1.1.5 Ukončení projektu

Po dokončení implementace a následném testování může dojít k tomu, že je potřeba doladit některé detaily. V takovém případě se projekt či jeho etapa dostává opět do fáze návrhu a proces probíhá ještě jednou. Jsou-li však splněny veškeré požadavky, které byly na výsledek projektu kladeny splněny, tak jak bylo definováno v analytické fázi je možné projekt úspěšně ukončit a následně předat výsledek zákazníkovi.

V souvislosti s ukončováním projektů je třeba zmínit, že zdaleka ne všechny projekty končí úspěchem. Neúspěch je ve velké většině způsoben právě selháním projektového managementu a to na jakékoliv z jeho úrovní.

Další skupinou projektů jsou právě ty, které jsou sice úspěšně dokončeny, ale na úkor překročení nákladů, které byly v úvodu stanoveny, nebo zpožděním jeho dodávky. Obojí pro cílového klienta zpravidla znamená finanční ztráty a tak snižuje hodnotu výsledku projektu. Je však věcí projektových manažerů, aby dostatečně kvalitním analyzováním, plánováním, kontrolou a dostatečným testováním svých produktů těmto problémům zabránili, nebo je alespoň s dostatečných časovým předstihem odhalovali a svým jednáním následně eliminovali.

1.2 Oblasti projektového managementu

Jak jsme již zmínili dříve, projektový management zasahuje do spousty oblastí a snaží se o komplexní pohled na vývoj projektu. Následuje popis jeho jednotlivých oblastí.

1.2.1 Řízení času

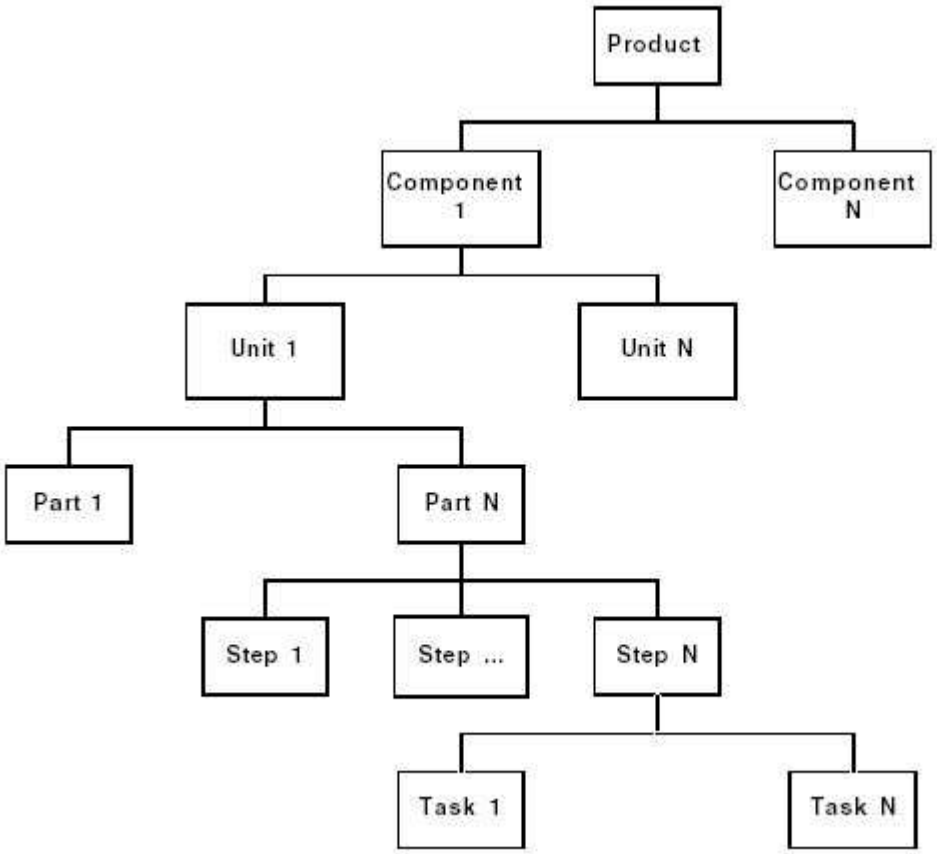
Řízením času v rámci projektu se jeho manažer snaží o správné naplánování projektu a jeho řízením zabezpečit dodržování časového plánu, tak aby nedošlo ke zpožďování. Základním východiskem pro plánování času je seznam činností, které je třeba během projektu udělat.

Pro plánování projektu je velmi důležité sestavení tzv. WBS diagramu (= work breakdown structure diagram). Většinou je reprezentován jako víceúrovňový hierarchický strom. Nejvyšším uzlem tohoto stromu je projekt jako celek. V první úrovni se strom dělí na fáze projektu, tedy je reprezentován fakt, že projekt se skládá z několika fází. Ty se pak dále dělí a to lze provádět až do několika úrovní tak, aby na nejnižší úrovni byly pouze základní činnosti. Všimněte si, že WBS vyjadřuje pouze to, které činnosti musejí být v rámci projektu udělány, neříká nám kdy a hlavně v jakém pořadí (Chapter Developing the work breakdown structure, [3]).

Pro popis a vytvoření struktury projektu se používá více druhů WBS diagramů a to:

- **Product WBS (produktový WBS)** – Je založený na rozdělení vývoje produktu na jednotlivé komponenty. Na nejvyšší úrovni se tedy nachází vytvořený produkt, na druhé úrovni jednotlivé komponenty ze kterých se skládá, ten se skládá z určitých částí, jejichž vývoj může být rozdělen na jednotlivé činnosti (úkoly).

- **Project WBS (projektový WBS)** – Je základním WBS modelem, který dělí projekt na jednotlivé fáze, následně na kroky a činnosti.
- **Hybrid WBS (hybridní WBS)** – Jedná se o kombinaci dvou výše uvedených modelů. Z hlediska dnešních moderních metodologií může tento model sloužit nejlépe pro vývoj v IT založený na vývoji komponent. Je možné výsledný produkt rozdělit na jednotlivé komponenty, což je velmi používaná technika a jejich vývoj provádět samostatně a to v několika fázích, krocích a na nejnižší úrovni činnostech (úkolech).



Hybridní WBS 1.2:1 (Chapter Developing schedule and work plan, [3])

Základním modelem pro plánování časového průběhu projektu je však tzv. síťový diagram. Ten obsahuje všechny aktivity a zobrazuje jejich závislost. Vychází tak z WBS diagramu, ale znázorňuje také závislost jednotlivých činností. Typů závislostí mezi činnostmi může být několik typů, nejdůležitější je pro nás typ *finish-to-start*, která nám říká, že následující činnost může být započata až po dokončení předchozí.

Máme-li sestaven síťový diagram činností je také potřeba určit jak dlouho bude jaká činnost trvat. Odhad trvání některých činností může být náročný a vyžaduje velké zkušenosti projektového managera a jeho konzultace s členy týmu.

Síťový diagram je základní pomůckou pro stanovení harmonogramu a zjištění možného data dokončení projektu. Budeme-li znát návaznost jednotlivých činností a délku jejich trvání, můžeme v grafu nalézt tzv. kritickou cestu. Jedná se o nejdelší možnou cestu grafem a vyjadřuje minimální možnou dobu trvání projektu. Činnosti, které se nacházejí na této cestě se nazývají kritickými

činnostmi a právě zpožďováním kritických činností může docházet k následnému prodlužování projektu. Ostatní činnosti musejí být samozřejmě také dobře naplánovány. Pokud je však možné je provádět paralelně ke kritickým činnostem, vzniká pro ně jistá časová rezerva, která je dána právě delší dobou provádění činností na kritické cestě. Ale pozor, opožděním některé z činností mimo kritickou cestu může dojít k takovému zpoždění, že se tato činnost stane součástí kritické cesty a také může ovlivnit délku trvání projektu.

1.2.2 Řízení nákladů

Cílem řízení nákladů je správné naplánování projektu z finančního hlediska. Skládá se z:

- **Plánování zdrojů** – Nejprve je potřeba určit, které zdroje jsou pro projekt potřeba. Je nutné určit která zařízení bude třeba použít a kteří lidé se musejí účastnit projektu a zapojit se do jeho tvorby.
- **Oceňování** – Jednotlivé zdroje je potřeba naplánovat tak, aby bylo možné provést jednotlivé činnosti a zjistit jejich předpokládanou cenu.
- **Rozpočtování** – Jakmile máme oceněny jednotlivé činnosti je třeba sestavit komplexní rozpočet projektu. Celkové prostředky je potřeba dobře rozdělit a správně naplánovat jejich využití.
- **Kontrola rozpočtu** – Po naplánování je potřeba kontrolovat správný průběh projektu a kontrolovat jeho změny.

1.2.3 Řízení kvality

Než začneme s řízením kvality v rámci projektu, je potřeba si vysvětlit co pojem kvalita znamená. Jedním z pohledů může být ten, že výsledná implementace projektu je natolik kvalitní, nakolik výsledek souhlasí se zadáním. Tento pohled však není zcela správný, jelikož je nutné si uvědomit, že chyba může vznikat již při specifikaci výsledku. V tomto případě můžeme výsledek považovat za kvalitní i když je vlastně pro koncového uživatele nepoužitelný.

Lépe pojem kvalita vyjádříme, podíváme-li se na ni komplexněji. Může ji ovlivnit mnoho faktorů, jako například:

- Splnění specifikace produktu
- Míra uspokojení potřeb zákazníka
- Použitelnost produktu
- Splnění průmyslových standardů
- Testovatelnost
- Rozšiřitelnost
- Bezpečnost
- Flexibilita

- Spolehlivost
- A další …

Vidíme, že faktorů ovlivňujících výslednou kvalitu projektu je mnoho. Je tedy potřeba plánovat a průběžně kontrolovat kvalitu projektu, tak aby nedošlo k jeho znehodnocování.

Na základě mnoha zkušeností lze obecně říci, že nejlepším způsobem pro zvýšení kvality práce je zkvalitnění jednotlivých procesů, které probíhají při jeho vývoji. Jedná se například o zavádění platných standardů, zavádění úspěšných metodik vývoje nebo o zlepšování technologie. Tato řešení jsou obecně lepší, než provádění zpětných kontrol již hotového produktu. Z výše uvedeného je tedy patrné, že lepšími opatřeními pro zvyšování kvality jsou právě ta preventivní. Může však být někdy problémem, že zákazníkovi může být přínos těchto opatření skryt na rozdíl od lépe znatelných zásahů při koncových opravách projektu. V té situaci totiž zákazník pocítí chybu systému a její následné odstranění. Naopak preventivní opatření se snaží těmto stavům předejít.

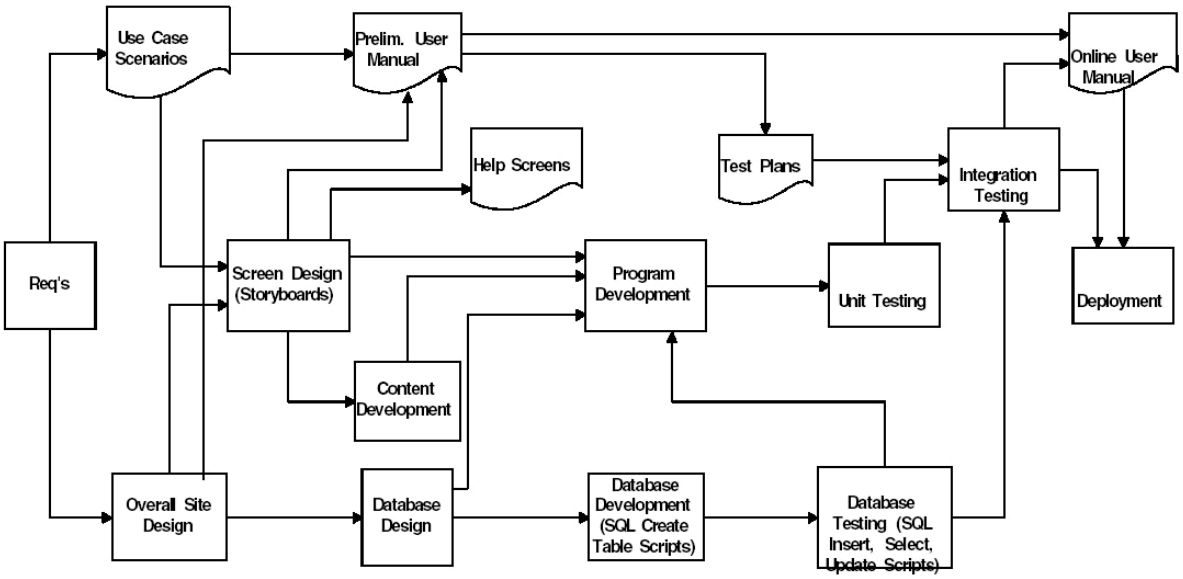


Schéma vývoje webových stránek 1.2:1

Plánování kvality je do velké míry propojeno s problematikou testování. Nedostatečným testováním v rámci jednotlivých etap projektu může docházek k zavádění chyb do výsledku projektu, což vede ke snižování jeho kvality. Na obrázku 1.2:1 vidíme například schéma vývoje webových stránek. Dopustíme-li se v tomto procesu chyby v definici požadavků na systém, na kterou přijdeme až při konečném testování projektu, bude nutné velkých zásahů a to téměř do celého projektu tak, abychom problém mohli odstranit, jelikož mezitím proběhla spousta činností, které jsou závyslé právě na definici požadavků. Zkvalitněním kontroly již během jednotlivých etap vývoje však můžeme zlepšit jeho kvalitu.

