VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

ZHODNOCENÍ PROJEKTU A NÁVRH METODIKY PROJEKTOVÉHO ŘÍZENÍ

PROJECT EVALUATION AND DESIGN OF PROJECT MANAGEMENT METHODOLOGY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ONDŘEJ PLUHAŘ
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ KŘÍŽ, Ph.D.
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2011/12
Ústav informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Ondřej Pluhař

Informační management (6209T015)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

Zhodnocení projektu a návrh metodiky projektového řízení

v anglickém jazyce:

Project Evaluation and Design of Project Management Methodology

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

DOLEŽAL, J.; LACKO, B.; MÁCHAL, P. Projektový management podle IPMA. Praha: Grada Publishing, a.s., 2009. 507 s. ISBN 978-80-247-2848-3.
FOTR, J.; SOUČEK, I. Investiční rozhodování a řízení projektů. Praha: Grada Publishing a.s., 2011. 408 s. ISBN 978-8-024-73293-0.
CHARVAT, J. Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects. New Jersey: John Wiley and Sons, Inc., 2003. 264 s. ISBN 978-0-471-22178-4.
PHILLIPS, J. IT project management: on track from start to finish. 2. Emeryville, CA: McGraw-Hill Professional, 2004. 535 s. ISBN 978-0-072-23202-8.
STACKPOLE, C. A User's Manual to the PMBOK Guide. New Jersey: John Wiley and Sons, Inc., 2010. 240 s. ISBN 978-0-470-58489-7.

Vedoucí diplomové práce: Ing. Jiří Kříž, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2011/12.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA PODNIKATELSKÁ 14 L.S.

Ing. Jiří Kříž, Ph.D.
Ředitel ústavu

doc. RNDr. Anna Putnová, Ph.D., MBA
Děkanka

V Brně, dne 23.3.2012

Abstrakt

Diplomová práce se zabývá problematikou projektového řízení – řízení činností při vývoji software. Vedle vysvětlení základních pojmů z dané oblasti, pojednává o výhodách a nevýhodách některých známých postupů, které jsou při tvorbě softwaru často následovány.

V praktické části je analyzován průběh konkrétního projektu, jeho zhodnocení a na jehož základě je dále navržena optimalizace použité metodiky vývoje, která by mohla pomoci při řízení podobných projektů v budoucnosti.

Abstract (EN)

The diploma thesis focuses on project management – managing of software development activities. It explains a basic terms of given issue and it also deals with advantages and disadvantages of some known practices and methods which are often followed when developing software.

The practical part includes an analysis and evaluation of concrete project. This project is then the base for designing an optimization of software development methodology. Designed optimization will be able to help manage similar projects in future.

Klíčová slova

Analýza, fáze projektu, informační systém, ITIL, metodika, optimalizace, projekt, projektový tým, RUP, testování, vývoj, životní cyklus

Key words

Analysis, development, information system, ITIL, lifecycle, methodology, optimization, project phases, project team, RUP, testing

Bibliografická citace

PLUHAŘ, O. *Zhodnocení projektu a návrh metodiky projektového řízení*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2012. 68 s. Vedoucí diplomové práce Ing. Jiří Kříž, Ph.D..

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským, ve znění pozdějších předpisů).

V Brně dne 23. května 2012

Podpis: …….....

Poděkování

Tímto bych chtěl poděkovat Ing. Jiřímu Křížovi, Ph.D. za odbornou pomoc při sepsání této diplomové práce.

Obsah

Úvod

V dnešní době se pro vývoj informačních, operačních i jiných systémů běžně využívají postupy podporující týmovou práci na projektech. Tento trend je pochopitelný, protože rozsah těchto systémů bývá natolik široký, že by je jeden člověk nebyl schopen vytvořit v rozumné kvalitě a potřebném čase.

V projektově orientovaných organizacích by se měli pohybovat lidé, kteří jsou ochotní projektu věnovat svůj volný čas a své vědomosti. Je poměrně složité stavět projekty bez předchozích zkušeností, proto se všeobecně doporučuje v takových organizacích začínat na nižší pozici a postupně se vyvíjet a postupovat na pozice vyšší. Kvalitní a zkušený management je totiž mnohdy určující veličinou rozhodující o úspěšnosti celého projektu.

K tomu aby projekt mohl fungovat, je vhodné využívat metodik, jež jsou již ověřené časem a zkušenostmi úspěšných manažerů. Může se jednat o přístupy k řízení lidí, činností, rizik, k posouzení stavu projektových týmů atd. Metodiky často mohou popisovat i vhodnou podobu organizační struktury, která se v dynamických projektových týmech liší od struktur běžných „neprojektových“ společností.

1 Cíl práce

Diplomová práce je zaměřena na problematiku řízení projektu, jehož cílem je vývoj softwarové aplikace. Protože se jedná o projekt, na němž se podílela celá řada pracovníků, bylo při jeho realizaci postupováno podle osvědčených metodik a postupů, které měly pomoci projektovému vedení k úspěšnému dokončení projektu. Společnost se během svých projektů řídí takovými metodikami vývoje softwaru, jako jsou RUP nebo ITIL. Pro samotné řízení projektů se vychází z poznatků metodologie PRINCE2.

V práci bude rozebrán konkrétní projekt z pohledu řízení vývojových činností. Důraz bude kladen především na správnost postupu podle metodiky RUP, podle níž se celý projekt realizuje. Vedle minoritních pochybení některých členů projektového týmu jsou rozebrány hlavní nedostatky projektu během celé doby jeho trvání. Ke všem zmíněným nedostatkům jsou postupně navržena opatření, která kdyby byla nasazena, tak k daným chybám nemuselo vůbec docházet. Konečný výsledek projektu by tak mohl být mnohem pozitivnější, než jak tomu ve skutečnosti bylo.

Další část práce bude věnována návrhu optimalizace metodiky RUP, která by mohla přispět budoucím projektům podobného zaměření. Optimalizace se bude zaměřovat na odstranění nejčastějších chyb, kvůli kterým projekty bývají neúspěšné (příp. by mohla snížit celkové náklady nebo dobu trvání projektu).

2 Teoretická východiska

Tato kapitola je věnována teoretickým rozborům pojmů, se kterými se budou potýkat kapitoly následující.

2.1 Projekt

Jako projekt lze označit soubor činností, které jsou prováděny na základě jasně stanoveného cíle, začátku a konce. Úspěšnost projektu nemůže být předem nikdy zaručena na sto procent, protože se musí pracovat s pevně danými zdroji a všechny postupy se do jisté míry vymykají běžné „mimoprojektové“ praxi. Projekt lze považovat za úspěšný tehdy, jestliže byl dokončen ve stanoveném termínu, náklady na jeho realizaci se pohybují ve stanovené výši a jestli byly splněny všechny dopředu vytyčené cíle (trojimperativ projektu).

Protože projekt v sobě zahrnuje různorodé činnosti lidí z různých oborů a vlastních specializací, je nutné tyto činnosti nějakým způsobem koordinovat a řídit. Vedle řízení samotných pracovníků je nutné spravovat i další důležité prvky charakteristické pro projekty (rizika, náklady, čas, změny, znalosti, zdroje atd.).

K tomu, aby mohlo být stále více projektů řízeno úspěšně, bylo vyvinuto několik metodologií řízení, které jsou celosvětově používány a respektovány (IPMA, PMBOK, PRINCE2 apod.). Vedle těchto uznávaných metodologií si mohou firmy vytvářet i vlastní postupy, které budou vyhovovat právě těmto firmám. Vždy se však jedná o postupy, které jsou nějakým způsobem zaznamenatelné a dopředu připravitelné. [31]

2.2 Informační systém

Informačním systémem se dá nazvat soubor lidí, technologických prostředků a metod, které zabezpečují sběr, přenos, zpracování a uchování dat za účelem tvorby prezentace informací pro potřeby uživatelů.

Informační systémy se využívají k upevňování vnitřní stability firmy. Měly by zde být shromažďovány a udržovány takové informace, které mohou pomoci firmě pružně reagovat na vnější podněty.

Velmi důležitým požadavkem je, aby se v rámci informačního systému firmy pohybovaly pouze relevantní informace, které lze pro daný cíl využít. Je tedy zapotřebí

informace udržovat aktuální, efektivně zapracovávat nové, nepotřebné odstraňovat, ale také distribuovat a využívat. [19]

Obecně lze o informačních systémech říci několik vlastností a zákonitostí. Podle druhu mohou být IS s nebo bez podpory počítačů, přičemž neautomatizované systémy si lze představit např. jako kartotéku knihovny, vedení účetnictví nebo telefonní seznam.

Automatizované IS umožňují mnohem větší záběr zpracovávaných činností a procesů. Proto je velmi důležitou vlastností celkové zaměření vlastního systému – jiné informace potřebuje ke své činnosti prodavač a jiné manažer společnosti. Ve chvíli, kdy je v rámci jedné firmy zapotřebí, aby systém sloužil různým osobám na různých pozicích, je ho možné upravit tak, aby jednotlivci nabízel pouze určitou část své funkčnosti či uložených dat. [29]

Když se společnost rozhodne pro zavedení informačního systému do svých činností, má na výběr prakticky ze tří možností, jak jej získat.