Zde předkládám ještě několik faktů o chybách v projektech, dle světových odborníků (:

- Chyby v projektech můžou být rozděleny podle jejich příčin do několika skupin: chyba požadavků, návrhu, kódování, dokumentace. Odstraňování jejich následků v pozdějších fázích projektů je zpravidla 40 – 1000x časově náročnější než prevence, která jim může předejít (Gause & Weinberg, 1989).
- Vyhledávání a oprava chyb, může projektu vzít až 80% jeho času (McConnell, 1997).
- Určité procento chyb (asi 22%) je nalezeno až ve chvíli, kdy je projekt zaběhnut a používán.
- Až 55% chyb vzniká při fázi analýzy požadavků a definici projektu.
- Pro systémy, které nevyužívají objektově orientovaný přístup k vývoji, který se snaží zapouzdřovat jednotlivé části řešení do menších částí, může odstraněním jedné chyby až ve 40% případů stát, že v důsledku vysoké provázanosti částí kódu vznikne chyba nová.

Pro vylepšení procesu vývoje lze tedy doporučit například následující pravidla:

- Zavádění již ověřených a otestovaných postupů – máme-li k dispozici již jednou vyvinutý produkt nebo jeho komponentu, která již funguje, použijme ji i v nově vyvíjeném projektu
- Průběžná validace mezivýsledků projektu – průběžně během projektu provádějme kontrolu, zdali jdeme správným směrem

Samozřejmě metod jak obecně zvyšovat kvalitu projektu je mnoho a lze říci, že zasahují do všech ostatních oblastí projektového managementu.

1.2.4 Řízení lidských zdrojů

Řízení lidských zdrojů je velmi komplexní činností, která vyžaduje velké úsilí projektových manažerů. Ve větších firmách dokonce dochází ke vzniku celých firemních oddělení, která se zabývají touto problematikou.

1.2.5 Řízení komunikace

Správná komunikace celého týmu je nezbytným předpokladem pro jeho správné fungování. Bez dobré komunikace může docházet k vzniku zbytečných problémů či dokonce vzniku krizových situací. Je povinností projektového manažera vytvořit správný komunikační plán, který zpravidla definuje jakými kanály mají členové týmu předávat informace. Jeho dalším úkolem je řešit možné problémy v rámci týmu, pro které je právě správná komunikace, která hledá řešení, jedinným možným východiskem.

1.2.6 Řízení rizik projektu

Chce-li člověk v dnešním obchodním prostředí uspět, musí počítat s tím, že na sebe bude muset vzít nějaká rizika. Měl by tak však dělat uvážlivě a do co největší možné míry se naučit rizika řídit. Jediným způsobem jak se naučit řídit rizika projektu je naučit se je identifikovat, dát jim měřitelná kritéria a umět vytvořit plán, který jim předejde. V dnešní době je právě podcenění analýzy rizik, zejména ve velkých IT projektech, jedním z nejčastějších činitelů, které způsobí selhání projektu.

V moderně pojatém projektovém managementu se však právě analýza rizik projektu stává jedním z nejdůležitějších úkolů projektového manažera.

Vždy je tak nutné sestavit seznam všech možných rizik, identifikovat, které z nich mohou být pro projekt skutečnou hrozbou, a právě na tyto se zaměřit a vytvořit plán pro situaci, že by se toto riziko skutečně vyplnilo. V neposlední řadě je pak během projektu neustále kontrolovat, jestli nenastala ona krizová situace, abychom mohli spustit případný potřebný krizový plán.

Z některých studií dokonce vyplívá, že analýzou a řízením rizik je možné jejich nebezpečnost snížit až o 90%.

Jednou s technik jak analyzovat možná rizika je tzv. SWOT analýza (strenghts, weaknesses, oportunities, threats). Dalším možným způsobem jak odhalit rizika je například analýza bezpečnosti. Identifikace rizik by měla proběhnout na základně analýz, zkušeností a historických souvislostí, informací z oblasti vyvíjeného projektu (například informace o vývoji na trhu) apod. Zdroje rizik by se daly rozdělit do několika kategorií:

- **Interní** – jedná se o rizika, která vznikají uvnitř projektového týmu, například indispozice některého z klíčových pracovníků.
- **Technická** – příčiny, které vznikají z důvodů techniky, například možnost změny technologie
- **Externí** – faktory které vznikají mimo tým, například možnost konkurenčního boje o cenu
- **Nepředpovídatelná** – cca 10% rizik

Jakmile máme identifikována jednotlivá rizika je potřeba zjistit jejich nebezpečnost a pravděpodobnost. Jedná se o komplikovaný proces, jelikož nejsou jasná a přesná pravidla. Velká míra se tak přikládá názorům odborníků, historickým souvislostem a zejména zkušenostem projektových manažerů.

Shrnutí

Projektový management je dnes nezbytnou součástí tvorby kvalitního softwaru. Využitím nejmodernějších technik projektového managementu se společnosti snaží docílit zásadní konkurenční výhody, která bude znamenat, že jejich software bude kvalitnější a lepší. Jednou z nejnovějších a nejvíce se rozvíjejících metodik je *Rational unified process* (RUP) o které se dozvíte více v další kapitole.

2 RUP – Rational unified process

RUP vzniká jako reakce na zvyšování komplexnosti softwarových projektů na konci 20. století, kdy selhávají tradiční metodiky. Snaží se projektovým manažerům poskytnout obecně uznávané postupy jak vylepšit řízení projektů vývoje softwarových systémů.

2.1 Historie

Prvopočátky RUP se pojí se společností Rational Software Corporation a 80. léty 20. století. Za jeho zakladatele jsou považování Paul Levy a Mike Devlin, kteří jej vyvíjeli jako novou metodiku pro vývoj rozsáhlých a komplexních projektů v oblasti IT. V této době byly však rozsáhlé projekty spojeny zejména s oblastí vládních projektů.

V průběhu dalších let docházelo k migraci na komplexnější programovací jazyky, jakým je např. C++, k masivnímu rozšíření personálních počítačů, či rozšíření mikroprocesorů do mnoha dalších systémů. Společně s rozvojem internetové sítě tyto problémy zvětšovaly náročnost vyvíjených projektů a samozřejmě přímo úměrně náročnost jejich kvalitního řízení. V této době změnila společnost Rational svou taktiku při vývoji RUP, strategie však zůstává stejná a to vytvořit pomůcku pro vývoj velkých a komplexních projektů.

V následujících letech začaly vznikat různé modelovací nástroje usnadňující návrh projektů. Trh byl však velmi roztříštěný. Ve společnosti Rational vznikla touha vyvinout jednotný modelovací jazyk, kterým se nakonec stal jazyk UML. Společně s dokumenty, které shromažďovaly informace o nejlepších metodách určených k vývoji softwarových projektů daly vzniknout již dnes známému RUP.

V roce 2003 byla společnost Rational Software zakoupena společností IBM, která nadále pracuje na vylepšování principů RUP.

2.2 Základní principy

Základním kamenem RUP je 6 postupů, které si nyní vysvětlíme.

Iterační vývoj - vyvíjet projekt v jednotlivých iteracích umožňuje v co nejkratší době vyvinout alespoň částečně fungující verzi konečného produktu, která plní určitou funkci a je tak možné ji představit i zákazníkovi. Před každou iterací je důležité zvolit její správnou délku tak, aby se skutečně podařilo vyvinout použitelný celek a také správně a exaktně definovat její cíl.

Management požadavků - Pro správné pochopení produktu, který má být v rámci projektu vytvořen je důležité správně identifikovat požadavky. Jen na základě jejich správného pochopení a zpracování může tým postavit kvalitní produkt, který je splňuje.

Používaní architektury založené na komponentách - Architektura založená na komponentách využívá k vývoji produktu jednoduchého principu. Vývoj složitého a komplexního produktu rozděluje na vývoj několika menších celků, které se nazývají komponenty. Je však předpokládána správná dekompozice systému, který se následně skládá z komponent, kde každá plní určitou funkci a je sama o sobě funkční jednotkou systému. Zpravidla je vývoj systému založeného na komponentách rozdělen tak, aby v jednotlivých iteracích docházelo k vývoji právě jedné z klíčových komponent systému, která je pak začleněna do projektu.

Vizuální modelování - již od dob vzniku modelovacích nástrojů je považována tato technika za kvalitní nástroj pro usnadnění návrhu komplexních systémů. Jejich rozsah může být natolik složitý, že bez správného vizuálního modelu, nejčastěji založeného na UML diagramech, může docházet právě ke špatné dekompozici systému a následným problémům s jeho implementací či např. k problémům s konzistencí jednotlivých komponent.

Průběžná kontrola kvality - Jak jsme si již uvedli v první kapitole, průběžná kontrola kvality vyvíjeného projektu je nejjednodušší technikou jak se vyhnout velkým problémům, které se mohou během projektu nakumulovat. Validací a verifikováním výsledků jednotlivých etap ve vývoji či validací jednotlivých komponent, můžeme zabránit následným problémům, které vznikají později a jejichž odstranění může být velkým problémem.

Management změn - Iterační vývoj dává oproti původně používaným modelům, například model vodopádovému, projektovému týmu výhodu flexibilnějšího procesu při vývoji softwarového produktu. Jelikož je projekt vyvíjen v několika cyklech máme lepší možnost reagovat na průběžné

změny a s jejich začleněním do projektu. Je však nutné řídit systém změn a pečlivě a svědomitě je shromažďovat a zdokumentovat ještě před započetím další iterace vývoje.

Uvedli jsme si stručný přehled principů RUP. Vývojáři však dále pokračovali na lepší specifikaci a přepracováním z října 2005 došly k následujícím konkrétnějším pravidlům, která si probereme jedno po druhé níže.

2.2.1 Přizpůsobení procesu vývoje

Jelikož každý projekt je jiný je potřeba mít možnosti změny procesu vývoje šitou na míru jednotlivým projektům. Pro menší projekty tak mohou být dány větší možnosti kreativitě jednotlivých členů s menším důrazem na byrokracii, která by v těchto případech mohla být zbytečná a omezovat členy týmu bez většího důvodu. Naopak u větších projektů, které mají zpravidla mnohem složitější strukturu je potřeba větší byrokracie a dodržování komplexnějších pravidel. Zpravidla je nutné do rozsáhlejších projektů zavést větší stupeň kontroly. Zejména i z důvodu delší doby potřebné k vývoji projektu je potřeba pečlivěji kontrolovat jeho průběh a provádět nutné změny.

Také počáteční fáze projektu dávají větší prostor pro kreativitu jednotlivých členů týmu. S postupem projektu je však nutné pečlivě dokumentovat všechny změny a dávat tomuto systému určitý řád. Je tedy na týmu a jeho manažerovi jakou míru kontroly zvolí, je však nutné aby byla adekvátní danému projektu.

Nakonec uvádím některé z možných vlivů, které mohou zvětšovat důležitost kontroly v rámci projektu:

- Projektový tým je geograficky rozdělen
- Komunita uživatelů je velká a geograficky rozdělená
- Projektový tým je sestaven z více firemních týmu, kde každý vyvíjí určitou část celku
- Projekt musí dodržovat určité standardy které je potřeba správně verifikovat
- Produkt, který je vyvíjen je technicky složitý
- Projekt se blíží do koncové fáze

2.2.2 Zpracovat požadavky zákazníka a přiřadit jim prioritu

Pro mnoho projektových týmů je standardním postupem shromáždění požadavků zákazníka a jejich následná implementace do vytvářeného systému. Tento způsob však nemusí být zcela správný. RUP zavádí lepší přístup ke správě požadavků.

Pro správné uspokojení potřeb zákazníka tak, aby systém skutečně splňoval veškeré požadavky, které jsou na něj kladeny avšak za nejlepší cenu a co nejdříve. Je nezbytné, aby projektový tým pochopil obchodní prostředí, do kterého má být systém zasazen. Jeho osvojení může pomoci rozlišit nejdůležitější požadavky, které systém musí opravdu plně uspokojovat od naopak těch

méně podstatných, které mohou být řešeny jednodušší cestou než by se mohlo z počátku zdát. Je tak možné určitým způsobem optimalizovat vyvíjený produkt a nabídnout zákazníkovi hodnotnější výsledek, který uspokojuje všechny potřeby na něj kladené avšak za kratší čas či menší cenu. Pochopení správného fungování výsledného systému na základě poznání prostředí ve kterém bude pracovat je vždy pomůckou, která nedovolí týmu, aby se při vývoji odchýlil a vyvíjel systém nesprávným směrem k následné nespokojenosti zákazníka.

2.2.3 Vytvořit kvalitní prostředí pro spolupráci

Vytvoření správného prostředí pro spolupráci v rámci týmu je mnohem složitější než řízení samotné komunikace. Je nutno vytvořit prostředí, ve kterém mají jednotliví členové týmu chuť sdílet rizika a odpovědnost s ostatními členy týmu a chtějí aktivně pracovat a také pomáhat s prací svým kolegům. Tvorba týmu, který bude na projektu pracovat je tak velmi složitá a právě výběr jednotlivých pracovníků je v tomto klíčový. Není však v možnostech této publikace zabývat se podrobnostmi.

2.2.4 Vytvářet produkt v jednotlivých iteracích

Základním kamenem vývoje projektu pomocí RUP je správnost iteračního vývoje, proto mu bude věnována celá kapitola. Nyní si jen shrňme důležitá fakta, která je dobré o iteračním vývoji znát:

- Je jednodušší vyvinout složitý projekt když jej můžeme složit s několika menších částí. Jeho rozdělení nám tak dává možnost nejen vytvářet větší celky, ale také lépe se zaměřit na detaily v rámci jeho jednotlivých částí.
- Máme lepší možnosti pro odhalení chyb. Iterační vývoj přináší možnost najít chyby rychleji, což nám umožní jejich rychlejší a tím i levnější odstranění.
- Výsledkem každé iterace by měl být testovatelný produkt, který je možno demonstrovat. Samozřejmě bude splňovat pouze část požadavků, kladených na celý systém, ale je možné jej ukázat uživatelům a průběžně tak ověřovat, že je projekt vyvíjen správným směrem.
- U složitých systémů může být pochopení celkového systému natolik složité, že jeho rozdělení na jednotlivé funkční celky je jediným řešením jak jej správně vytvořit.

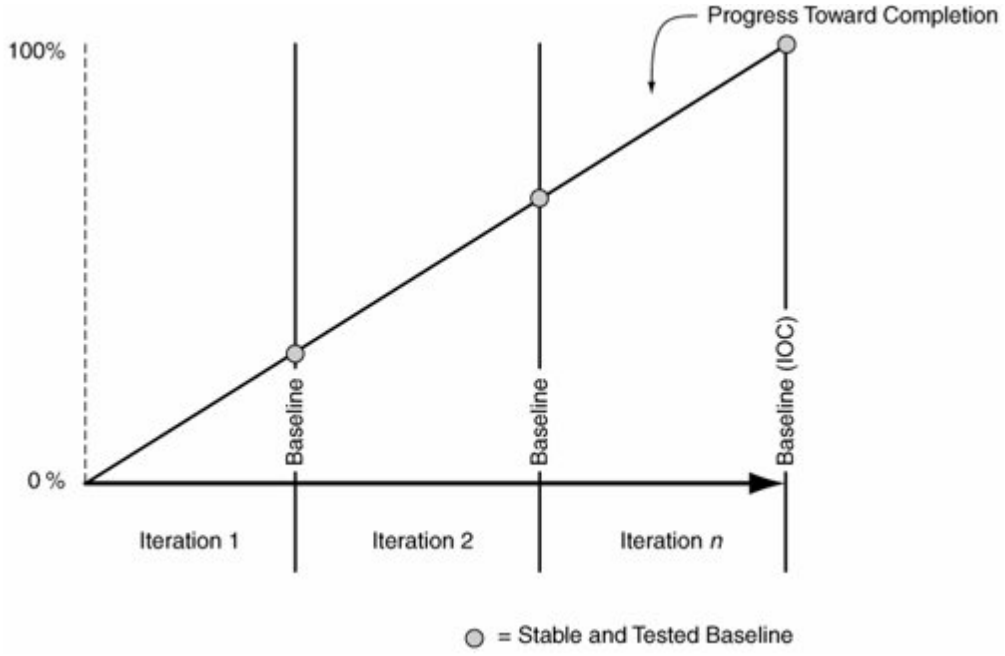


Schéma iteračního vývoje 2.2.4:1 (Chapter 2 Overview of Rational unified process, [2])

Iterační vývoj přináší několik výhod jako je například větší flexibilita při zavádění změn, jelikož zákazník vidí výsledek každé iterace a může tak aktivně zasahovat do vývoje a směrovat projekt správným směrem. Navíc dostane-li se projekt do zpoždění, bývá k dispozici v čase plánovaného spuštění verze, která splňuje alespoň většinu požadavků na systém (bez některých funkcí) a může tak být dočasně nasazena.

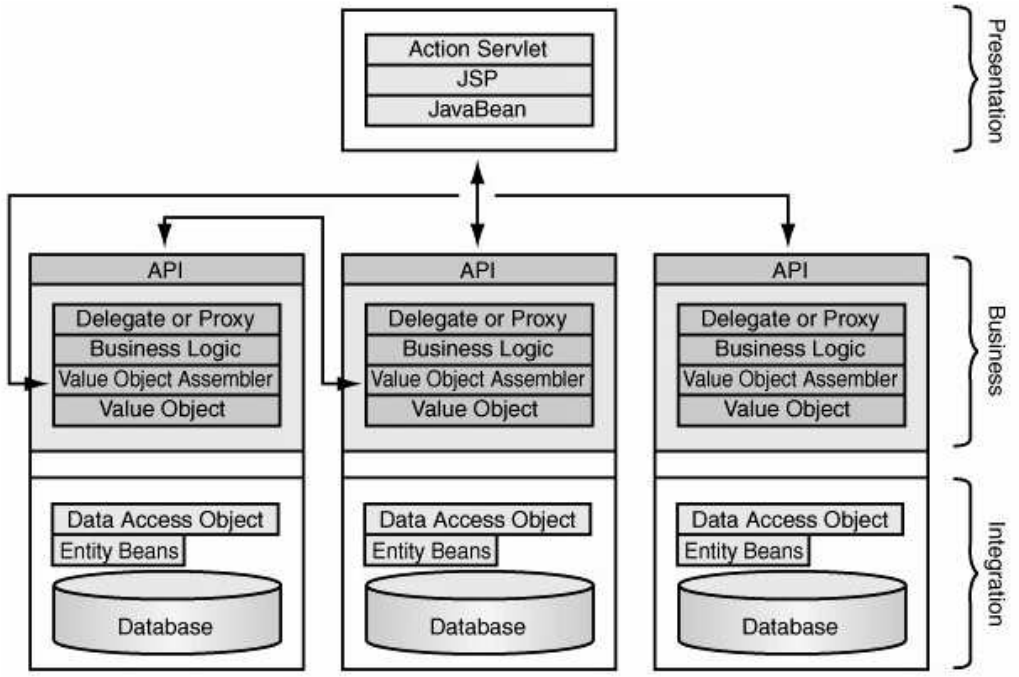
2.2.5 Zvětšit úroveň abstrakce

Dnešní projekty v oblasti IT nabývají takové složitosti, že i pro odborníky z oblasti projektového managementu a architektury informačních systémů je návrh systému jako jednoho celku bez použití abstrakce nezvladatelný. Pojďme si vysvětlit jaké techniky nám RUP nabízí, abychom tento problém mohli vyřešit.

Jak již jsme uvedli, jedním ze základních principů RUP je vývoj založený na komponentách. Systém je rozdělen na jednotlivé komponenty, kde každá plní určitou funkci. Komponenta má definované své rozhraní, což je sada funkcí, které mohou využívat okolní objekty (systém nebo další komponenty). Tímto dochází k abstrakci. Komponenty jdou však dělit na další menší celky. Jsou určitou kolekcí tříd, které mají vzájemnou souvislost a plní společný úkol. Třída je tak základním kamenem abstrakce v systému. Shromažďuje určitou sadu funkcí a vlastností a reprezentuje objekt, který má určitý stav a určitou funkci. Sama o sobě je taky uzavřená vůči ostatním třídám ve svém okolí a opět poskytuje pouze veřejné rozhraní funkcí, které mohou využít ostatní třídy.

Princip abstrakce nám tedy dovoluje dekompozici velkého funkčního celku na jednotlivé objekty, kde každý má svou funkci. Všimněme si na obrázku, že systém lze taky rozdělit do jednotlivých vrstev, kde každá vrstva se skládá z určitých částí a plní určitou funkci. Nejpoužívanější

architekturou dnešních vyvíjených systémů je 3 vrstvová architektura, která se skládá z prezenční vrstvy (ta se stará o zobrazení informací a o celkovou interakci s uživatelem), bussiness vrstvy (ta transformuje data z databáze a uplatňuje určitá pravidla, která jsou specifická pro každou organizaci) a databázové vrstvy (která v sobě zapouzdřuje práci s databází a samotná data). Jednotlivé vrstvy tak plní každá určitou funkci a tak opět zjednodušují práci při implementaci.



Dělení systému na uzavřené komponenty 2.2.5:1 (Chapter 2 Overview of Rational unified process, [2])

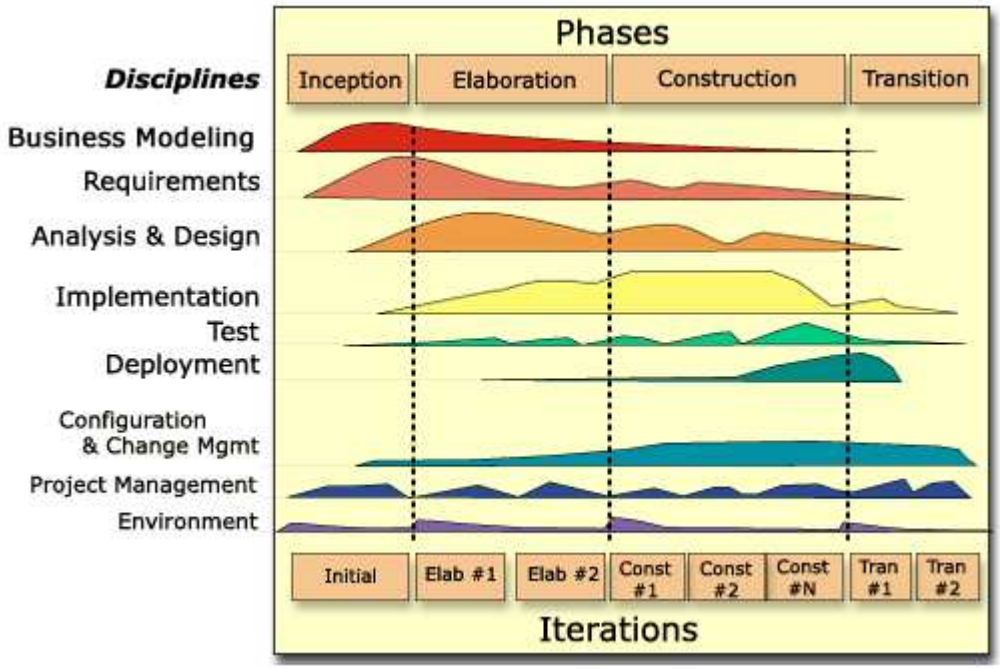
Abstrakce je také jedním z principů jak zvyšovat kvalitu vyvíjených produktů. Umožňuje totiž vznik komponent či celých vrstev v systému, které je možno testovat a poté úspěšně používat ve více projektech a tím zabránit zanášení nových chyb do systému. Jedná se o velmi silný nástroj, který je založen na principech objektově-orientovaného paradigmatu. Pro detailnější informace Vás odkazuji na některou z publikací, která se tímto tématem zabývá.

2.2.6 Zaměřit se na dosažení vysoké úrovně kvality

Již samotné dodržování předchozích principů velmi zvyšuje kvalitu vyvíjených produktů. Všechny výše uvedené techniky mají za úkol zlepšením návrh či lepším způsobem kontroly snižovat možnost vzniku chyb. O oblasti testování, která velkou měrou přispívá ke zvyšování kvality a je nezbytnou součástí RUP si povíme v kapitole 2.6 Testování.

2.3 Fáze RUP projektu

Projekt, který probíhá podle RUP lze rozdělit do několika fází jak ukazuje obrázek. Na něm také vidíme činnosti, které jsou v jednotlivých činnostech prováděny. Přehledné rozložení jednotlivých prací podle typu do iterací vidíme na obrázku 2.3:1.



Rozložení prací do fází projektu 2.3:1 (Chapter 4 Disciplines ans workflows [4])

2.3.1 Fáze zahájení

V zahajovací fázi je hlavním úkolem správné naplánování iterací projektu. Je potřeba definovat cíle jednotlivých iterací a navrhnout jejich správné parametry, které jsou vhodné pro daný projekt. Více o návrhu iterací se dozvíme v kapitole 2.5 Tvorba iterací. Po základní definici projektu je potřeba provést následující úkoly:

- Zjistit zda zákazník souhlasí s definicí klíčových aspektů projektu. Je nutné si uvědomit, že v této fázi nemáme k dispozici kompletní analýzu projektu. Musíme však mít představu o systému alespoň na nejvyšší úrovni. Musíme vědět komu bude systém sloužit, kolik jej asi bude obsluhovat lidí, jaká bude základní architektura, apod.
- Zjistit, zdali zákazník souhlasí s harmonogramem a rámcovým rozpočtem projektu.
- Identifikovat rizika projektu a seznámit s nimi zákazníka.