První z nich je vývoj nového IS přesně na míru dané společnosti. Tento způsob je vhodný pro firmu pohybující se ve specifickém odvětví. Jde o způsob získání IS, který je velmi náročný na čas a finance. Stejně tak nelze vyloučit přítomnost chyb v prvních verzích. Na druhou stranu je ale výhodný v tom, že firma si může sama nadefinovat, jak má systém vypadat a fungovat a přizpůsobit si ho tak svým požadavkům. V případě, kdy se firma rozhodne vytvořit si IS svépomocí, odpadá i jakákoliv závislost na dodavateli. Pokud se však má jednat o větší systém, tak se závislost na dodavateli zvyšuje, protože jde o zakázku pro specializovanou společnost. [29]

Druhým způsobem obstarání informačního systému je zakoupení hotového řešení, které do jisté míry požadovaným parametrům a funkcím vyhovuje. I tento způsob má své výhody a omezení. Ve srovnání s vyvíjeným IS jde rozhodně o levnější a časově méně náročnou variantu získání IS do společnosti. Dalším kladem je skutečnost, že hotový systém už má ověřenou a ošetřenou funkčnost na mnoha předchozích instalacích. Mezi nevýhody takového řešení je v prvé řadě fakt, že dodaný systém nebude plně odpovídat potřebám firmy (ať už funkčně nebo organizačně). Další nevýhodou může být obtížnost integrace s aplikacemi třetích stran, které již firma využívá, nedostatečná nebo žádná lokalizace, zastaralost architektury nebo technologie

daného systému. Závislost na dodavateli IS je srovnatelná s předchozím případem protože úspěšnost implementace systému závisí především na něm. [29]

Třetí a poslední možnost, jak pořídit požadovaný informační systém je formou pronájmu IS, jako služby. Poskytovatel ASP (Application Service Provider) je firma, která zákazníkovi požadovaný informační systém pronajímá prostřednictvím zabezpečeného internetového připojení. V krátkém období se jedná o velmi levné řešení získání IS, které je ještě umocněno rychlostí implementace do prostředí daného podniku. Podobně jako u hotového IS ani zde nemohou být splněny veškeré požadavky firmy. Obrovskou nevýhodou, jež od tohoto řešení odrazuje mnoho firem, je velmi vysoká závislost na dodavateli IS v kombinaci s ohrožením bezpečnosti dat. [19]

Tabulka 1 - Způsoby získání IS (převzato z [19])

Faktor	Vyvíjený IS	Hotový IS	ASP IS
Cena	↑	↓	→
Čas	↑	↓	↓
Přizpůsobení požadavkům firmy	↑	↓	↓
Závislost na dodavateli	→	→	↑

2.3 Životní cyklus IS

Tvorba jakéhokoliv informačního systému by měla probíhat v určitých základních krocích, díky nimž se kvalita výsledné aplikace může jen zvyšovat. Jak bude uvedeno dále, tak některé kroky jsou často v různých metodikách potlačeny a některé naopak vyzdvihovány. Bez ohledu na zvolenou metodiku by se ale tyto činnosti během vývoje IS měly vyskytovat.

Některé ze zmíněných částí životního cyklu IS lze bez pochyby využít i v jiných oblastech než je IT. Prakticky pokaždé, než člověk začne něco vyrábět, musí si uvědomit, co to vlastně má být a zda bude výsledný produkt vůbec k něčemu užitečný. Dalším důležitým faktem, může být cena, za kterou bude moci být daný produkt vyroben (např. židli do jídelny nebude vhodné vyrábět z mahagonového dřeva, které by jistě svými vlastnostmi vyhovovalo požadavkům, ale jeho pořizovací cena je pro daný účel neúměrně vysoká). Všechny tyto zmíněné aspekty lze převést i do procesu tvorby IS, kde se dají nazvat **předběžnou analýzou**. U informačního

systému se vedle ceny řeší i možnost jeho navázaní na blízké okolí (integrovatelnost), základní funkčnosti, rozsah atd. Jde o nejobecnější pohled na celý připravovaný projekt.

Obsahem druhé části životního cyklu informačního systému je podrobnější rozbor všech poznatků zjištěných v části předchozí. Vedle důkladnějšího rozpracování časových plánů a nákladů se jedná i o detailnější specifikaci vlastností, hardwarových a softwarových požadavků, identifikaci uživatelů, stanovení základních pravidel pro zpracovávání dat v systému atd. Kvalitní provedení této **analýzy systému** je pro další fáze projektu velmi zásadní a ve výsledku může ušetřit mnoho času a peněz na úpravy nedostatků zjištěných v pozdějších chvílích.

Všechny podstatné informace vyplývající z analýzy navrhovaného systému jsou sepisovány v dokumentu, který slouží jako hlavní podklad pro vytvoření závazné smlouvy se zákazníkem (výroba na zakázku). **Projektová studie** by měla obsahovat veškeré smluvně dohodnutelné podmínky jak pro dodavatelskou, tak i pro odběratelskou firmu:

- Základní informace o dodavateli
- Základní informace o zákazníkovi (+ kontaktní a kompetentní osoby)
- Popis současného stavu informačních technologií zákazníka
- Obecný návrh IS – logický datový model (návrh funkcí a dat systému bez ohledu na technologické prostředí)
- Detailní návrh IS – fyzický datový model (funkční analýza systému, datová analýza, popis veškerých datových toků v organizaci a popis funkcí řízených událostmi)
- Popis průběhu nasazování produktu v praxi (včetně specifikací SW a HW)
- Popis průběhu testování systému, včetně poskytování záručního servisu
- Harmonogram spolupráce (termíny dodávek, platby, ceny, podmínky dodání atd.)

Tento dokument by měl být tvořen s ohledem na to, že management zákazníka, který má zakázku v kompetenci, nemusí být natolik zasvěcen do problematiky IT, aby porozuměl technickým termínům a pojmům.

Podle podmínek sjednaných mezi oběma smluvními stranami se provádí **implementace**. Jde o proces, v němž leží hlavní váha celého projektu. Měl by být prováděn specialisty a kontrolován analytiky, kteří tvoří jakousi spojnici mezi požadavky zákazníka a reálným produktem. Samotné programování se provádí podle

všech zjištěných informací v dosavadním průběhu projektu. Implementace zahrnuje vedle programování jednotlivých funkčních bloků i jejich propojení a následně i propojení s dalšími (zpravidla zákazníkovými) systémy – tzv. integrace. Výsledkem by tak měla být hotová aplikace obsahující požadovanou funkčnost a vlastnosti, které si zákazník přál.

S postupem času, kdy se začal vývoj softwaru uplatňovat na poli masivní produkce, dostává se ke slovu stále více **testování**. Developerské firmy si začaly uvědomovat, že jedinou cestou, jak vypustit do světa kvalitní produkt je jeho dostatečné testování. Tato činnost má za úkol ověření funkčností a splnění vytyčených požadavků na daný systém. Tím, že se nedostatky odhalí a mohou tak být opraveny ještě dříve než se software dostane do ostrého provozu, jeho hodnota pro budoucího uživatele výrazně roste i za cenu toho, že se mu produkt do rukou dostane později.

Ve chvíli, kdy je systém připraven pro předání do provozu k zákazníkovi, přichází ke slovu fáze **zavádění systému**. Je to fáze, při níž se celá aplikace instaluje k zákazníkovi k běžnému používání. Během instalace je pokaždé nutná konfigurace systému tak, aby mohl bez potíží fungovat v novém prostředí, než byl doposud. Vedle nasazování se během této fáze kompletují dokumentace a manuály a probíhají školení koncových uživatelů.

Samotné zavádění systému lze řešit několika způsoby [19]:

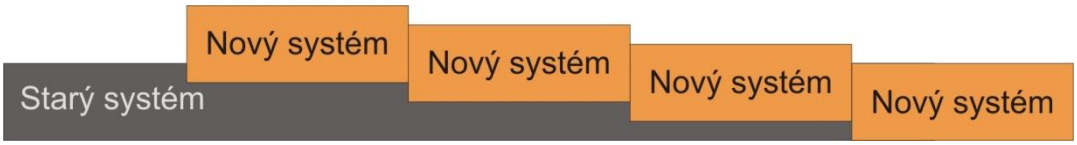
- Při použití *souběžné strategie* jsou po nějakou dobu v provozu oba systémy (původní i nový) současně. Tato strategie je náročná především na kapacity.

Obrázek 1 - Souběžná strategie zavádění IS (převzato z [19])

- Při uplatnění *pilotní strategie* se systém zavede a vyzkouší v jednom oddělení podniku, po odsouhlasení je zaveden v celém podniku.

Obrázek 2 - Pilotní strategie zavádění IS (převzato z [19])

- *Postupná strategie* doporučuje zavádění informačního systému po jednotlivých modulech.



Obrázek 3 - Postupná strategie zavádění IS (převzato z [19])

- Ve výjimečných případech lze využít i *nárazovou strategii*, kdy je původní systém vystřídán novým takřka bez přechodové fáze.

Obrázek 4 - Nárazová strategie zavádění IS (převzato z [19])

Když je systém úspěšně nasazen u zákazníka a je používán v běžném každodenním provozu, tak by měla probíhat předem sjednaná lhůta, kdy je dodavatel povinen zajišťovat okamžitý servis. Ten může být nutný tehdy, když se během **zkušebního provozu** vyskytne problém nebo dokonce chyba, jež nebyla objevena a odstraněna během testování. Takové chyby by se měly vyskytovat velmi zřídka a mělo by jít o chyby, které nejsou pro celou aplikaci nebo její část nijak kritické.

Ukončením projektu je období, kdy je systém používán v **rutinním provozu.** Zásahy dodavatele se projevují jenom údržbou a zajištěním bezchybného běhu aplikace. Často jsou v této fázi zahrnuty servisní linky, na které je možné v případě závady nebo nesnází uživatelů zavolat a nechat si svůj problém vyřešit proškoleným personálem.

Fáze, která u projektů ani nemusí nastat je tzv. **reengeneering.** Ve chvíli, kdy zákazník zjistí, že mu již dodaná aplikace nepostačuje nebo nevyhovuje a náprava není možná pouhou úpravou některých částí, může sepsat nové požadavky a může se tak zahájit nový projekt. Tímto krokem se životní cyklus IS dostává opět na samý začátek k předběžné analýze.[36]

Tabulka 2 - Fáze životního cyklu IS

1	Předběžná analýza (specifikace cílů)
2	Analýza systému (specifikace požadavků)
3	Projektová studie (návrh)
4	Implementace

5	Testování
6	Zavádění systému
7	Zkušební provoz
8	Rutinní provoz a údržba
9	Reengineering

Jak již bylo zmíněno, tak tento seznam činností v rámci životního cyklu je pouhým nástinem toho, jak by se mělo při reálných projektech postupovat. V další kapitole budou uvedeny některé postupy vývoje softwaru (informačních systémů), které se s uvedenými činnostmi ztotožňují, další si tyto činnosti poupravují a některé jejich počet dokonce redukují.

Vedle základních metodik, které buď v praxi nelze použít vůbec, nebo jenom na malé projekty – často v osazenstvu do tří osob, jsou zmíněny i postupy, které lze aplikovat na velké projekty s lidmi s různou specializací.

2.4 Metodiky vývoje SW

Pro co nejvyšší efektivitu práce projektového týmu, je zapotřebí aby všichni jeho členové ctili a uznávali stejná pravidla a metodiky. Nejlepší možností, jak jít správnou cestou je jít po stopách celosvětově rozšířených a ověřených postupů. Mezi takové postupy patří bezesporu metodiky vývoje softwaru RUP, metodologie PRINCE2, z níž takové postupy mohou vycházet, nebo systém řízení služeb IT ITIL, který se částečně také zabývá metodikou pro vývoj.

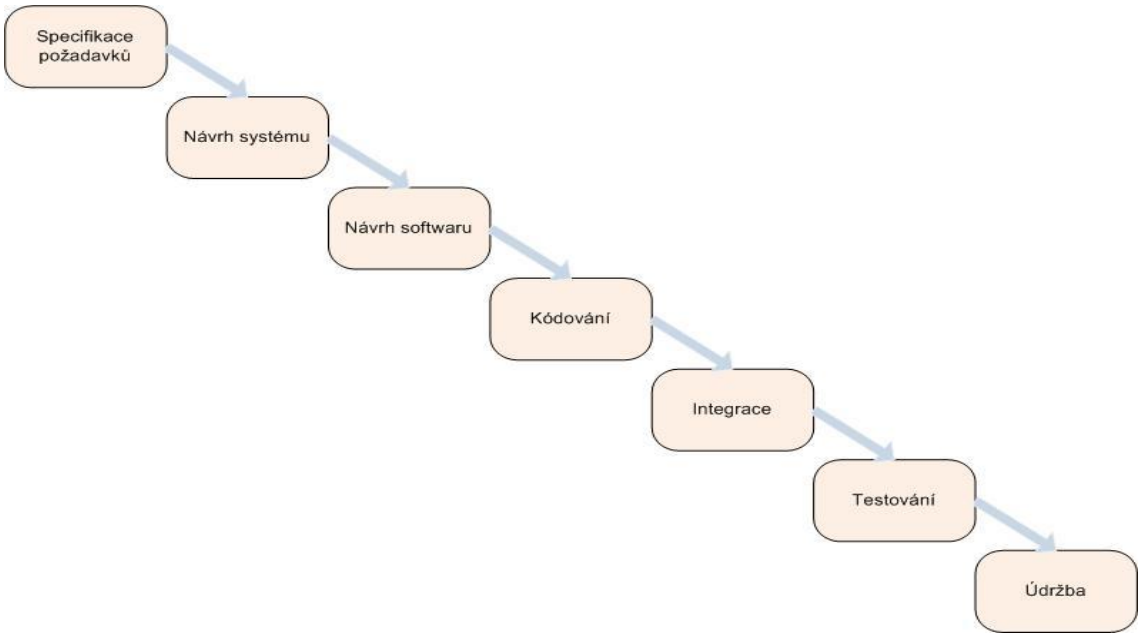
V následujících odstavcích budou jednotlivé metodiky a postupy rozepsány ve svých detailech, kladech a nevýhodách jejich použití u projektového řízení. Některé tyto metodiky byly rovněž využity jako inspirace pro tvorbu nové metodiky, kterou se společnost Unicorn Universe při svých projektech řídí. [21]

2.4.1 Waterfall

V roce 1970 publikoval Winston W. Royce článek „Managing the Development of Large Software Systems“, v němž popisuje nejjednodušší model přístupu k řízení v systémovém inženýrství – vodopád. I přes Royceovo vlastní zpochybňování

použitelnosti tohoto modelu v praxi se stal vodopád velmi rozšířeným a spousta dalších metodik z něj buď vychází, nebo jej alespoň využívá. [33]

Model vodopádu se skládá ze sedmi základních činností, které provázejí každý větší projekt vývoje softwaru.



Obrázek 5 - Waterfall model (převzato z [34])

Specifikace požadavků: veškeré požadavky na funkčnosti a vlastnosti systému by měly být zjištěny v tomto kroku. Požadavky se zjišťují od uživatele a analytik má za úkol rozlišit, jaké požadavky jsou splnitelné a zda jsou zadány vyčerpávajícím způsobem. Výsledkem této činnosti by měl být dokument, sloužící jako vstup pro následující fázi modelu.

Návrh systému: systém by měl být podrobně navržen ještě před zahájením implementace. Návrh zahrnuje architekturu, která definuje a popisuje hlavní části a komponenty včetně integrace. Dále jsou stanoveny požadavky na hardware (platforma, operační systém atd.). Rovněž software musí splňovat požadavky uživatele. Každá z těchto činností bude ve výsledku sepsána do dokumentů, které opět poslouží jako vstupní materiál pro zahájení práce v dalších činnostech.

Návrh softwaru: základní architektura, která definuje hlavní bloky systému, bude v této části rozebrána na jednotlivé moduly. Jsou detailněji popisovány funkce

a interface jednotlivých modulů. Pro samotnou implementaci bude vytvořena dokumentace, která bude popisovat vlastnosti a chování softwaru.

Kódování: podle dokumentace vytvořené v minulém kroku je zahájeno kódování, jenž se ze začátku zaměřuje hlavně na menší části budoucího softwaru (units). Tyto části k ověření své funkčnosti zatím nepotřebují další moduly a budou integrovány ke zbytku aplikace později.

Integrace: moduly vytvořené v předchozím kroku lze samostatně otestovat pomocí tzv. unit testů. Tím se ověří, zda splňují očekávanou funkčnost, která byla zjištěna během analytické části vývoje. Po otestování dílčích částí, jsou tyto propojeny pomocí připravených rozhraní, čímž je vytvářen kompletní výsledný software.

Testování: po finálním sestavení všech modulů dohromady je software otestován co do komunikace mezi jednotlivými moduly (integrační testy), tak i co do požadavků stanovených na začátku. Tato činnost již musí probíhat na hardwaru, stanoveném v definici architektury.

Údržba: kompletní software je předán uživateli (zákazníkovi), jenž jej poprvé použije. Uživatel by si měl zkontrolovat, zda je vše implementováno podle očekávání (včetně správnosti prvotních požadavků). V případě, že uživatel usoudí, že je nutné provést další změny kvůli použitelnosti softwaru, nebo doplnění některých funkčností, přesouvá se projekt do nikdy nekončící fáze údržby. Všechny problémy, které se vyskytly během některé z předchozích fází projektu, se v této části vyřeší (dříve či později). [33] [34]

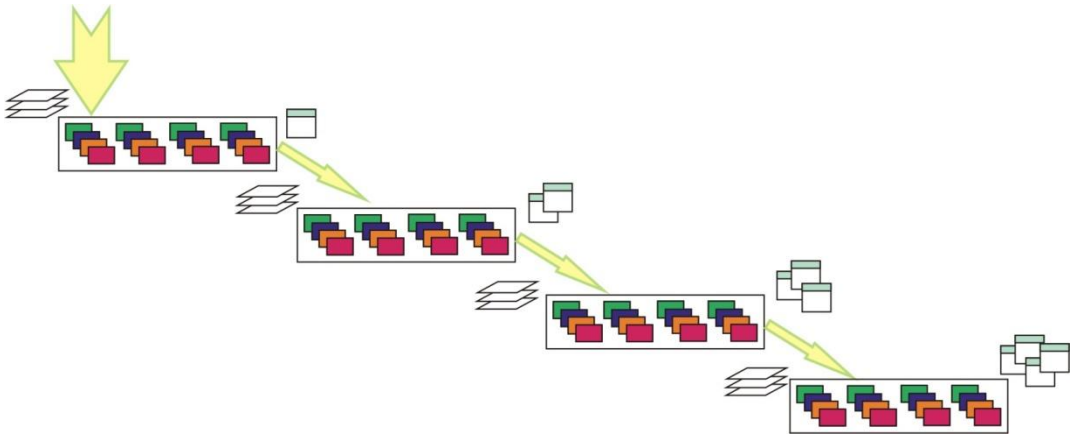
Nejdůležitější vlastností celého modelu je, že k započetí následné činnosti musí být předchozí činnost kompletně dokončena (není možný návrat o krok zpět). V tom tkví i největší nevýhoda modelu. Při nesprávně stanovených požadavcích (neúplné analýze) může být tento nedostatek odhalen až při konečném spuštění hotového systému. Když je zapotřebí zjištěný nedostatek napravit, musí se začít téměř od začátku novou analýzou, dalším implementováním a opětovným testováním. Předchozí práce tedy víceméně ztratila význam a projekt se spouští prakticky znovu. Vedle času stráveného na nezdařilé implementaci se rovněž rapidně navyšují náklady na celý projekt.

Na rozsáhlé projekty je tedy metoda sama o sobě naprosto nepoužitelná. Pokud však má být menší projekt alespoň nějak řízen, tak waterfall s opatrností a důsledností použít lze.

2.4.2 Iterativní model

Poněkud paradoxně se může jevit odstranění hlavní nevýhody vodopádového modelu v modelu iterativním. Odstranění spočívá ve vícenásobném použití metody waterfall po sobě – v iteracích.

Uživatel tak má možnost průběžně kontrolovat jak jeho aplikace roste a v případě jakéhokoliv nedostatku zasáhnout do vývoje a požadavky na software upravit nebo upřesnit. Tyto úpravy se mohou promítnout v některé z následujících iterací. Stejně tak se uživatelské požadavky stále zpřesňují a konkretizují, což zabraňuje předání softwaru, jenž je naprosto odlišný od očekávaného výsledku.



Obrázek 6 - Iterativní model

Na druhou stranu to, že se požadavky mohou měnit i v průběhu zahájeného vývoje dává prostor k laxnosti při jejich analýze. Tento fakt může v souhrnu vést k horšímu návrhu systému než u metody vodopádu.