Tato fáze je zvlášť nutnou pro analýzu rizik, která musejí být identifikována, aby nemohla v pozdější fázi ohrozit projekt. Velmi důležitou součástí je naplánování iterací v rámci projektu, což může být náročným úkolem. Zpravidla však s narůstajícími zkušenostmi projektového manažera s podobnými projekty narůstá i kvalita tohoto návrhu. Je také možné, že již v této fázi budou objeveny některé nedostatky, je však ještě relativně jednoduché je odstranit.

2.3.2 Fáze zpracování

V této fázi projektu je již jasné jaké jsou na něj kladeny požadavky. V tuto chvíli se tým zaměřuje na vytvoření správné architektury vyvíjeného systému. Tým vychází z požadavků na systém a jejich důležitosti. Její cíle lze shrnout do několika bodů:

- Identifikovat správné cíle projektu, klíčové požadavky a vytvořit si představu o projektu jako celku
- Vytvořit správnou architekturu systému a navrhnout podobu jednotlivých verzí, které budou cílem jednotlivých iterací. Součástí je také vytvoření testovacích kritérií pro každou verzi.
- Ujistit se, že jsme identifikovaly všechny požadavky na systém. Je velmi pravděpodobné, že některé požadavky na systém mohou být stále skryty. Avšak většina (cca 80%) a hlavně nejdůležitější požadavky musejí být bezpodmínečně identifikovány.
- Harmonogram a rozpočet projektu musí být odsouhlaseny zákazníkem.
- Vytvořit rámcový plán všech iterací a detailní plán nejbližších iterací.

2.3.3 Konstrukční fáze

V této fázi již známe architekturu systému, známe jeho největší rizika a máme k dispozici plán iterací projektu. Nyní je cílem každé iterace vytvořit novou verzi systému, která je funkční, testovatelná a je možné předvedení její funkčnosti. Nová verze tak vzniká rozšířením verze předchozí o novou funkcionalitu. Testování, které je prováděno testovacím týmem na konci každé iterace má za úkol nalézt nedostatky a předat je vývojovému týmu, od kterého se předpokládá, že jejich opravu provede v některé z následujících iterací podle důležitosti.

Pro některé zvláště rozsáhlejší projekty může být užitečné, vytvořit na konci projektu jednu nebo dvě prázdné iterace, ve kterých není naplánováno žádné další rozšiřování funkcionality. Tyto iterace slouží jako časová rezerva a mohou být využity pro dodatečné opravy chyb, které mohou vznikat během projektu, či pro zabudování funkcionality, která nebyla na začátku projektu známa a naplánována.

2.3.4 Přechodná fáze

Jedná se o poslední fázi, ve které bude dokončena iterace. Hlavním úkolem je vytvořit některé dokumenty které s touto iterací souvisí, jako například některé části dokumentace, manuály, nápověda apod. Součástí může být také oprava některých chyb, které byly nalezeny testováním. Délka této fáze je silně závislá na povaze projektu. Její délka roste s náročností a počtem uživatelů systému.

2.4 Správný návrh projektu z pohledu RUP

Správný návrh projektu je jedním ze základních předpokladů, pro jeho úspěšné dokončení. Chyby v návrhu jsou těmi nejhoršími a mohou způsobit velké projekty při konstrukci projektu. Proto je nutné správně verifikovat navržený model, který by měl splňovat následující kritéria:

- Tým je si jist, že jím zvolená architektura splňuje všechny klíčové požadavky na systém, zejména z oblastí: bezpečnosti, rychlosti obsluhy, dostupnosti a rozšířitelnosti.
- Jejich jistota je prokazatelně ověřena verifikací navržené architektury.

2.5 Tvorba iterací

Správné naplánování iterací je dalším důležitým krokem pro realizaci projektu.

Jedním z důležitých kroků je naplánování délky iterací. Samozřejmě na to není univerzální pravidlo a iterace by měla být adekvátní náročnosti a rozsahu projektu. Jednotlivé iterace by měly být tak dlouhé, aby tým zvládl vyvinout další verzi. Například pro menší týmy 2 až 3 lidí může být délka iterace 1-2 týdny. Pro velké týmy 20 až 30ti lidí, které zpravidla pracují na velmi rozsáhlých projektech může být délka iterace až několik měsíců (Chapter 10 Staying on target [2]).

Dalším důležitým rozhodnutím je správné naplánování a rozdělení práce do jednotlivých iterací. Pro rozhodování, které požadavky implementovat v jednotlivých iteracích, mohou být důležité následující kritéria:

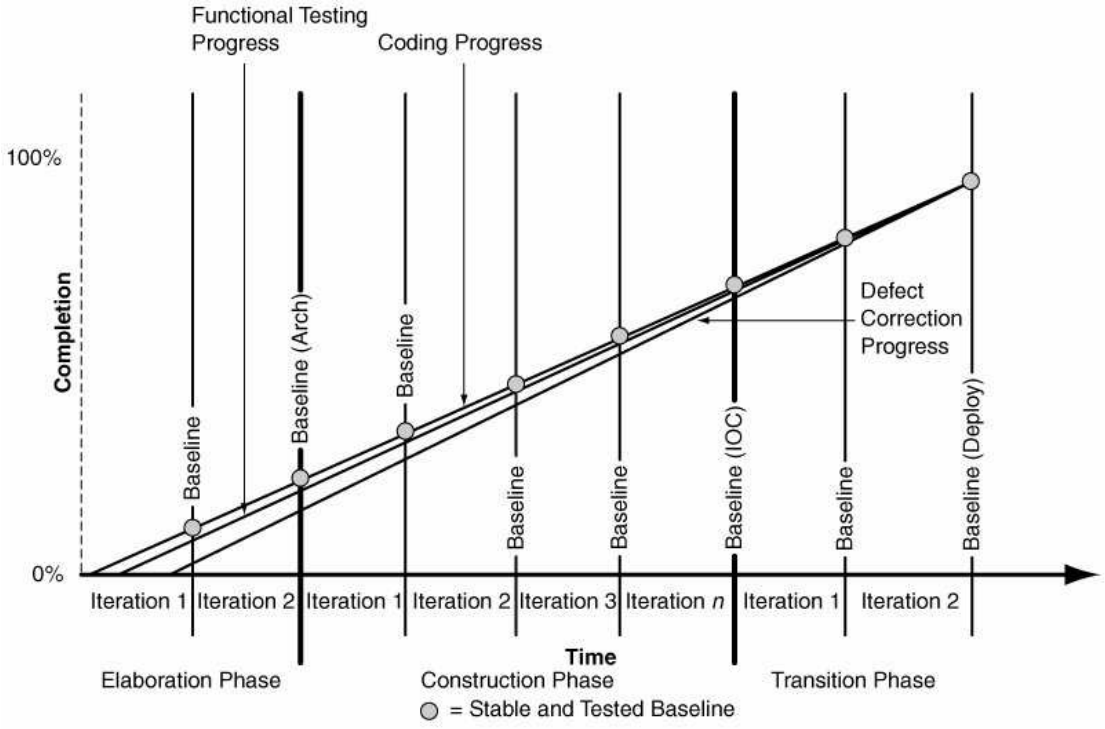
- Je dobré zařadit náročnější požadavky na začátek projektu do některé z prvních iterací. Tento přístup nám umožňuje co nejdříve implementovat nejrizikovější části projektu a co nejdříve eliminovat rizika. Jedná se o požadavky, které vyžadují složité algoritmy, se kterými nemá implementační tým zkušenosti nebo ještě nemá přesnou představu o jejich řešení. Projekt se tak vyhýbá odsunování nejrizikovějších části na závěr. Je však důležité si uvědomit, že implementací nejsložitějších požadavků na začátku projektu může být první verze nevhodná pro demonstraci. Je tak vhodné do prvních iterací zapojit i některé ze snadno demonstrovatelných a implementovatelných požadavků tak, aby bylo možno zákazníkovi lépe prezentovatelnou verzi produktu.
- Do následující iterace je třeba zařadit nedokončené požadavky z předchozích iterací.
- Do dřívějších iterací je třeba naplánovat vytvoření komponent, které budou využívány dalšími částmi systému tak, aby byla k dispozici před započetím vývoje těch částí systému, které tuto komponentu využívají.
- Do prvních iterací zařadit požadavky, které jsou zvláště důležité pro uživatele systému.

Máme-li rozdělenu práci do jednotlivých iterací nastává potřeba vytvořit detailní plán pro první iterace. Tato operace se opakuje vždy po dokončení jednotlivé iterace v rámci projektu. Je potřeba provést tyto kroky:

- Identifikovat hlavní cíle iterace. Pokud by docházelo ke zpožďování a nutnosti rozhodnout, které požadavky je potřeba dokončit, je možné některé z požadavků zařadit až do dalších iterací avšak vždy je nutné dokončit alespoň implementaci hlavních cílů.
- Přiřadit její provádění jednotlivým pracovníkům

2.6 Testování

Z uvedeného grafu je jasně patrné, že základem pro úspěch projektu je neustálé testování meziproduktů, které vznikají na konci jeho jednotlivých fází. Provádíme tak průběžnou kontrolu dosavadních výsledků projektu, což umožňuje rychlé odhalení možných chyb. Zejména nejzávažnější chyby jsou pomocí průběžného testování zpravidla odhlaleny při prvotních testech, například při ověřování správnosti navržené architektury. Na obrázku 2.6:1 vidíme optimální průběh testování. Je prováděno průběžně a tak také dochází k průběžnému odstraňování chyb. Nejvíce je jich nalezeno v počátečních fázích a s blížícím se koncem projektu jsou již téměř všechny odstraněny. Samozřejmě se může stát že některé chybu v projektu zůstanou, avšak díky průběžnému testování lze předpokládat že to budou spíše menší chyby vzniklé při implementaci, než nějaké závažné koncepční chyby.



Průběh testování během projektu 2.6:1 (Chapter 11 Testing, [2])

Shrnutí

Metodika RUP nabízí komplexní přístup k managementu projektů a její principy uplatníme zejména při vývoji náročných a rozsáhlých projektů.

3 Analýza systému

Analýza systému vyústila k definování základních dokumentů pro implementaci projektu. Jedná se zejména o definici požadavků a funkční specifikaci systému.

3.1 Definice požadavků

- Nezávyslost na platformě a operačním systému
- Jednotné grafické prostředí s autentifikací ve kterém zaměstnanci pracují
- Pracovníci mohou v systému zastávat různé role – manažer, pracovník
- Správa firem a kontaktů
- Správu zaměstnanců firmy
- Správa a plánování projektů
- Projekt se dělí do jednotlivých workflow, které mohou být hierarchicky zanořovány
- Projekt obsahuje činnosti, které jsou plánovány pro daného pracovníka a zařazovány do jednotlivých workflows
- Udržování informací o provedené práci ke každému úkolu, práci zadává pracovník, nebo manažer příslušného projektu (vždy datum +počet odpracovaných hodin)
- Správa stavu úkolů, jakmile byla na úkolu provedena práce, nelze již odstranit
- Při dokončení úkolu pracovníkem je dále schválen ještě projektovým manažerem
- Systém funguje přes standardní webové prohlížeče (Internet Explorer, Firefox, Opera)
- Možnost vkládat komentáře k jednotlivým úkolům (pracovník a manažer projektu ke kterému úkol náleží)
- Možnost vyhledávání úkolů podle kritérií – pracovník, projekt, stav, datum ukončení
- Možnost změny hesla do systému pracovníkem
- Statistiky ukazující průběh projektu
- Jednotlivé úkoly mohou mít své prerekvizity, u úkolu zobrazit také informace o stavu jeho prerekvizit
- Úkol nelze naplánovat dříve, než budou dokončeny jeho prerekvizity
- Jednotlivé workflows jsou seřazovány a plánovány do posloupností
- Workflow nelze naplánovat dříve, než bude dokončena předchozí etapa
- Plán projektu se přizpůsobuje činnostem, jsou-li do daného worfklow naplánovány činnosti, které svým rozsahem přesahují daný workflow, bude prodloužen a veškeré workflows a úkoly, které na něm závysí budou také automaticky přeplánovány
- Pracovník může prohlížet jen své úkoly a projekty ve kterých má některý ze svých úkolů, ostatní úkoly a projekty mu budou skryty
- Projektový manažer má v rámci svého projektu veškerá práva na úpravy úkolů, kromě mazání odvedené práce

3.2 Funkční specifikace

Obecně

- Před vstupem do systému je potřeba uživatele autentifikovat. Po zadání uživatelského jména a hesla je uživateli přidělen unikátní identifikátor (uložený v databázi) a na jeho počítač je zaslána autentifikační cookie.
- Při vstupu do systému uživatel vidí přehled svých aktivních úkolů, tj. těch, které jsou ve stavu který od pracovníka vyžaduje práci pro dokončení. Úkoly, které již byly dokončeny se nezorazují. Jednotlivé úkoly se řadí chronologicky podle data jejich dokončení. Njestarší úkoly jsou zobrazeny nejvýše. Úkoly, které ještě nejsou dokončeny přesto, že již měly být, jsou vyznačeny.
- Na úvodní obrazovce uživatel vidí aktuální komentáře, které jsou k jeho příslušným úkolům. Jsou seřazeny chronologicky dle data od nejnovějšího ke starším.
- Systém umožňuje projektovéme managerovi správu kontatků, firem a pracovníků. Pod pojmem správa se rozumí možnost vkládání, modifikace dat a mazání údajů. Součástí funkce systému je kontrola vzájemných vabez objektů, tak aby při mazání či úpravách nedocházelo k neintegritním stavům databáze, viz níže. Jedinnou vyjímku tvoří změna hesla pracovníka, tu smí provádět jen pracovník sám.
- Systém umožňuje vyhledávání projektů podle jednotlivých pracovníků, stavu a data ukončení. Výsledky je možné seřadit vzestupně i sestupně podle všech parametrů.