Tím, že se testování vyvíjené aplikace provádí v průběhu každé iterace, je mnohem vyšší pravděpodobnost, že bude nalezeno více chyb – výsledná aplikace bude kvalitnější. [25]

2.4.3 Spirálový model

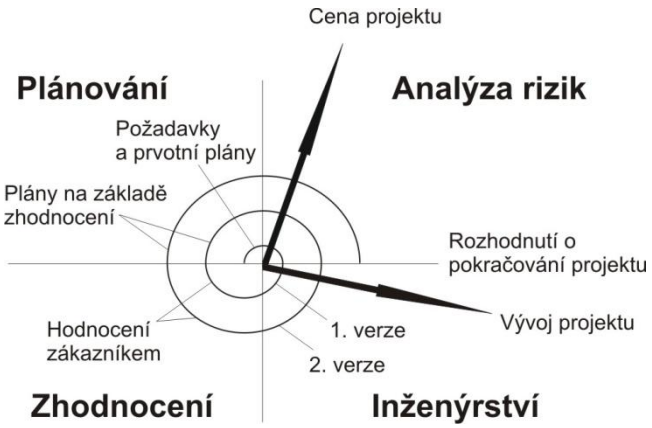
Již z názvu spirálového modelu je zřejmé, jak podle něj vývoj probíhá. Jedná se o opakování jednotlivých cyklů vývoje, přičemž při každém dalším oběhu po spirále je problematika více zvládnutá a přidávají se tak další funkčnosti. Důležitou vlastností modelu je, že vedle prototypového přístupu, kdy se zákazníkovi dostane do rukou funkční část celého softwaru, dává velký důraz na analýzu rizika.

Model je rozdělen do čtyř fází (kvadrantů), které se opakují tak dlouho, dokud není software dokončen. První z fází je plánování, kde v prvním cyklu jsou sestaveny požadavky na software a vytvoření plánu celého vývoje. V dalších cyklech jsou požadavky upravovány podle představ zákazníka a následně jsou plány adekvátně upraveny.

Ve druhé fázi se provádí odhady ceny za hotové řešení a analyzují se rizika, která by mohla během vývoje nastat. Obě tyto činnosti se liší podle toho, zda jsou prováděny v úvodním cyklu (vychází se z prvotních požadavků), nebo v některém z dalších cyklů (na základě zákazníkem upravených požadavků).

Na základě provedené analýzy rizik se rozhoduje, zda v projektu pokračovat nebo ne. Toto rozhodnutí se prování na rozmezí mezi druhou a třetí fází vývoje.

Fáze inženýrství se vyznačuje kódováním a následným vytvořením prototypu vyvíjeného softwaru. Sestavený prototyp je předložen zákazníkovi na jeho zhodnocení a poukázání na případné odchylky mezi obdrženým prototypem a jeho představami. Dosažené závěry jsou v dalším cyklu zanalyzovány a na jejich základě je upraven plán pro další fáze projektu.



Obrázek 7 - Spirálový model (převzato z [6])

2.4.4 Agilní metodologie

Další metodika, podle níž lze vytvářet software je v porovnání s ostatními zmiňovanými poněkud alternativní. Vznikla v roce 2001 jako pokus o nalezení nové cesty pro efektivní vývoj zaměřený na rychlost, kvalitu a cenu. Tato metodika má za cíl odlehčit vývoj od těžkopádné dokumentace, která projekty prodražuje a odvádí pozornost od komunikace se zákazníkem. [13]

Samotný vývoj probíhá v kratších iteracích, kterých je ale více než např. u obyčejného iterativního modelu. Každá iterace se skládá z návrhu, implementace, testování a oprav nalezených chyb.

Vzhledem k rychlosti průběhu jednotlivých iterací je pochopitelné, že nové funkcionality softwaru přibývají po malých částech, což skýtá značnou výhodu v tom, že implementovaná část může být ihned konzultovaná se zákazníkem, čímž dochází k okamžitému ověření správnosti implementace.

Návrh spočívá v časté a pravidelné komunikaci se zákazníkem, který jasně definuje své požadavky. Tyto požadavky jsou následně převedeny do kódu, nad kterým mohou být provedeny testy. V případě nalezení nedostatků ve funkčnosti se provede oprava. Výsledek iterace se předkládá zákazníkovi a ten ověří správnost a pochopení zadání.

U agilních metod se naprosto zanedbává klasická dokumentace, která je k vidění u ostatních postupů vývoje softwaru. Tato dokumentace je nahrazena samotným kódem, jenž musí být o to více přehledný. Přehlednost kódu se zabezpečuje několikanásobným refaktoringem více lidí. [38]

Projekty založené na postupech agilních metod jsou charakteristické stanovením dostupných zdrojů a času, který bude na vývoj potřebný. Tyto faktory jsou neměnné po celou dobu trvání projektu. Výsledek projektu (propracovanost a rozsah) však dopředu určen není. V tom tkví největší rozdíl oproti klasickým přístupům, kde je vždy na začátku projektu stanoven rozsah výsledné aplikace a teprve na základě této informace se stanovuje doba trvání a náklady na realizaci (tyto dvě veličiny se většinou v průběhu projektu mění). Protože si tento typ přístupu ale komerční společnost nemůže dovolit, tak se tato metodika využívá především u open source aplikací. [30]

2.4.5 RUP

Rational Unified Process je metodika (framework) pro vývoj softwaru vytvořená společností Rational Software Corporation v roce 1987. Metodika RUP vychází z kolekce osvědčených praktik a postupů (tzv. Best practices) při vývoji softwaru. Metodika je sice použitelná pro jakýkoliv rozsah projektu, ale je zapotřebí ji přizpůsobit konkrétním potřebám daného projektu. Jedná se o druh iterativního vývoje – na konci každé z fází by tedy měla být zákazníkovi k dispozici verze softwaru, podle které bude moci sledovat postup prací a směr kterým se práce ubírají. [32]

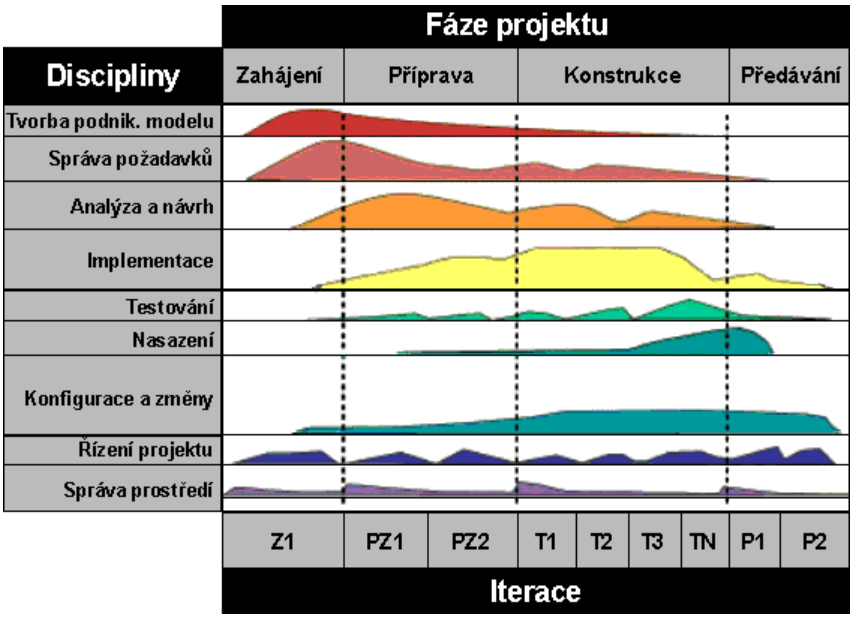
Základem metodiky Rational Unified Process je šest základních pravidel (disciplín), tzv. Best practices, jenž vyšly najevo z rozsáhlých průzkumů mezi managementem projektových týmů po celém světě [32]:

1. Iterativní vývoj software
2. Správa požadavků
3. Komponentová architektura
4. Vizuální modelování
5. Ověřování kvality software
6. Řízení změn software

Iterativní vývoj

V rozsáhlém systému již není možný postupný průběh jednotlivých činností, kde se na začátku celý problém definuje, poté se navrhuje řešení, provádí se implementace a na závěr se celý systém otestuje.

Iterativní přístup spočívá v rozdělení celého projektu na čtyři fáze (zahájení, příprava, konstrukce a předávání), z nichž každá se skládá z několika iterací. Počty iterací mohou být v každé fázi různé a různě časově náročné. Na konci každé ze čtyř fází by měla být k dispozici odpovídající verze softwaru, která může být konzultována se zákazníkem.



Obrázek 8 - Iterativní přístup (převzato z [3])

Během každé iterace jsou činnosti rozděleny do devíti disciplín:

1. **Tvorba podnikového modelu** – Cílem je převod požadavků zákazníka do jazyka srozumitelného pro vývojáře – sestavení specifikací.
2. **Správa požadavků** – Požadavky musejí být řízeny, musejí přicházet v logických sekvencích.
3. **Analýza a design** – Návody a postupy pro implementaci a testování.
4. **Implementace** – Převedení modelu do spustitelného kódu a provedení prvního kola testování.
5. **Testování** – Provedení objektivního zhodnocení dosažené kvality vyvíjeného softwaru. Tím je myšleno nalezení chyb, kontrola, zda systém pracuje tak, jak má nadefinováno ve specifikacích a ověření, že jsou splněny předem dané požadavky.
6. **Nasazení** – Příprava systému k nasazení a k užívání pro koncové uživatele
7. **Konfigurace a změny** - Zajištění přístupů jednotlivých členů (skupin) k adekvátním zdrojům informací. Řízení shromažďování informací a také jejich správa (kontroly, aktualizace, odstraňování).
8. **Řízení projektu** – Řídí aktivity celého projektového týmu tak, aby byly úspěšně splněny požadavky zákazníka v zadaných termínech a v zadaném rozpočtu. Projektový management se také vypořádává s mírou rizika, kterou na sebe vzal přijetím zakázky.

9. **Správa prostředí** – Podpora projektového týmu zajištěním potřebných prostředků, návodů, lidí s potřebným know-how atd.