Plánování projektů

- Projektový manažer má při vstupu na detail projektu k dispozici přehled o aktuálně naplánovaném projektu. Vidí přehledně uspořádané workflows, které jsou také chronologicky seřazeny podle času od nejstaršího, při aktivaci jednoho z nich jsou mu zobrazeny také detaily o vybraném workflow a úkoly, které do něj patří.
- Projektový manager má možnost upravovat časové naplánování jednotlivých etap s tím, že určuje návaznost jednotlivých etap a úkolů, avšak systém se musí postarat o správnou kontrolu data naplánování tak, aby úkoly resp. Workflows, které jsou prerekvizitami byly naplánovány před úkoly resp. Workflows, které jsou jejich následníky. Přeplánování projektu dělá software automaticky i v případě že upravený harmonogram přesahuje rozsah projektu. Je-li provedena úprava, která je z hlediska plánování nemožná, uživatel bude upozorněn.
- Při úpravách projektu není možné mazat workflows, které obsahují úkoly, či dále úkoly, na kterých již byla odvedena práce. Samotné záznamy o práci také nemohou být odstraňovány.

3.3 Cíl projektu

Cílem projektu je navrhnout nástroj určený pro podporu projektového managementu v IT organizacích. Je potřeba se zejména zaměřit na plánování časových a finančních hledisek projektu. Pro tyto účely bude navržen komplexní informační systém, který implementuje požadavky, které byly definovány.

3.4 Shrnutí

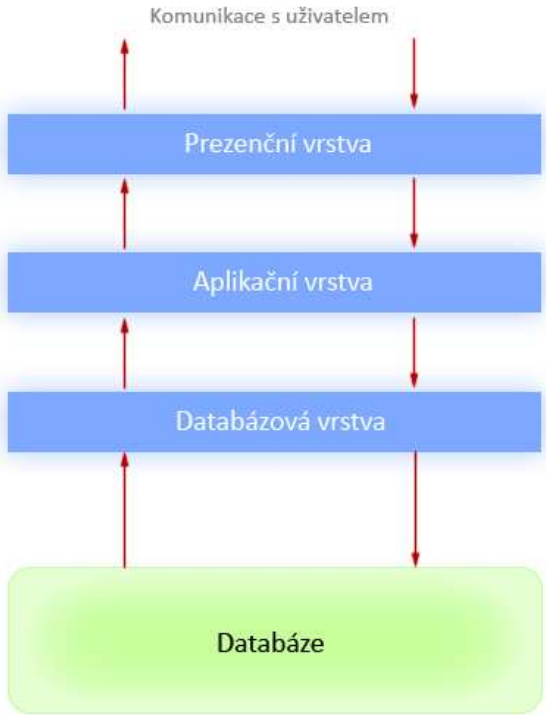
Analýza projektu vyústila v definici základních požadavků na systém a vytvoření jeho funkční specifikace. Tyto dokumenty se staly základem pro následující návrh systému.

4 Návrh

Vyvíjený systém bude implementován jako informační systém založený na webovém prohlížeči. Pomocí vhodně zvolené architektury bude rozdělen do 3 vrstev, což umožňuje rozdělit funkcionalitu systému a umožnit jeho stadnější nasazení. Další výhodou je usnadnění tvorby různých rozšíření (viz kapitola Rozšíření).

4.1 Architektura

Pro implementaci požadovaného informačního systému byla zvolena architektura **klient – server** založená na webovém rozhraní. Server slouží jako datové úložiště a zázemí pro provoz IS. Na webovém serveru je kompletně nainstalována celá aplikace a společně s webovým serverem umožňuje se systémem pracovat prostřednictvím sítě internet. Tímto jsou také odstraněny náklady na distribuci a instalaci aplikací na klientské počítače, těm stačí pouze některý ze standardních webových prohlížečů – Internet Explorer, Mozilla Firefox apod.



Architektura systému 4.1:1

A**rchitektura systému (na obr. 4.1:1) je složena ze 3 vrstev – prezenční, aplikační a databázová**. Rozhraní a zprostředkování informací uživateli vytváří *prezenční vrstva*, jež se skládá se sady komponent, které pro přihlášeného uživatele generují přehledně uspořádaná data uživatele. Prezenční vrstva komunikuje s *business vrstvou*, která se stará o aplikaci obchodních pravidel a transformaci dat z databáze na výstup pro prezenční vrstvu. Samotná business vrstva nepřistupuje do

databáze, ale využívá *databázové vrstvy*, která v sobě zapouzdřuje provádění operací s daty v databázi prostřednictvím SQL příkazů, nebo voláním procedur uložených na databázovém serveru.

4.2 Objekty systému

Pro správné naprogramování funkcionality systému byly vytvořeny objekty, reprezentující příslušné entity, které jsou součástí aplikační logiky. Následuje specifikace jednotlivých objektů, které jsou implementovány jako třídy (kromě objektu Osoba) a mají příslušné metody a atributy, které definují jejich funkci a vzájemné vazby. Pro perzistentní uložení objektů slouží databáze a její tabulky, jejichž struktura odpovídá požadavkům na ukládané informace o jednotlivých objektech.

4.2.1 Projekt

Reprezentuje jeden vyvíjený projekt a je základním objektem v systému. Pro popis projektu slouží:

- Unikátní číselný identifikátor, Jméno, Datum zahájení, Datum ukončení, Manažer projektu, Popis projektu

Důležitou součástí je také plán projektu. Skládá se z jednotlivých workflow. Projekt se tak hierarchicky dělí na jednotlivé části, které následně obsahují sérii úkolů. Výsledkem plánování projektu je vytvoření harmonogramu, který je pak základním prostředkem pro jeho řízení. Obsahuje informace kdy a kým mají být prováděny jednotlivé činnosti. Samozřejmností je možnost provádění změn v načasování jednotlivých částí projektu tak aby bylo možné reagovat na nové požadavky, které mohou vznikat v průběhu projektu. Rozdělením projektu do jednotlivých workflows může být stanovené základní rozdělení do jednotlivých iterací a jejich detailní plánování může být prováděno průběžně, tak jak je doporučováno metodikou RUP.

4.2.2 Workflow

Workflow modeluje určitou část projektu, která je definována takto:

- Unikátní číselný identifikátor, Jméno, Datum zahájení, Datum ukončení, Nadřazený workflow, Předchozí workflow, Popis

Je charakterizován svým časovým vymezením a množinou činností, které jsou v jeho rámci prováděny. Workflow zpravidla definuje určitou část vývoje projektu, která jako vstup přijímá určitý vstupní produkt a jejím výsledkem je výstupní produkt, který rozšiřuje jeho funkcionalitu. Proto také obsahuje informace o předchozím resp. Následujícím workflow. Vzniká tak návaznost jednotlích částí projektu a jednoznačný předpis v jakém pořadí mají být prováděny.

Rozložením projektu do jednotlivých workflows na nejvyšší urovni definuje jednotlivé iterace projektu a vytváří milníky, které jsou zpravidla také místem, kdy je uvedena nová verze vyvíjeného systému s rozšířením jeho funkcionality o další množinu definovanou v jeho funkční specifikaci.

Workflows tak umožňují plánování projektu založené na iteračním vývoji což je jeden ze základních principů RUP.
Pro možnost vytvoření složitější struktury projektů mají worfklows definován také svůj nadřazený workflow. To umožňuje hierarchickou stavbu struktury projektu a umožňuje detailnější rozdělení na menší části.

4.2.3 Úkol

Úkol představuje jednu pracovní jednotku v rámci projektu. Jedná se o definici specifické činnosti pro jednoho pracovníka, který ji má provést. Je definována takto:

- Unikátní číslo, Jméno, Datum zahájení, Datum ukončení, Předpokládaná délka (hod.), Identifikátor pracovníka, Identifikátor workflow, do kterého patří, Identifikátor projektu, do kterého patří, Identifikátor stavu

Jednotlivé úkoly vycházejí z činností, které jsou definovány při návrhu systému. Ty jsou poté zařazeny do workflows a tak vzniká plán, kdy mají být realizovány.
Jednotlivé úkoly však mohou mít 0 až *n* prerekvizitních úkolů. Je úlohou projektového manažera stanovit prioritu jednotlivých úkolů a jejich správnou skladbu do harmonogramu tak, aby byla jejich návaznost správná. Pro zjednodušení vazeb je možné specifikovat prerekvizitní úkoly pouze v rámci workflow, do kterého patří. Při vytvoření nového úkolu a jeho zařazení do workflow je velmi důležité definovat jeho prerekvizity, jelikož to umožňuje systému sledovat vzájemné vazby a kontrolovat správná data naplánování úkolů.
Během samotné realizace projektu je také důležité sledovat jak probíhají práce na projektu, resp. Na jednotlivých úkolech. Vzhledem k tomu, že jednotkou práce na projektu je úkol, je odvedená práce sledována na jeho úrovni. Pracovník proto vždy po odvedení práce zadá odpracovanou sumu hodin do karty příslušného úkolu. Tím je ledováno množství času, které bylo potřeba na jeho dokončení a posléze je možné provádět výpočty odpracovaných hodin na jednotlivých workflows či celém projektu.
Jednotka práce obsahuje tyto informace:

- Identifikátor úkolu ke kterému byla práce provedena, Čas přidání do systému, Počet hodin

Během práce prochází úkol několika stavy. Po jeho vytvoření je ve stavu *naplánovaný* až do chvíle než je započata práce. Po přidání prvních odpracovaných hodin se stav úkolu mění na stav *ve vývoji*. Ten trvá tak dlouho, dokud jej pracovník neoznačí jako *dokončený*. Po dokončení je úkol schválen projektovým managerem a stane se *potvrzeným*. Pro různé případy může mít úkol také stav *pozastavený* či *zrušený*.
Vzhledem k nutnosti komunikace mezi managerem a pracovníky, kteří realizují jednotlivé úkoly je do karty úkolu zařazena také možnost posílání vzájemných zpráv. Ty slouží obecně ke komunikaci mezi manažerem a pracovníkem, např. při upřesňování některých podrobností atd.

- Identifikátor úkolu ke kterému náleží, Identifikátor pracovníka který jej vložil, Datum vložení, Samotnou zprávu ve formátu HTML

Pozn.: Jednotlivé komentáře každého pracovníka jsou zobrazeny na úvodní obrazovce, tak aby měl vždy aktuální přehled.

4.2.4 Společnost (klient)

Představuje v systému objekt reprezentující společnost a to v roli zákazníka, pro kterého je určitý projekt vyvíjen. Obsahuje následující informace:

- Unikátní číselný identifikátor, Název společnosti, IČO, DIČ, Adresa, Město, PSČ, Země, Telefon, Fax, Email

4.2.5 Osoba

Obsahuje informace o osobě, zejména identifikační a kontaktní údaje:

- Unikátní číselný identifikátor, Jméno, Adresa, Město, PSČ, Země, Telefon, Fax, Email

4.2.6 Pracovník

Pracovník je objekt reprezentující uživatele systému. Obsahuje tyto údaje:

- Login do systému, Heslo, Hodinovou sazbu, Počet hodin které odpracuje během jednoho měsíce, Identifikátor kontaktu

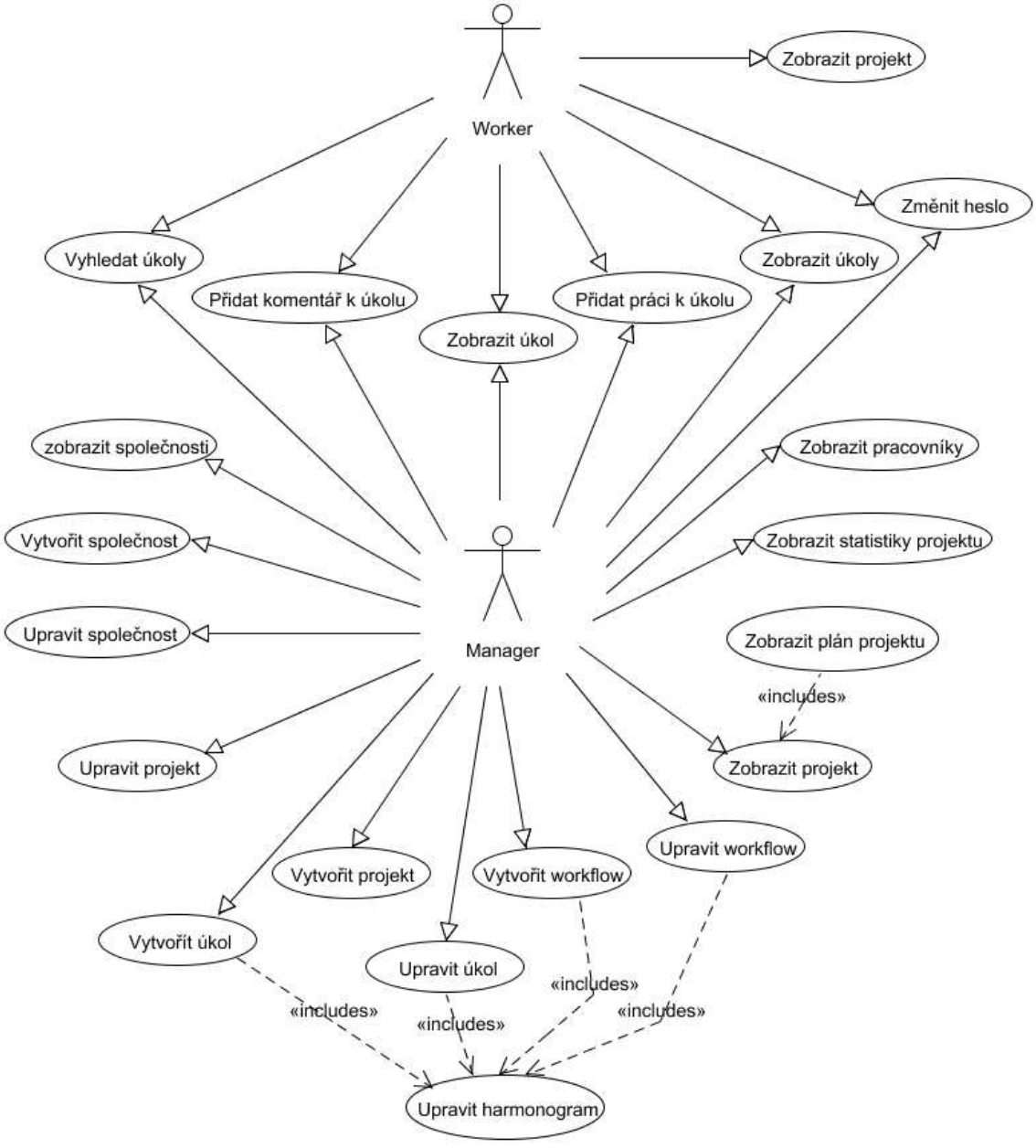
Při vytváření pracovníka v systému je automaticky vygenerovat také nový kontakt.

4.3 Model případu užití (role uživatelů)

Uživatelé v systému mohou mít 2 základní role:

- Worker (Pracovník) – pracuje na jednotlivých úkolech
- Project manager (Projektový manažer) – plánuje a řídí projekty

Jednotlivé role se liší akcemi, které mají v systému povoleny. Přehledně jsou zobrazeny na diagramu.



Use case 4.3:1

4.4 Diagramy tříd aplikační vrstvy

Aplikační vrstva tvoří základní logiku a funkčnost celého systému. Na jedné straně spolupracuje s prezenční vrstvou, do které generuje data a z které přijímá požadavky a události od uživatele. Na druhé straně spolupracuje s databázovou vrstvou, které zasílá požadavky na data a následně je zpracovává a podle aplikačních pravidel transformuje k zobrazení v prezenční vrstvě.

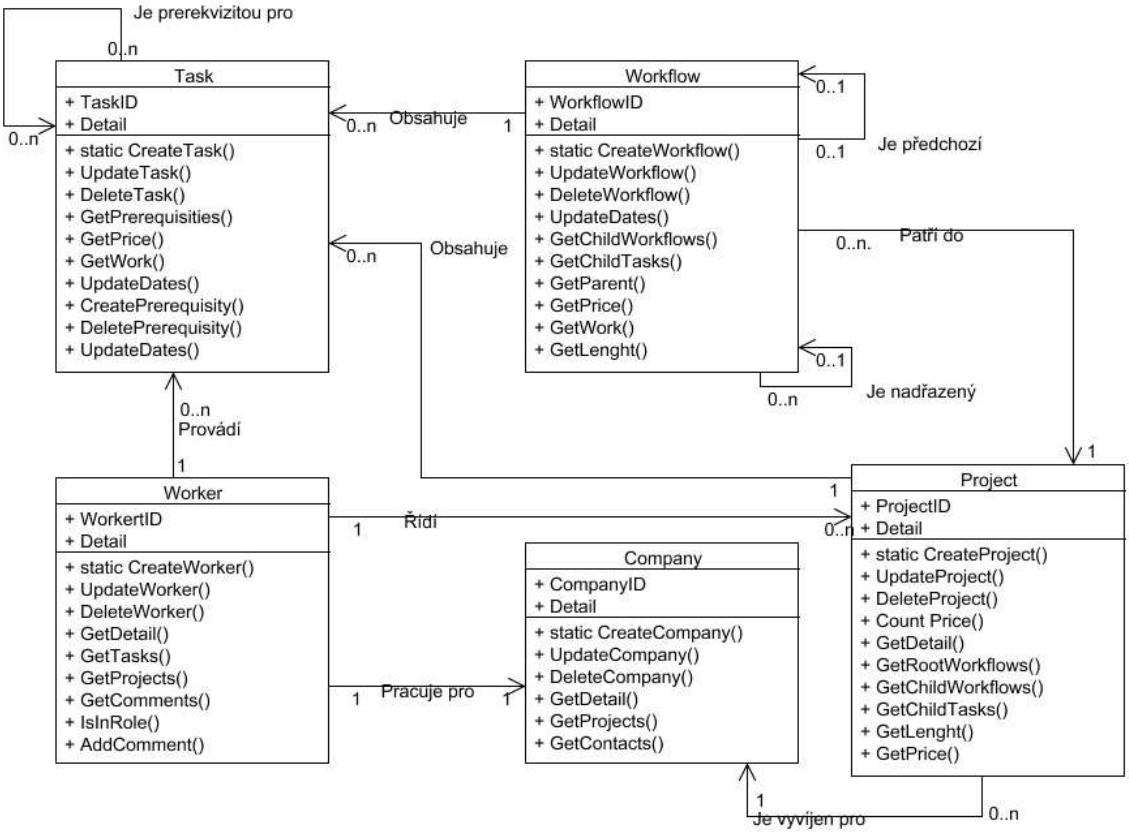


Diagram tříd aplikační vrstvy 4.4:1

Na obrázku 4.4:1 vidíme třídy reprezentující jednotlivé objekty, které vytvářejí činnost aplikační vrstvy. V kapitole 4.2 Objekty systému je blíže popsána jejich funkce v systému.

4.5 Databáze

Databáze se skládá z tabulek, které jsou zobrazeny v diagramu Schéma databáze 4.5:1. Pro urychlení práce s databází byly vytvořeny pohledy:

- TaskAtWorkflow – obsahuje informace o úkolu, workflow do kterého je zařazen, o projektu, manažerovi a pracovníkovi, který jej realizuje
- Project – obsahuje informace o projektu, příslušném zákazníku a managerovi

- Worker – spojuje informace z tabulky pracovníka s přislušným záznamem z tabulky kontaktů a záznamem z aspnet_User, která obsahuje přihlašovací údaje

Součástí navrhovaného řešení jsou tabulky z databáze označené prefixem *prm_*. Tabulky s prefixem *aspnet_* byly vygenerovány interním systémem správy uživatelů, který je součástí .NET Frameworku. Schéma tabulek databáze je na obrázku 4.5:1

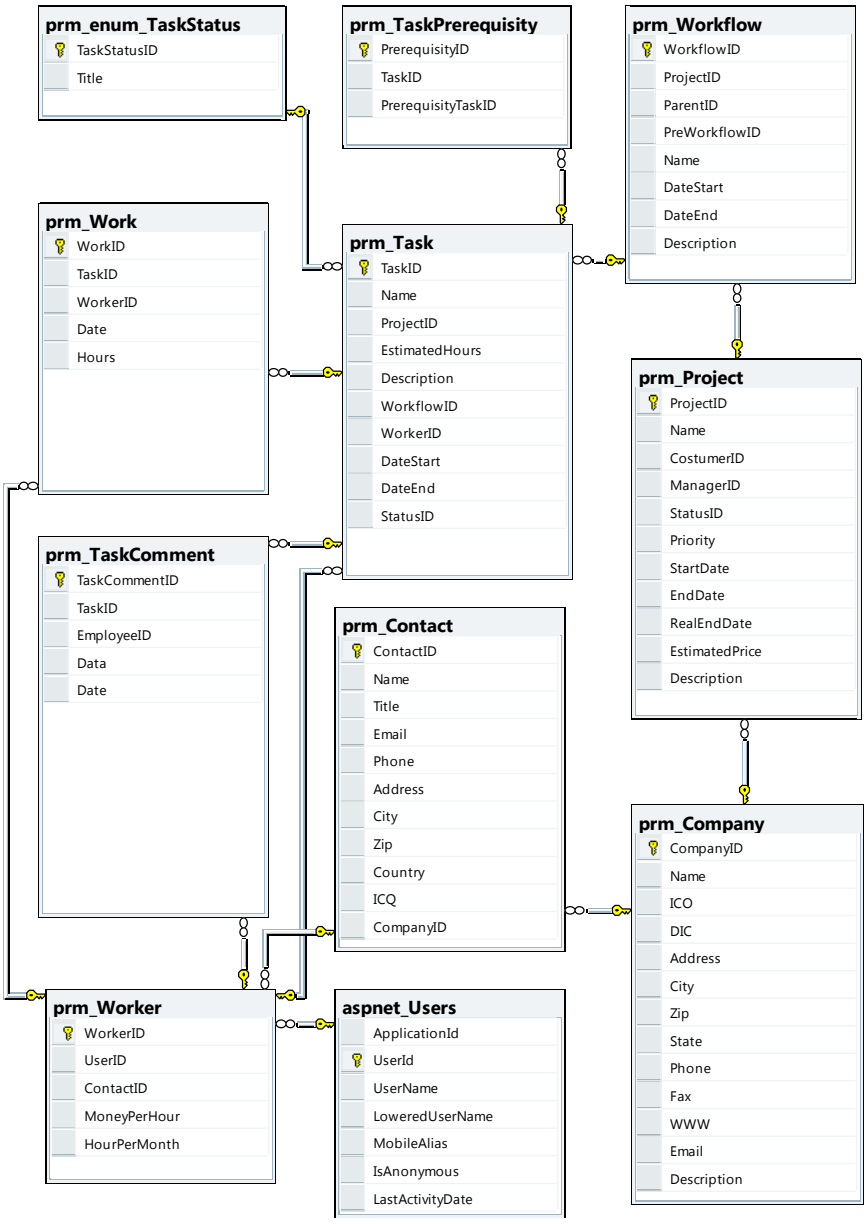


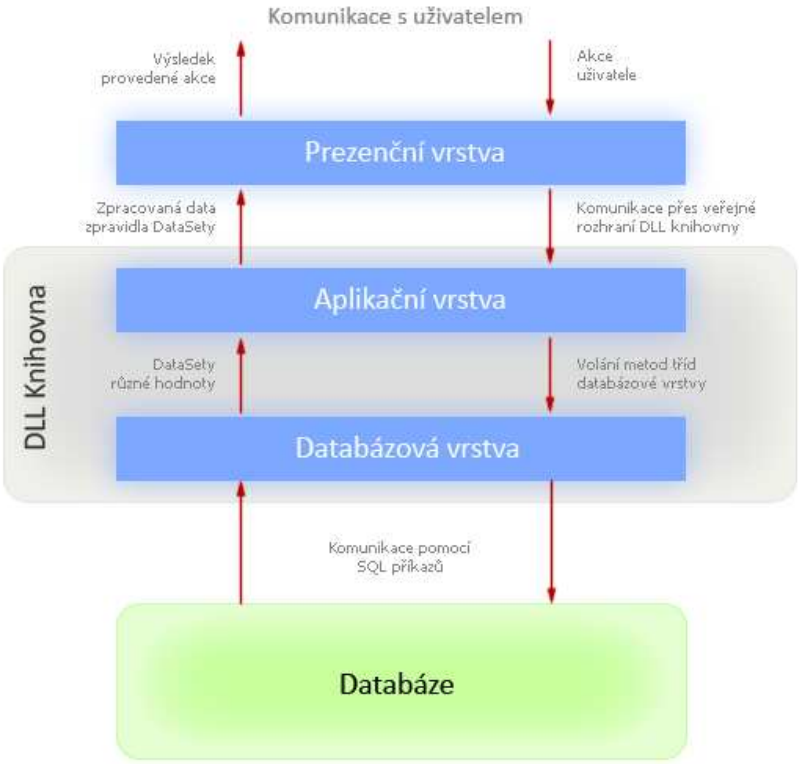
Schéma databáze 4.5:1

Shrnutí

Během návrhu byl vytvořen ucelený model celého vyvýjeného systému, který byl popsán a namodelován pomocí jazyka UML. Vytvořené diagramy se staly východiskem pro následnou implementaci systému.