Iterativní vývoj má následující výhody:

- Projekt může být objektivně posouzen
- Je umožněno rovnoměrnější pracovní vytížení vývojového týmu.
- Testování se může provádět již v průběhu projektu na tzv. meziverzích
- Projektový tým může být v kontaktu se zákazníkovými uživateli a ihned tak dostávat zpětnou vazbu
- Veliká šance ušetření nemalých nákladů za úpravy produktu provedené v pozdní fázi projektu (především průběžným testováním)
- Jednodušší možnost zapracování změn požadavků

Aktivní správa požadavků

RUP využívá aktivní správu požadavků, která spočívá v neustálých konzultacích zadavatele a dodavatele. Konzultace mají za cíl upřesňovat a doplňovat požadavky zadavatele na finální produkt i v průběhu projektu.

Komponentová architektura

Komponenty označují netriviální části systému zajišťující jeho určitou funkcionalitu. Využití komponentové architektury systému je výhodné z několika důvodů:

- Projekt lze snáze rozdělit mezi několik vývojových týmů
- Zapracování změn je snazší a snadněji identifikovatelné
- Je možný postupný vývoj systému (nejprve jsou vyvinuty např. komponenty na back-endu, později se přechází k vývoji front-endu)

Vizuální modelování

Grafické znázornění systému a jeho chování zpřehledňuje návrh a umožňuje udržet konzistenci mezi modelem a vlastní implementací.

Pro vizuální modelování se v metodice RUP využívá jazyka UML.

UML (Unified Modeling Language) je grafický jazyk využívaný pro vizualizaci, snazší specifikaci, návrh a dokumentaci programových systémů a aplikací. Výhodou

UML je to, že nabízí standardizovaný způsob zápisu jak návrhů systému včetně konceptuálních prvků jako jsou business procesy a systémové funkce, tak konkrétních prvků jako jsou příkazy programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty.

UML rovněž podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů. Je tedy vhodný k použití v prostředí, kde se využívá objektově orientovaných programovacích jazyků (Java, C# apod.). [37]

Ověřování kvality software

Softwarový produkt je možné zákazníkovi předat pouze v případě, že splňuje předem určená kvalitativní kritéria. Tato kritéria jsou zpravidla zaměřena na správnou funkcionalitu (podpora funkcí vymezených pomocí use case specifikací), spolehlivost (správné řešení chybových stavů) a výkon (odezvy systému jsou v přijatelných mezích).

Hlavním procesem pro zajištění výše uvedených parametrů je testování.

Řízení změn

Během vývoje nového informačního systému může nastat okamžik, kdy je potřeba např. zavést novou funkcionalitu nebo doplnit specifikaci stávající funkcionality. Možnost provádět takovéto změny i v pozdějších stádiích projektu s sebou samozřejmě přináší i problém jejich řízení a dokumentace.

2.4.6 ITIL

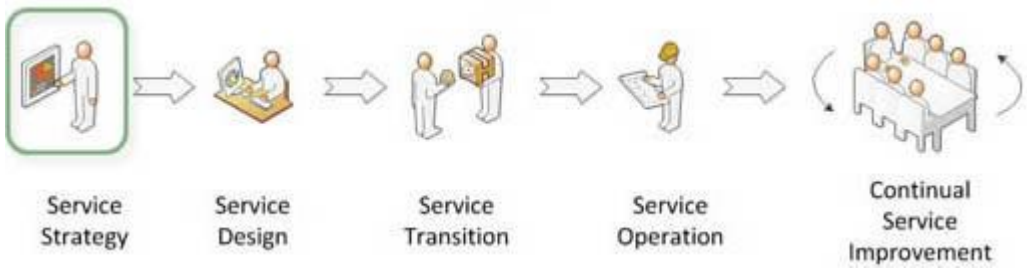
Nejlepší používané praktiky pro správu IT služeb v podnicích jsou shrnuty a sepsány v knihovně ITIL (Information Technology Infrastructure Library). Jde o framework, který zastřešuje zajišťování IT služeb, jejich měření a neustálé zlepšování jejich kvality.

První verze ITIL vznikala v letech 1989 až 1995 na popud britské vlády. Agentura CCTA (Central Communications and Telecommunications Agency) tehdy dostala za úkol vytvořit metodiku, pomocí níž by bylo možné zavést pořádek v IT ve státních úřadech. Takto vzniklo 31 knih různých návodů, postupů a doporučení pro poskytování služeb IT.

V letech 2000 až 2004 byla vydána druhá verze ITIL, v níž byly poznatky z první verze redukovány, revidovány a doplněny. Výsledkem bylo sedm velmi těsně

souvisejících knih, které byly přijaty mnoha organizacemi po celém světě jako základ pro efektivní poskytování služeb IT. [19]

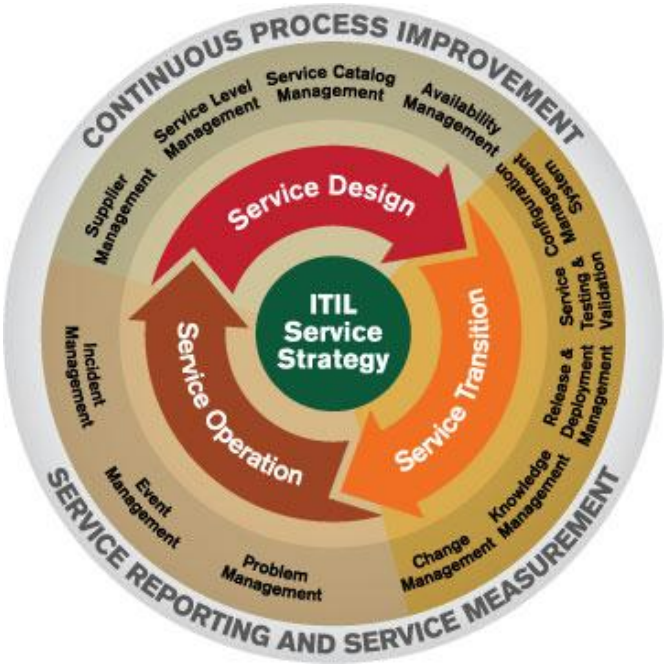
Dosud nejaktuálnější verze frameworku ITIL vznikla v roce 2007. Jedna se o tzv. ITIL V3. V této verzi se opět podařilo redukovat počet knih na pět základních. Nejdůležitějším aspektem ITIL V3 je pohled na služby IT jak z pohledu zákazníka, tak i z pohledu dodavatele (business) těchto služeb. Proto se podařilo poměrně rychle tento přístup rozšířit do celého světa, kde získává stále větší podporu a popularitu. Mezi nejčastěji zmiňované přínosy zavedení souboru metodik ITIL patří zvýšená spokojenost se službami IT (jak businessu, tak i zákazníků), zkrácení času pro uvedení nových produktů a služeb na trh, finanční úspory, které plynou ze snížení opakovaných prací, zlepšené správy a využití zdrojů atd. [16]



Obrázek 9 - Životní cyklus služby IT podle ITIL v3 (převzato z [1])

Stejně jako u předchozích metodik, tak i rámec ITIL respektuje svou vlastní interpretaci životního cyklu, jehož jednotlivé fáze jsou rozepsány právě ve zmíněných základních pěti knihách. Celý životní cyklus se skládá z počáteční definice v rámci strategie (Service strategy), návrhu služeb (Service design), přechodu služeb do živého prostředí (Service transitition), samotném provozu služeb (Service operation) a jejich neustálé zlepšování (Continual service improvement). [19]

Vedle pěti základních knih existuje i řada dalších publikací o dodatečných službách a procesech, které se během životního cyklu mohou využívat.



Obrázek 10 - ITIL (převzato z[17])

Service strategy

První kniha ITIL tvoří základní osu mezi zákazníkem a poskytovatelem služeb. Zaměřuje se na prvotní identifikaci potřeb zákazníka a řeší, zda a jakou službu vůbec zákazník potřebuje.

Pro knihu Service strategy není klíčová otázka jak službu zavést a používat, ale proč.

Všechny knihy rámce ITIL obsahují několik základních procesů. Pro první knihu jsou to následující čtyři [15].

Strategy management for IT services (strategická správa služeb IT) – Proces odpovědný za definici a udržování plánů a charakteristických znaků organizace s ohledem na její služby a správu těchto služeb.

Service portfolio management (správa portfolia služeb) – Proces odpovědný za správu portfolia služeb – poskytovatel služeb zajišťuje zákazníkovi správnou kombinaci služeb, které odpovídají požadovaným podnikovým výstupům při přijatelné úrovni vynaložených investic.

Financial management for IT services (správa financí pro služby IT) – Procesy týkající se stanovení odpovídajících cen za poskytované služby.

Business relationship management (správa vztahů s businessem) – Proces, který je odpovědný za udržování pozitivních vztahů dodavatele s businessem. Jsou identifikovány potřeby zákazníka, na jejichž základě je vybírán vhodný katalog služeb.

Service design

Druhá kniha se zabývá podobou nově zaváděné nebo pozměněné služby, která by měla vyhovovat jak zákazníkovi, tak i dodavateli. Dále by měla zabezpečovat to, aby bylo možné po uvedení služby do reálného prostředí ji okamžitě využívat (poskytovat). Vedle architektury služby se tak musí řešit i její zapracování do organizace jako takové. Pro zajištění splnění veškerých požadavků na tuto fázi životního cyklu lze při zavádění postupovat podle následujících bodů:

- Návrh služby a jejích funkčních požadavků
- Návrh správy dodávky služby (podmínky poskytování, propojení s dalšími službami)
- Návrh architektury (příprava služby k používání)
- Návrh procesů pro zajištění provozu služby
- Návrh metrik a způsobů jejich měření

Splněním všech bodů se dosáhne vytvoření dokumentace (Service design package) k celé dodávané službě, která bude sloužit jako podklad pro fyzickou tvorbu dané služby. [19]

Kniha Service design v sobě zahrnuje čtyři základní procesy [15]:

Supplier management (správa dodavatelů) – Proces zajišťující to, že zákazník skutečně dostane od dodavatelů očekávanou hodnotu za peníze. Žádná smlouva není v rozporu s potřebami zákazníka, a všichni dodavatelé dostávají svým závazkům.