5 Implementace

Pro implementaci bylo zvoleno prostředí ASP.NET a programovací jazyk C# a to zejména díky velmi kvalitnímu vývojovému prostředí Microsoft Visual Studio 2005, vysoké bezpečnosti architektury .NET Frameworku a možnosti rozdělení implementace systému do více programovacích jazyků podle potřeby. Jak bylo specifikováno v kapitole *4 Návrh* (str. 28), architektura systému se skládá z prezenční, aplikační a databázové vrstvy.



Implementace architektury systému 5:1

Vzhledem k vysoké provázanosti aplikační a databázové vrstvy, byly tyto implementovány do společné DLL knihovny (obr. 5:1), která v sobě zapouzdřuje kompletní funkcnionalitu systému. Nutnou součástí naprogramované knihovny je správně nainstalovaná MSSQL databáze. Takto vytvořená knihovna (společně s databází) definuje API rozhraní veřejných tříd a jejich metod, které je možné využít jak k získávání tak k zadávání dat do databáze podle navržené funkcionality systému. Takto oddělené části aplikace byly navrženy zejména pro možnost budoucího rozvoje systému, jakým může být např. napojení plnohodnotné Windows desktop aplikace určené pro plánování projektů na vytvořený systém. Toto rozšíření je popsáno v kapitole *6.1 Aplikace pro plánování projektů* (str. 42).

5.1 Bezpečnost

Správné zabezpečení systému se stalo jedním ze základních požadavků na vytvářený systém, jelikož je nutností pro profesionální softwarová řešení.

Jedná se zejména o zabránění vstupu neautorizovaným osobám, či provádění činností, ke kterým nemá příslušný uživatel právo. Základní struktura bezpečnostních funkcí vyvíjeného systému proto vychází z interního ASP.NET modelu, který řeší požadavky jak z oblasti autentifikace, tak také z oblasti autorizace. Samotný .NET Framework také obsahuje implementaci důvěrnosti a integrity, založené na digitálních podpisech a kvalitním šifrování. Jedná se o velmi dobře otestovaný systém založený na bezpečném ukládání autentizačních cookies na počítač klienta, který se systémem právě pracuje i na vzdálené stanici.

Současně se zavedenými bezpečnostními principy byly dodrženy doporučení a pravidla zásad bezpečného programování. Základními kroky, které přispěly k větší bezpečnosti systému jsou:

- **Kontrola uživatelského vstupu** – informace zadané uživatelem jsou od něj přejímána tak, aby modifikací vstupních dat nedocházelo k některým vyjímečným stavům, které lze považovat za útok na bezpečnost systému.
- **Bezpečný přístup do databáze** – komunikace s databází je uzavřena uvnitř databázové vrstvy a je striktně oddělena od ostatních vrstev systému. Samozřejmostí je parametrizované vytváření dotazů do databáze, což zabraňuje zejména populárním SQL Injection útokům.
- **Zabezpečení dat** – Samotná data, která jsou považována za citlivá jsou ukládána pouze v databázi a nejsou uchovávána v žádnými jinými metodami jako například předávány v URL či ve skrytých formulářových polích

Jedním z důležitých kamenů při návrhu a implementaci bezpečnosti je přítomnost tzv. strážných (Gatekeepers) (Kapitola 19 Bezpečnostní model ASP.NET, [1]. Tento model je založený na proudovém zpracování (pipelining model), což je určitý model, který umožňuje zvýšit zabezpečení aplikace. Při použití tohoto modelu se předpokládá použití více mechanismů než by mohlo být ve skutečnosti nutné. Každý z jednotlivých mechanismů je implementován jako tzv. strážný, které je odpovědný za dodržení určitých podmínek pro pokračování ve zpracování prováděné operace. Pokud se útočníkovi podaří obelstít jednoho strážného, narazí na dalšího. Strážní mechanismy jsou tedy implementovány jak v prezenční tak v aplikační vrstvě. V prezenční vrstvě se jedná zejména o samotnou autentifikaci a autorizaci uživatele a ověřování základních přístupových práv k jednotlivým zdrojům. V aplikační vrstvě jsou kontrolována práva k provádění operací, které manipulují s citlivými daty.

Autentizace uživatelů probíhá na bázi Formulářové autentifikace, která je zakomponována do .NET Frameworku. Ta dovolí do systému vstup pouze uživatelům s platným uživatelským heslem a jménem. Po úspěšném přihlášení uživatele je autorizován, také interním mechanismem .NET Frameworku, a je určeno, do kterých rolí uživatel náleží. Ve výsledku máme po provedení k dispozici informace o tom, který uživatel je aktuálně přihlášen a jaká má v systému práva. Ta jsou poté ještě několikrát ověřována při přístupu k chráněným datům či při akcích, které s nimy manipulují.

Zabezpečení chráněných dat 5.1:1

Jedinným zásahem do interního autentifikačního a autorizačního systému je přesunutí informací o uživatelích systému do databáze. Jedná se o flexibilnější variantu, která umožňuje pohodlněji manipulovat s daty a navíc umožňuje lepší zabezpečení, jelikož implicitně jsou data ukládána v souboru web.config, které je snázeji napadnutelný než samotná databáze.

5.2 Prezenční vrstva

Prezeční vrstva je realizována jako ASP.NET webové stránky. Při jejich vytváření bylo využito zejména velmi robustního .NET Frameworku, který nabízí velmi kvalitní řešení pro většinu požadavků, které byly na systém kladeny. Základními požadavky na vyvíjenou prezenční vrstvu jsou:

- Uživatelská přívětivost – jednotné a přehledné grafické uživatelské prostředí
- Bezpečnost – zejména správné zpracování vstupních dat

Z hlediska uživatelské přívětivosti systém nabízí pohodlné možnosti pro práci se systémem. Prezenční vrstva je charakterizována jednotným grafickým rozhraním, které je vytvářeno pomocí CSS (kaskádových stylů) a interního systému Themes, který poskytuje .NET Framework. Uživatelské rozhraní je navíc možno pouze změnou těchto Themes a CSS pravidel možno graficky změnit a přizpůsobit např. firemnímu stylu společnosti.

Vzhledem k tomu, že právě prezenční vrstva tvoří pro uživatele vstupní rozhraní do systému a je implementována jako webová prezentace, byla kladena velká důležitost na její správné vytvoření tak, aby výsledek vyhovoval dnešním standardům kladeným na internetové prezentace, či obecně systémy fungující na internetu. Vytvořený systém tak splňuje validační kritéria XHTML 1.0 Strict, která jsou určena W3C konsorciem. Tento fakt společně se správně vytvořenými pravidly v kaskádových stylech umožňuje použití systému v běžně dostupných prohlížečích, které využívají

standardizované postupy pro zobrazení webových stránek. Systém byl otestován zejména v prohlížečích Internet Exploreru 7, Internet Exploreru 6, Mozilla Firefox 1.5 a Opera 9.02, které jsou nejpoužívanějšími na trhu.

Dalším důležitým prvkem v řešení prezenční vrstvy je přítomnost tzv. Control Adapters, což je mechanismus, který umožňuje optimalizovat vygenerovaný XHTML kód. Vzhledem k tomu, že si společnost Microsoft nemohla dovolit přerušit zpětnou kompatibilitu i s jejími nejstaršími prohlížeči, jako jsou první verze Internet Exploreru, je kód generovaný pomocí ASP.NET v některých případech velmi neefektivní. Zejména proto, že můžeme u uživatelů systému předpokládat použití modernějších prohlížečů, byly do systému implementovány Control Adapters, které optimalizují vygenerovaný kód a tím urychlují práci systému. Zejména při generování rozsáhlejších tabulek může jejich použití znamenat značnou úsporu a tedy snížení velikosti přenášených dat.

Pro pohodlnější usnadnění práce se systémem jde implementace ještě dále. Do systému byly implementovány některé prvky založené na dnes velmi populární technologii AJAX. Systém obsahuje použití některých volně dostupných prvků z kolekce AJAXControlToolkit. Tyto prvky byly použity zejména pro usnadnění plánování projektů. Jedná se o nadstandardní doplnění funkcionality HTML formulářů. Při vkládání nových činností, či zakládání projektu není uživatel nucen vypisovat datum přesně v zadaném formátu, ale při aktivaci příslušného pole mu systém nabídne velmi propracovaný a pohodlný kalendář, který umožňuje pohodlné zadání data i pouze pomocí myši. Dalším s využití technologie AJAX je zabránění vkládání neplatných znaků do formulářových polí. Například je-li potřeba zadávat jen čísla, jsou veškeré ostatní znaky odfiltrovány. Samozřejmostí je také následná kontrolu správnosti formátu při odesílání dat na server. Technologie AJAX také nabízí možnost asynchronního načítání dat založené na webových službách. Vzhledem k jeho kvalitnímu zapracování do .NET Frameworku bylo jeho nasazení velmi jednoduché. Ulehčuje tak práci při sledování statistik projektu, kdy uživatelské rozhraní pomocí AJAXu načítá historii odpracovaných hodin k aktuálně kontrolovanému úkolu v pozadí a neruší tak pracujícího managera častým celkovým znovuzobrazováním celé stránky. I tento princip se snaží o úsporu přenášených dat, kdy není potřeba nahrávat znovu celou stránku, ale pomocí XML schématu jsou mezi serverem a klientem přenášena jen aktuálně potřebná data.

5.2.1 Prezentace dat

Jednou z hlavních činností prezenční vrstvy je právě zobrazení dat uživateli. Pro tyto účely poskytuje .NET Framework širokou škálu nástrojů. Jedná se o tzv. *pokročilé ovládací prvky*, mezi které patří zejména GridView a DetailView, které jsou také nejpoužívanějšími prvky v systému. Daný ovládací prvek je umístěn na stránku a jsou definovány pouze některé základní parametry, které ovlivňují jeho chování. Dalším pomocným mechanismem jsou CSS styly a Themes, které se postarají o správný a jednotný vzhled. Poslední nutností je tak pouze vložení dat do ovládacího prvku. To se v systému

provádí mechanismem, který se nazívá DataBinding. V kontextu webových stránek ASP.NET jsou data reprezentována pomocí DataSetů, které jsou získány z aplikační vrstvy. Jedná se o data, která obsahují kompletní tabulku dat, tak jak ji známe z databáze. Jedinný rozdíl je v tom, že DataSet nemá perzistentní uložení a je ukládán v paměti. Tato data jsou pomocí DataBindingu „navazována“ do příslušných datových ovládacích prvků.

5.2.2 Získávání dat

Další důležitou funkcí prezenční vrstvy je získávání dat od uživatele. Všechna data, která uživatel vkládá jsou do systému zadávána pomocí HTML formulářů. Jednotlivé vstupní hodnoty jsou ověřovány již na straně klienta pomocí JavaScript kódu, který je generován Validátory, což je také jeden z interních principů .NET Frameworku určený právě pro ověřování uživatelského vstupu. Klíčové položky jsou poté ještě jednou kontrolovány před voláním metod tříd aplikační vrstvy a tedy před jejich vložením do systému. Jakákoliv neplatná hodnota, která by mohla projít i přes tyto dva kroky nakonec zkončí vyvoláním vyjímky při vkládání dat přes databázovou vrstvu, což je další mechanismus sloužící k ochraně dat a systému.

Pro pohodlné užití systému je u některých vstupních hodnot do systému zařazen také online wysiwyg XHTML editor. Umožňuje pohodlné vytváření dat, aniž by uživatel musel psát HTML kód. Jedná se o FCKEditor založený na JavaScriptu a šířený jako open source nástroj.

5.3 Aplikační vrstva

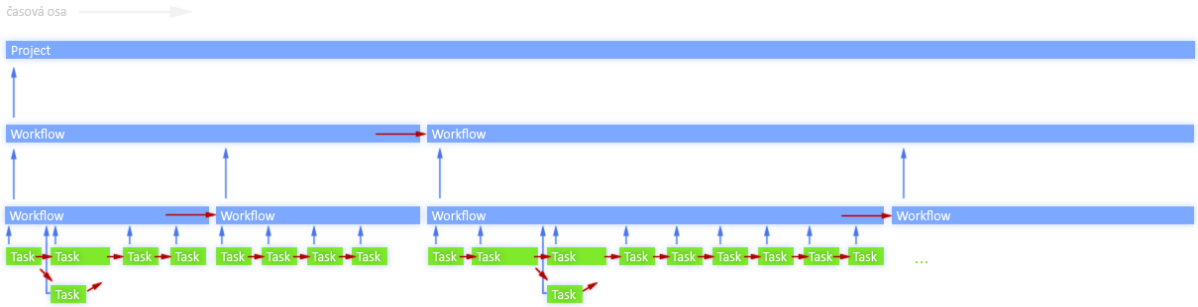
Aplikační vrstva se skládá ze tříd, jejichž diagram naleznete na obrázku Diagram tříd aplikační vrstvy 4.4:1. Třídy aplikační vrstvy jsou jádrem funkcionality celého systému. Jak již bylo napsáno obsahuje definici tříd, jež implementují základní objekty, které v systému fungují. Jejich metody slouží zejména k získávání ucelených imformací o jednotlivých objektech systému. Nejnáročnějším úkolem aplikační vrstvy je však zejména kontrolování integritních pravidel jednotlivých objektů. Jedná se právě o podporu plánování projektů, která je nejnáročnější součástí implementované aplikační vrstvy, jelikož právě při plánování projektu dochází k vytváření vazeb mezi objekty.