Service catalogueue management (správa katalogu služeb) – Proces, který odpovídá za katalog služeb – jeho údržbu a dostupnost.

Service level management (správa úrovní služeb) – Proces, který odpovídá za zajištění a plnění dosažitelných dohod o úrovních služeb (SLA).

Availability management (správa dostupnosti) – Odpovídá za dostupnost služby podle potřeb zákazníka. Dostupnost musí odpovídat vynaloženým nákladům a zároveň současným i budoucím cílům businessu

Někdy se uvádějí ještě další tři procesy, jimiž jsou Capacity management, Information security management a IT service continuity management.

Service transitition

Třetí kniha ITIL je o převedení návrhu v Service design package do skutečného produktu – služby. Veškeré součásti vytvořeného návrhu se testují, vyhodnocují, dokumentují a připravují k předání do užívání. Výsledkem této fáze je předvedení a předání výsledné služby zákazníkovi. [19]

Knowledge management (správa znalostí) – Odpovídá za manipulaci se znalostmi a za jejich dostupnost (využívá znalostní báze).

Change management (správa změn) – Proces odpovědný za řízení životního cyklu všech změn (zavedení změn bez narušení dalších služeb).

Release and deployment management (správa releasů a nasazení) – Proces odpovědný za plánování, načasování a integrace, testování a nasazení služby.

Configuration management system (systém správy konfigurací) – Nástroje, které slouží k podpoře aktiv služeb a správy konfigurací. Je součástí celkového systému správy znalostí o službách a zahrnuje i informace o incidentech, problémech, známých chybách, změnách atd.

Service validation and testing (validace a testování služby) – Odpovědný za provedení validací a testování nově zavedené služby. [15]

Service operation

Ve čtvrté knize je pozornost soustředěna na provozní aktivity, které následují po úspěšném zavedení daných služeb do procesů firmy zákazníka. Jde zejména o pravidelnou údržbu a zajištění takové dostupnosti a stability, aby byla zachována hodnota nakoupených služeb.

Charakteristickým prvkem je pro tuto fázi životního cyklu tzv. Service desk, který slouží zejména jako zprostředkovatel komunikace mezi zákazníkem a poskytovatelem (dodavatelem) služby. [19]

Třemi základními procesy jsou [15]:

Event management (správa událostí) – Zahrnuje odpovědnost za správu událostí během jejich životního cyklu.

Incident management (správa incidentů) – Správa incidentů zajišťuje, aby byl v případě jejich vzniku obnoven normální provoz služby tak rychle, jak je to možné (s minimalizací dopadů na zákazníka).

Problem management (správa problémů) – Zamezuje výskytu incidentů.

Continual service improvement

Cílem poslední knihy ITIL je identifikace a následná optimalizace těch služeb, jejichž vylepšení je pro zákazníka přínosné a které jim pomáhají v konkurenčním prostředí obstát.

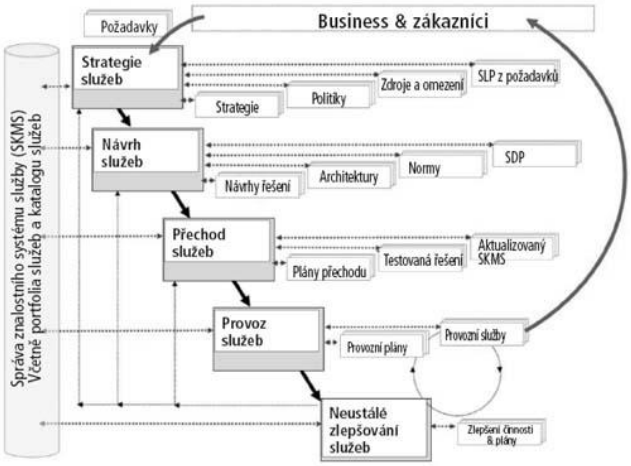
Proces neustálého vylepšování se řídí podle Demingova cyklu (PDCA), který se skládá ze čtyř základních kroků [19]:

Plan – naplánování konkrétních změn (zlepšení) a strategie, jak jich efektivně dosáhnout

Do – Naplánované zlepšení se převede ve skutečnost pomocí stanoveného rozpočtu, uživatelů v jednotlivých rolích, stanovení jejich odpovědností, vytvoření dokumentace provedených prací atp.

Check – Kontrola implementovaných zlepšení je prováděna vyhodnocením porovnání plánů a skutečnosti na základě předem stanovených kritérií. Bez reálného splnění stanovených cílů nelze přejít k další fázi.

Act – Inovace je převedena do běžného užívání a jsou zavedena opatření, která zabraňují v tom, aby se dané zlepšení nevyužívalo a nedošlo se do stavu před jejich zavedením.



Obrázek 11 - Souvislosti v rámci ITIL (převzato z [16])

2.4.7 RUP vs. ITIL

Při pohledu na metodiku RUP a soubor knih ITIL člověka může napadnout jistá podobnost jejich jednotlivých částí, ze kterých se skládají. Proto by neměl být žádný problém v tom, že by se oba postupy využívaly najednou. Mohly by se tak navzájem doplňovat a společnými silami odstraňovat nedostatky toho druhého. [18]

Následující tabulka ukazuje, že jednotlivé fáze obou postupů jsou víceméně ekvivalentní a přitom činnosti, kterými disponuje RUP, nejsou pokaždé obsaženy mezi činnostmi v ITIL (a naopak).

Tabulka 3 - Spolupráce RUP a ITIL

RUP		ITIL	
Zahájení	Případy užití (20%)	Service strategy	Identifikace požadavků
	Slovník projektu		Zdroje a napojení
	Počáteční plán projektu		Právní ujednání
	Obchodní případ		Strategie
	Odhad rizik		
	Prototyp		
Příprava	První spustitelný základních	Service design	Balíčky modelů služeb
	Model UML		Standardy
	Popis architektury aplikace		Architektura
	Vývojový plán projektu		Modelová řešení
	Předběžný uživatelský manuál		
Konstrukce	Spustitelný software	Service transiton	Aktualizované balíčky modelů služeb
	Testovací sada		Testovací řešení
	Uživatelské příručky		Plány nasazení
Předávání	Aplikace		
	Uživatelské příručky		
	Plán podpory		
		Service operation	Provozní plány
			Provoz služeb
		Continual service improvement	

3 Analýza současného stavu

Životní cyklus projektů stavěných na základech metodiky RUP bývá rozdělen do čtyř základních fází, z nichž je každá ukončena milníkem. Milníkem je myšleno zhodnocení a určení splněných cílů během dané fáze života projektu. Následná fáze by měla být zahájena pouze v případě splnění všech kritérií požadovaných ve fází předcházející.

3.1 Zahájení

Tato fáze má za úkol vymezení požadavků na systém, na němž se shodnou obě zainteresované strany – zákazník i dodavatel. Je zapotřebí vymezit rozsah vytvářeného systému a pro klíčové činnosti vytvořit modely případů užití. Projekty, při nichž se nevytváří nový systém, ale pouze se upravuje systém stávající, je tato fáze podstatně méně náročná, protože výše uvedené kroky byly již dříve zčásti provedeny. [2]

3.1.1 Úkoly specifické pro fázi zahájení projektu

Tvorba podnikového modelu: Základní charakteristika podniku

Business analytik má popsat pomocí modelů případů užití procesy, které se týkají vytvářeného systému. Dále má nastínit rozdělení oblastí use case modelů podle toho, jak spolu UCs souvisí.

Správa požadavků: Analýza problémové oblasti

Systémoví analytici by měli spolupracovat se zadavatelem a dohodnout zaměření a rozsah systému. Analytici by měli být dostatečně proškoleni jak v businessových záležitostech zákazníka, tak i v technologiích, které budou pro konstrukci systému použity – měl by rozeznat, co z požadovaných funkčností není možné použitou platformou splnit.

V této chvíli již může být nastíněna první podoba modelu případů užití.

Správa prostředí: Příprava prostředí projektu

Specialista na vývojové nástroje zvolí prostředky pro analýzu požadavků. Hlavní systémový analytik vytvoří první verzi pravidel pro modelování případů užití (jak mají vypadat specifikace, co mají a nemají obsahovat atd.).

Řízení projektu: Plánování vývoje software

Management má za úkol sestavit úvodní rozpočet pro následující fáze projektu. Plán by měl být sestaven i pro vývoj systému. V úvodní části by mělo být určitě možné sepsat seznam rizik, která projektu hrozí – k tomuto kroku je již potřebná zkušenost projektových manažerů.

Je jasné, že odhady vytvořené v této části projektu jsou pouze orientační a budou v následujících fázích zpřesňovány a doplňovány.

3.1.2 Milník - cíle projektu

Na konci první fáze projektu nastává důležitý milník (milestone), který rozhodne o jeho dalším pokračování.

- Zákazník se s dodavatelem musí shodnout na harmonogramu a předběžně stanovené ceně za dodaný systém.
- Z revize výstupních dokumentů zákazníkem musí vyplynout, že dodavatel pochopil zadání a je schopen jej splnit.
- Odhady rizik, nákladů a dob provádění musí být reálné (nepřikrášlené nebo naopak zbytečně nenavyšované).

V případě, že nebudou splněny tyto podmínky, je třeba rozhodnout se o dalším postupu. Jednou z možností je přepracování a přehodnocení plánů, druhou je úplné zastavení projektu.

3.2 Příprava

Druhá fáze má za cíl navržení stabilního modelu architektury, který by měl posloužit jako základ pro implementaci. Protože fáze přípravy následuje po fází zahájení projektu, je velmi důležité mít zpracovanou úvodní verzi plánu vývoje systému obsahující i plán jednotlivých iterací pro tuto fázi. [2]

3.2.1 Úkoly specifické pro fázi přípravy projektu

Řízení projektu: Plán iterace

Projektový manažer vytvoří plán pro druhou iteraci na konci předchozí fáze. Softwarový architekt doplní popis kritérií, podle nichž bude iterace hodnocena. Může se zde i doplňovat seznam rizik.

Správa prostředí: Příprava prostředí pro iteraci

Potřeba nakonfigurovat vývojové nástroje a prostředky a vytvořit šablony dokumentů pro analýzu a návrh.

Správa požadavků: Stanovení prioritních případů užití

Systémoví architekti a projektoví manažeři mají za úkol vybrat ty případy užití, které se budou v této iteraci zpracovávat. Mělo by se jednat zejména o ty části systému, které určují podobu architektury. Na základě tohoto seznamu se dané případy užití musí zpracovat co možná nejdetailněji, aby bylo možné zahájit jejich vývoj.

Analýza a návrh: Návrh a přidělení subsystémů

Softwarový (systémový) architekt má za úkol zpřesnit definici architektury pro potřeby zvoleného implementačního prostředí. Jednotlivé případy užití jsou přiděleny vývojářům.

Testování: Určení obsahu testů

Manažer testů zjistí, jaké části systému budou testovatelné již v této iteraci. Následně se zabývá návrhem způsobů testování.

Implementace: Integrace systému

Integrátor provede postupnou integraci dříve vytvořených subsystémů do podoby funkčního prototypu.

Testování: Ověření stability systému

Provádění testů stávajícího systému. Na základě výsledků tohoto testování se mohou doplnit postupy pro testování další.

Řízení projektu: Ukončení a zhodnocení iterace

Se závěrem druhé iterace může projektový manažer provést její hodnocení. K tomu mu poslouží porovnání dosažených výsledků s dříve sestavenými plány. Vedle stavu vyvíjené aplikace by měl zhodnotit i stav nákladů a splňování termínů. Pokud se ukážou nepřesnosti v dosavadním průběhu prací, musí se plány pro další iterace doplnit a upřesnit (včetně plánu vývoje a seznamu rizik).

3.2.2 Milník - architektura

Druhý milník projektu nastává na konci druhé fáze projektu. Projekt by v tomto okamžiku měl splňovat následující body:

- Vize produktu je již pevně daná a neměnná.
- Do této chvíle bylo identifikováno více než 80% z celkového počtu případů užití a zároveň byla alespoň polovina z nich analyzována.
- Alespoň 10% případů užití má již hotový návrh a je naimplementováno.
- Návrh architektury je již konečný a nebude docházet k zásadním změnám.
- Jsou definovány postupy pro hodnocení a testování vytvářeného systému.
- Testování prvních verzí aplikace potvrdilo, že byly správně identifikovány rizikové oblasti.
- Je vytvořen a doplněn plán pro fázi konstrukce, tak aby podle něj mohl vývoj pokračovat.
- Zadavatel souhlasí s realizací vývoje systému za použití navrhované architektury a plánu vývoje.
- Skutečná spotřeba zdrojů (náklady, čas atd.) nepřevyšuje plán.

I v této fázi se musí zhodnotit celkový stav projektu a zvážit jeho pokračování. Ve chvíli, kdy hrubým způsobem nejsou splněny předchozí body, projekt může být zastaven.

3.3 Konstrukce

Třetí fází projektu je konstrukce. Jde o nejnáročnější část projektu a jejím cílem je dokončení návrhu a vytvoření a otestování první kompletní verze systému. Na rozdíl od předchozích dvou fází je tato zaměřena převážně na implementaci. Projekt je rozdělen na několik nezávislých, přidělených jednotlivým vývojářským týmům.

Požadavky na systém by mely být stabilní a neměnné (radikální změny). Mělo by být možné zapracovávání změnových a dodatečných požadavků od zákazníka. [2]

3.3.1 Úkoly specifické pro fázi konstrukce projektu

Řízení projektu: Plán iterace

Projektový manažer má za úkol naplánovat a určit seznam funkcionalit, na nichž se během této iterace bude pracovat.

Implementace: Naplánování integrace systému

Plánování integrace určuje, kdy a jaké moduly vytvářeného systému budou předány k testování.

Testování: Tvorba scénářů systémových testů

Test designer by měl na základě use case specifikací vytvořit postupy, které umožňují ověřit splnění požadavků na funkcionalitu systému.

Implementace: Implementace komponent

Vývojáři mají za úkol implementaci jednotlivých komponent, které si i zároveň otestují (unit testy). Tyto testy mají prokázat, že se daná komponenta při určitých vstupních datech (správných i chybných) chová správně.

Implementace: Integrace subsystémů

V předešlém kroku vytvořené komponenty se použijí pro tvorbu subsystémů.

Testování: Tvorba scénářů integračních testů

Test design postupů, jež umožní ověřit úroveň integrace jednotlivých podsystémů.

Testování: Testování a hodnocení

Na jednotlivé subsystémy se aplikují systémové testy. V případě nalezení chyb ve funkčnosti systému, jsou tyto zaevidovány a nahlášeny.

Implementace: Integrace systému

Po dostatečném testování subsystémů má integrátor za úkol tyto subsystémy poskládat do jednoho celku. Skládání se děje postupně aby bylo možné průběžným testováním odhalit případné nesrovnalosti.

Testování: Testování a hodnocení

Celý systém je otestován z hlediska integrace. Všechny podsystémy by spolu měly komunikovat podle určených pravidel a přes přesně daná rozhraní.

Celý systém by měl být funkčně otestován. V případě nalezení chyb ve funkčnosti systému, jsou tyto zaevidovány a nahlášeny.

Řízení projektu: Ukončení a zhodnocení iterace

Projektový management opět zhodnotí využití vývojových nástrojů a dodržování finančního a časového plánu.

Konec třetí fáze by se měl nést především v duchu testování.

3.3.2 Milník - první funkční verze

Fáze konstrukce by měla skončit poskytnutím funkční verze a schopností systému být předán. Měla by být vytvořena aktuální dokumentace a uživatelský návod. Produkt by měl splňovat následující kritéria:

Systém je natolik stabilní, aby mohl být předán uživatelům.

Zadavatel má připravenou půdu pro nasazení systému.

Pakliže produkt nesplní některou z uvedených podmínek, mělo by se uvažovat o odložení předání aplikace o jednu verzi. V tuto chvíli již není příliš reálné projekt zastavit (pokud tedy není v úplně kritickém stavu), protože už do něj bylo investováno příliš mnoho prostředků.

3.4 Předávání

Jak je patrné z názvu, tak poslední fází projektu je předání finálního produktu zákazníkovi. Během této fáze se provádí beta testování a jsou upravovány už pouze dílčí detaily, které byly během testování odhaleny. V této fázi už by nemělo docházet k markantnějším změnám ve funkcionalitě. [2]

3.4.1 Úkoly specifické pro fázi předávání

Řízení projektu: Plán iterace

Projektoví manažeři by měli rozhodnout o dalším vývoji. Ten by měl pokračovat především na základě změnových požadavků zákazníka.

Nasazení: Beta testování

Před oficiálním předáním hotového produktu se doporučuje dát uživatelům možnost otestovat jeho beta verzi. Na základě jejich připomínek by se měla verze určená pro předání dotvořit, tak aby byl zákazník spokojen.

Analýza a návrh, Správa požadavků

V této fázi projektu by se už požadavky neměly měnit a systém by se měl už pouze odlaďovat. V případě, že se objeví ještě další potřeba změny, může se tato provést.

Implementace

Vývojářská činnost by se měla zaměřovat především na opravování nalezených chyb během beta testování, případně starších. Jedna část vývojářského týmu se může stále ještě věnovat implementaci změnových požadavků. V této fázi projektu už by se rozhodně neměly objevovat nové požadavky, které by mohli rapidně změnit pohled na celý systém.

Testování

Testování by se mělo provádět několikrát dokola. V ideálním případě by měla být vhodná doba pro kompletní přetestování celého softwaru. Veškeré opravené chyby by měly být ověřeny a pokud možno uzavřeny. Veškeré činnosti by měly směřovat k zahájení přejímacích testů.

Nasazení: Přejímací testy

Zákazníkův tým má provést testování, na jehož výsledcích bude záviset možnost dokončení předání aplikace. K provedení testování může zákazník použít jak vlastní testovací scénáře, tak i scénáře, které mu poskytne dodavatel – scénáře, podle nichž probíhalo systémové testování.

Vlastní dodávka hotového systému

Výsledný software by měl být zkomprimován a měl by z něj být vytvořen instalační balíček, pomocí kterého bude možné aplikaci zprovoznit u zákazníka. Společně s instalací by měla probíhat i konfigurace všech souvisejících systémů, aby byla aplikace schopna fungovat v ostrém provozu.

Ve stejnou chvíli by měly být dokončeny a nachystány k předání veškeré uživatelské manuály a dokumentace.

Školení

Veškerý školící materiál by mě být připraven pro zahájení školení uživatelů z řad zákazníkových zaměstnanců.

3.4.2 Milník - předání

Poslední milník znázorňuje ukončení vývoje dané verze systému, který je předán zákazníkovi a je nasazen do užívání. Úspěšnost projektu se posuzuje podle spotřebovaných zdrojů a spokojenost uživatelů, kterým má systém sloužit.

Pro dodavatele systému nastává období podpory. Podpora spočívá v tom, že během času, kdy je dodaný systém nasazen u zákazníka, vyplynou návrhy na zlepšení některých jeho částí.

Podle toho, jaké připomínky se objeví v ostrém provozu, se zákazník s dodavatelem mohou dohodnout buď na nové verzi celého systému (nový projekt) nebo se připomínky zapracují pouhými úpravami stávajícího systému.

4 Současný stav zadané problematiky

Firma si zakládá na tom, aby projekty, jež jí jsou přiděleny, byly tvořeny na základech metodiky UESPC. Ta má základy v metodikách ITIL, PRINCE2, RUP a dalších. Už od počátečních okamžiků projektu je tak vidět vliv této metodiky na celý projekt.

Je samozřejmé, že jak RUP, tak i další metodiky, z nichž UESPC vychází, nelze použít na jakýkoliv projekt beze změny. Pokaždé je zapotřebí do jisté míry poupravit průběh jednotlivých fází a jejich iterací tak, aby bylo možné v projektu postupovat. Neměly by se ale vyskytovat diametrálně odlišné postupy a závěry.