5.3.1 Plánování projektů

Vytváření a plánování projektu je činnost, která v systému vede k vytvoření kompletního projektového plánu. Na konci plánování je k dispozici kompletní harmonogram, který obsahuje informace o jednotlivých úkolech, ke kterým jsou přiřazeni pracovníci, čímž je definováno kdo má kdy jakou činnost udělat.

Právě závyslost jednotlivých činností a následně tedy objektů Workflow a Task (úkol), které jsou v systému jejich reprezentací vytváří nemalé požadavky na kontrolu při jejich plánování.

Objekt *úkol* v systému reprezentuje jednu konkrétní činnost pro jednoho pracovníka a objekt *workflow* je tzv. pracovní proces a obsahuje posloupnost úkolů, nebo další podřazené workflows. Vzhledem k tomu, že jedním ze základních požadavků je právě flexibilita systému při plánování obsahuje aplikační vrstva funkce pro jeho podporu. Pokud dochází k přidávání úkolů do workflow, který je definován na časové rozmezí, které nestačí pro implementaci daných úkolů, je tento dynamicky rozšířen a systém se musí postarat o správné „přeplánování“ dalších workflows, které mají s daným workflow nejakou vazbu (nadřazený, následující, podřazený). Objekty tak dělají vzájemným zasíláním zpráv o nutnosti nového naplánování a předávají si potřebné časové údaje. Výsledkem je, že systém flexibilně upravuje plán celého projektu na základě požadavků projektového manažera, ten nemusí dělat tuto důležitou práci sám s tím že systém se postará o správné načasování jednotlivých etap a činností. Pokud by mělo dojít k plánování workflows či úkolů na čas, který je nereálný, nebo který není možno do projektu začlenit ani po jeho případném „přeplánování“, systém jej upozorní, že zařazení takového úkolu či workflow nelze provést.



5.1.1:1 Vztahy mezi objekty workflow, task a project a algoritmus plánování

Popis obrázku 5.1.1:1 Vazba mezi objekty znázorněná modrou šipkou říká, že jeden objekt je podřazen druhému. Vazba znázoněná červenou šipkou znamená že jeden objekt následuje za předchozím. Dojde-li k úpravě načasování jednoho z těchto objektů, musí být rekurzivně posunuty objekty, které na sebe ukazují červenou šipkou tak, aby ten na jejím konci začínal když předchozí končí. Pokud podřazené objekty přesahují rámec nadřazeného, šíří se zpráva o nutnosti aktualizace data po modré šipce k nadřazeným objektům, které ji poté posílají po červené šipce k dalšímu objektu, a následně opět do nejnižší úrovně proti směru modrých šipek a proces se opakuje dokud se nenarazí na konec projektu.

5.3.2 Zachování integrity při mazání a úpravách objektů

Dalším důležitým úkolem, který plní aplikační vrstva je zachování integrity dat, které vychází s aplikačích pravidel. Systém tak zabraňuje mazání úkolů, na kterých již byla odvedena práce, mazání firem, pro které jsou nebo byly vyvíjeny projekty apod.

5.4 Databázová vrstva

Skládá se z tříd, jejichž cílem je oddělení databáze od aplikační logiky. Tyto třídy vytvářejí jednotný vstupní bod pro práci s databází. Pokud by bylo při nasazení systému do praxe nutno použít jiný databázový server, postačí pouze úpravy databázové vrstvy. Pokud by se jednalo o databázi, která je podporována .NET Frameworkem, postačí změny na úrovni DataTableAdaptérů, pokud by se jednalo o jakýkoliv nepodporovaný typ databáze či jiného úložiště, je možnost upravit jednotlivé třídy databázové vrstvy, které právě poskytují naprostou flexibilitu při úpravách napojení systému na databázi.

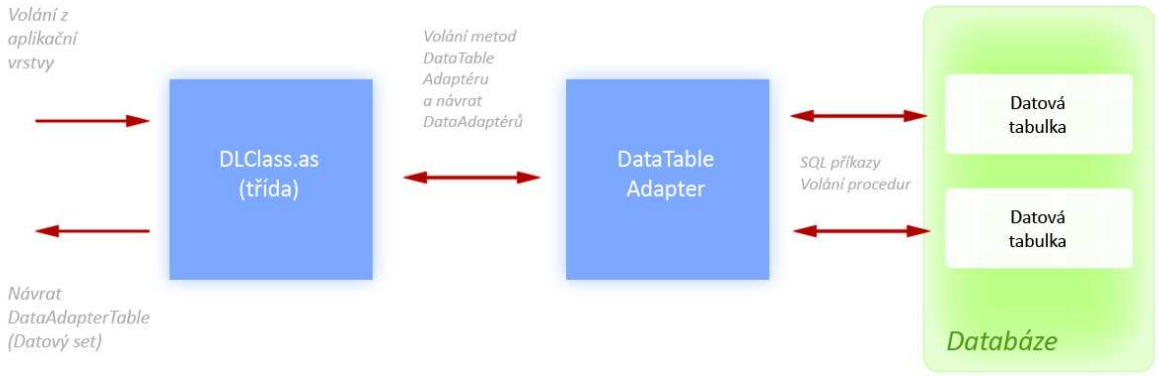


Schéma databázové vrstvy 5.4:1

Jednotlivé třídy databázové vrstvy jsou označeny prefixem „DL“, jsou uloženy ve společném *namespace DatabaseLayer*, jsou statické a obsahují statické metody, které je možné volat z aplikační vrstvy. Tyto třídy dále používají *DataSet*, který obsahuje množství *DataTableAdapterů*, které slouží pro napojení na databázové tabulky. DataAdaptéry obsahují jednotlivé dotazy do databáze a to buď formou SQL příkazů, nebo ve složitějších případech formou volání uložených procedur na SQL serveru. Adaptér zpravidla pracuje nad jednou množinou potřebných výsledků, například *seznam úkolů*, který se skládá z dat, která jsou získána z několika dalších tabulek databáze.

Shrnutí

Během implementace se vycházelo z předem definovaných dokumentů, zejména z UML diagramů či funkční specifikace.

6 Rozšíření

Navržený informační systém nabízí spoustu variant pro mnoho dalších rozšíření. Jedná se zejména o přidávání dalších nástrojů pro zdokonalení současné funkcionality či pro podporu dalších oborů projektového managementu.

6.1 Aplikace pro plánování projektů

Jedním z nejzajímavějších rozšíření je možnost napojení současného systému na plnohodnotnou desktop aplikaci, která by mohla nabídnout větší pohodlí a lepší funkčnost pro plánování projektů a zejména pro práci projektového manažera. Toto rozšíření by zachovalo možnost použití systému v distribuovaném prostředí internetu, aplikace by byla instalována pouze na počítače projektových manažerů. Ti by tímto získali plnohodnotnější software zejména pro plánování projektu s možností správy vytíženosti jednotlivých zdrojů a podobně jak je tomu např. v aplikaci Microsoft Project. Po vytvoření či modifikaci plánu projektu by došlo k jeho uploadu na webový server. Jednotlivý pracovníci pracují pouze s těmito uploadovanými daty a využívají současné funkcionality systému. Během průběhu projektu aplikace stahuje aktuální data o provádění projektu a aktualizuje informace pro projektového manažera, který tak stále má ucelený přehled o průběhu projektu.

6.2 Správa souborů

Vzhledem k tomu, že při zadávání některých úkolů je potřeba komplexnějších informací, například diagramy z předchozích etap projektu atd. nabízí se jako jedno s možných rozšíření systému správa souborů. Manažer projektu by měl možnost přidávat libovolná data k jednotlivým úkolům a pracovník by mohl prostřednictvím systému např. odevzdávat vypracované zdrojové kódy či dokumentaci. Správa souboru by tak efektivně umožnila zefektivnit správu dokumentů celému týmu, jelikož by existovalo jednotné a bezpečné úložiště jejich binárních dat.

6.3 Správa zdrojů

Správa zdrojů je nezbytnou součástí nejmodernějších nástrojů určených pro plánování projektů. Proto by se mělo stát jedním ze základních rozšíření současného systému. Detailní správa zdrojů, zejména kalendáře pracovních dnů jednotlivých členů týmu umožňuje projektovému managerovi lepší přehled nad přiřazováním zdrojů k jednotlivým úkolům a umožňuje přesnější naplánování projektu.

6.4 Správa požadavků

Vzhledem k předpokladu, že systém bude použit v dynamickém prostředí, stává se správné zpracování a třídění požadavků nezbytnou součástí plánování. Správa požadavků musí obsahovat mechanismy pro jejich třídění, zejména z pohledu jejich priorit a také mechanismy, které pomáhají projektovému managorovi v efektivním plánování jejich implementace do realizovaného projektu.

6.5 Import dat z Microsoft Project

Microsoft Project je jedním z nejmodernějších nástrojů projektového managementu, který je dnes k dispozici. Jakožto moderní nástroj umožňuje ukládání plánu projektu do XML souborů, které je poté možné zpracovat. Vytvořením modulu pro import XML dat je tak možné napojit současný systém například na tento silný nástroj.

Shrnutí

Vytvořením výše uvedených rozšíření dojde zejména k vylepšení nástrojů pro projektového managera. Plnohodnotná desktopová aplikace by projektovému managerovi dala lepší možnosti plánování, zejména díky větším možnostem uživatelských rozhraní desktop aplikací a společně se správou souborů, požadavků a dalších důležitých součástí projektového managementu by mohl vytvořený systém konkurovat i některým komerčním řešením, která v současné době existují.

Závěr

Vytvořený softwarový systém naplňuje požadavky definované během analýzy a uspokojuje všechny důležité požadavky, které na něj byly kladeny. Jedná se o bezpečný a rozšířitelný informační systém, který již ve své podobě postačuje pro plánování a řízení malých a středních projektů. Jedním z důležitých prvků vytvořeného systému je také požadovaná podpora principů metodiky RUP.

Avšak pro nasazení ve větších projektech, kde jsou vyžadovány velmi kvalitní postupy projektového managementu je potřeba stávající implementaci rozšířit zejména o v poslední kapitole navržená řešení. V případě jejich implementace by byl vytvořen mnohem robustnější softwarový systém, který by umožňoval sofistikovanější metody pro řízení náročnějších projektů.

Literatura

[1] MacDonald, M., Szpuszta, M. *ASP.NET 2.0 a C# tvorba dynamických stránek PROFESIONÁLNĚ*. I. Vydání. Zoner Press, Brno, 2006. ISBN: 80-86815-38-2.

[2] Gibbs, R. D. *Project Management with the IBM® Rational Unified Process®: Lessons from the Trenches*. IBM Press, 2006. ISBN-10: 0-321-33639-9

[3] Brandon, D. Phd. PMP *Project Management for Modern Information Systems*. IRM Press, 2005. ISBN: 1-59140-695-1

[4] Wikipedia *IBM Rational Unified Process* [online]. Poslední modifikace 1.5.2007 [cit. 1.4.2007]. Dostupné na URL: <http://en.wikipedia.org/wiki/Rational_Unified_Process>.

[5] Puš, P. *Poznáváme C# a Microsoft .NET*. [online]. Poslední modifikace 20.9.2006 [cit. 1.4.2007]. Dostupné na URL: <http://www.zive.cz/h/Programovani/AR.asp?ARI=119978>. Jedná se o seriál 78 článků.

[6] Kreslíková, J. *Přednášky k předmětu IRP – řízení projektů informačních systémů*. Brno, Fakulta informačních technologií VUT Brno, 2006.

[7] Kadlec, V. *Rational unified process – Základní pojmy* [online]. Poslední modifikace 5.8.2006 [cit. 1.4.2007]. Dostupné na URL: <http://www.zive.cz/h/Programovani/AR.asp?ARI=112011&CAI=2039>

[8] Bittner, K., Spence, I. *Managing Iterative Software Development Projects*. Addison Wesley Professional, 2006. ISBN: 0-321-26889-X

[9] Gary, Ch. *Agile Project Management: How to Succeed in the Face of Changing Project Requirements.* AMACON, 2004. ISBN 0814471765.

[10] Kadlec, V. *98 % zakázek neúspěšných? Ještě že máme softwarové inženýrství!* [online]. Poslední modifikace 27.2.2005 [cit. 1.5.2007]. Dostupné na URL: <http://www.zive.cz/h/Programovani/AR.asp?ARI=104343&CAI=2039>

Seznam příloh