Lidé, kteří vedou projektové týmy podle takových metodik, jako je RUP by ve svých řadách měli mít i člověka, jenž si už dané postupy mohl osahat a projít si s nimi několika úspěšnými projekty. Zkušenosti, které tento člověk dokázal nasbírat v minulosti, mohou být pro současný tým velkým přínosem, a mohou pomoci řešit různé nástrahy mnohem efektněji a efektivněji než kdyby se jimi zabývali pouze nováčci v daném oboru.

4.1 Projekt

Zákazník vyhlásil výběrové řízení s cílem najít takového partnera, který by byl ochoten a schopen vytvořit zcela nový informační systém, jímž by chtěl nahradit systém současný, poněkud těžkopádný a necustomizovatelný pro potřeby celosvětově rozšířených poboček.

Zakázka byla získána společností Unicorn Systems, a.s. na základě kvalitně zpracovaných návrhů řešení, relativně nižších nákladech a časových plánech.

Na následujících stranách budou shrnuty postupy, tak jak se odehrávaly během trvání projektu. Postupy při vývoji softwarové aplikace budou rozděleny do čtyř fází projektu – v souladu s metodikou RUP. Jednotlivé fáze jsou odděleny tzv. milníky, jenž představují úspěšně dosažené dílčí cíle.

4.1.1 Zahájení

Práce na úvodní studii pro projekt byla zahájena 28. března 2011. Náročnost projektu byla dána především rozsahem a termínem dokončení, který byl projektovým vedením vyjednán se zákazníkem. Termín nasazení aplikace přímo u zákazníka (Go live) byl stanoven na 17. října 2011.

Na průběh celého projektu dohlíželi dva garanti z řad zákazníka. Přímo pod garanty projektu byl postaven člověk odpovědný za úspěšnost implementace a dodávky navrhovaného systému. V úvodních dvou částech projektu se zákazníkovy zaměstnanci podíleli na běhu projektu pouze externě formou konzultací a schůzek s projektovým managementem a analytiky.

Celý projektový tým byl veden hlavním projektovým manažerem, který společně se zbytkem nejužšího vedení týmu komunikoval se zákazníkem a vytvářel rámcovou představu o výstupu projektu. Plně v jeho kompetenci bylo dovedení projektu do zdárného konce.

Na základě jednání managementu se zástupci zákazníka byl sestaven předběžný časový harmonogram a rozpočet celého projektu.

Během první fáze byla sestavena personální kostra celého týmu. Jednalo se o pozice, jež jsou nezbytné od samého počátku projektu. Šlo tedy o další projektové manažery, business analytika, systémové analytiky, systémové integrátory a konfigurátora. Zároveň byly vyhledány vhodné kancelářské prostory, které by měly sloužit po dobu celého projektu (případně i po něm).

První fáze projektu se nesla v duchu nastiňování základního obrysu zadavatelem. Konzultacemi s business analytiky bylo dosaženo rozdělení systému na několik hlavních oblastí, v nichž byly představeny jednotlivé funkční požadavky. Tyto informace mají později posloužit jako podklad pro tvorbu úvodní studie projektu.

Milník, zachycující pochopení zadání dodavatelem a odsouhlasení navrhované ceny zákazníkem byl splněn – proto bylo oboustranně rozhodnuto o pokračování projektu.

4.1.2 Příprava

Druhá fáze projektu byla charakteristická bližším specifikováním požadavků na systém. Zákazník připravoval seznamy komponent, které dodá pro integrování a pro souběh s navrhovanou aplikací.

Proběhlo proškolení analytiků na problematiku business procesů a seznámení s technologiemi, které budou využity pro konstrukci. Na základě školení a doposud domluvených funkčností a vlastností navrhované aplikace byly případy užití rozděleny do devíti základních oblastí. Konzultace systémových a business analytiků s konzultanty zadavatele budou probíhat minimálně během této a následující fáze projektu.

Jednotlivé funkčnosti jsou systémovými analytiky sepisovány do samostatných specifikací. Na základě těchto specifikací byla sepsána úvodní studie. Dne 30. 5. 2011 byla úvodní studie uzavřena a akceptována. Tím vznikl smluvní závazek mezi oběma společnostmi.

Během revize doposud sepsaných analytických specifikací zákazníkem, byla zjištěna jejich nedostatečná propracovanost a nepoužitelnost pro předání do vývoje. Z tohoto důvodu bylo nastaveno několik základních pravidel pro tvorbu use case specifikací a modelů. Následně byla vytvořena první verze UC modelu a bylo blíže specifikováno, jaké technologie budou pro tvorbu aplikace využívány (softwarové nástroje vývojářů).

V úvodu druhé fáze projektu byla vypsána výběrová řízení na pozice juniorských vývojářů a testerů. Po skončení výběrových řízení bylo spuštěno školení pro vybrané vývojáře, kteří se tak měli seznámit s technologiemi vývoje a postupy, jež měli následující měsíce využívat. Toto školení trvalo čtrnáct dní a po jeho skončení, bylo odstartováno týdenní školení testerské (testerský tým měl pochopitelně na projekt nastoupit později než vývojáři).

Vedle zmíněných junior developerů byli na pomoc s projektem osloveni i senioršti vývojáři, kteří se měli postavit do čela jednotlivých vývojářských týmů, případně měli na starost integrační a architektonické záležitosti.

Co se týče managementu, tak byl sestaven seznam rizik, s nimiž se počítalo již od začátku. Dále byla vytvořena pracovní verze plánu vývoje, v níž byly základní představy o délce trvání a rozpočtu pro následující fáze projektu.

Na samotném projektu se v této době odehrávala konfigurace vývojových nástrojů, prostředí a prostředků potřebných ke konstrukci.

Systémoví architekti, business analytici a projektoví manažeři společně se zástupci zákazníka zvolili stěžejní případy užití, jež se začnou vyvíjet jako první. Šlo o use cases, které svým způsobem zachycují hlavní funkcionality systému, ale také o ty, jež byly nejjednodušší (kvůli následné optimalizaci vývojářských týmů). Těmto vybraným částem systému se poté začali detailněji věnovat systémoví analytici. Ti měli za úkol co nejpodrobněji popsat funkce v dokumentech nazývaných use case specifikace (UCS). V mezičasech, které sloužily k revizím specifikací business analytikem a následně i zákazníkem, analytici pracovali na zbylých případech užití. Před skončením druhé fáze měli mít analytici více či méně podrobně rozepsané všechny prvky připravovaného systému (v závislosti na aktuální důraz na požadovanou funkcionalitu).

Vzhledem k požadavkům analytiků se stále měnil logický datový model, který je zásadní částí architektury celého systému. Změny se však netýkaly základní vize projektu, ale měly pouze minoritní charakter. To ve chvíli, kdy ještě neodstartoval vývoj, není velký problém.

S přiblížením konce druhé fáze projektu skončilo i vývojářské školení a vývojáři se tak mohli pustit do zmapování případů užití a na základě pokynů jednotlivých team leaderů zahájit vývojovou činnost. Testerské školení trvající jeden týden následně mohlo začít.

Na základě seznamu případů užití vybraných pro první fázi vývoje, testový architekt zvolil způsoby, jakými se bude testovat a vytvořil šablony, podle nichž se bude testování provádět.

Koncem fáze přípravy projektu byla naplánována společná (zástupci zadavatele i dodavatele) off-site revize doposud sepsaných UC specifikací.

Následující odstavce zahrnují několik bodů, které lze označit za milník pro ukončení druhé fáze projektu.

Architektura systému se zdá být navržena – její podoba je odsouhlasena i zástupci zákazníka.

Projektoví manažeři vytvořili plán vývoje, podle nějž může začít konstrukce aplikace.

Vize produktu je jasně definovaná a nemění se. Všichni členové projektového týmu jsou si vědomi, co je jejich úkolem a co bude výsledkem jejich snažení. Stejně tak každý jednotlivec zná klíčové termíny, které byly stanoveny na řídícím výboru a všem delegovány na projektových schůzích.

Z celkového počtu 163 případů užití je v tuto chvíli identifikováno celých 140, přičemž 122 z nich má i svou specifikaci. Naimplementováno je pouhých několik málo jednoduchých („cvičných“).

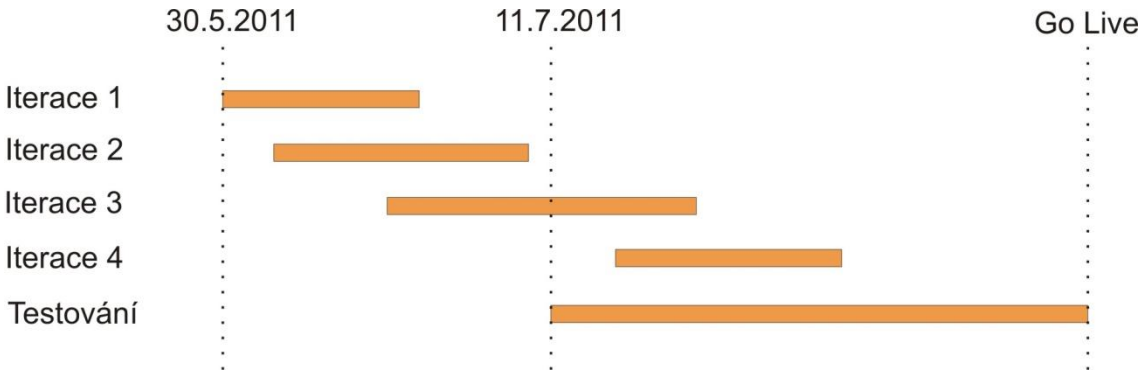
Zákazník odsouhlasil realizaci vývoje (akceptace úvodní studie), s tím že doposud nebyl překročen plánovaný rozpočet a časový harmonogram.

4.1.3 Konstrukce

Bezpochyby nejnáročnější a nejdéle trvající fáze projektu byla odstartována s koncem května roku 2011.

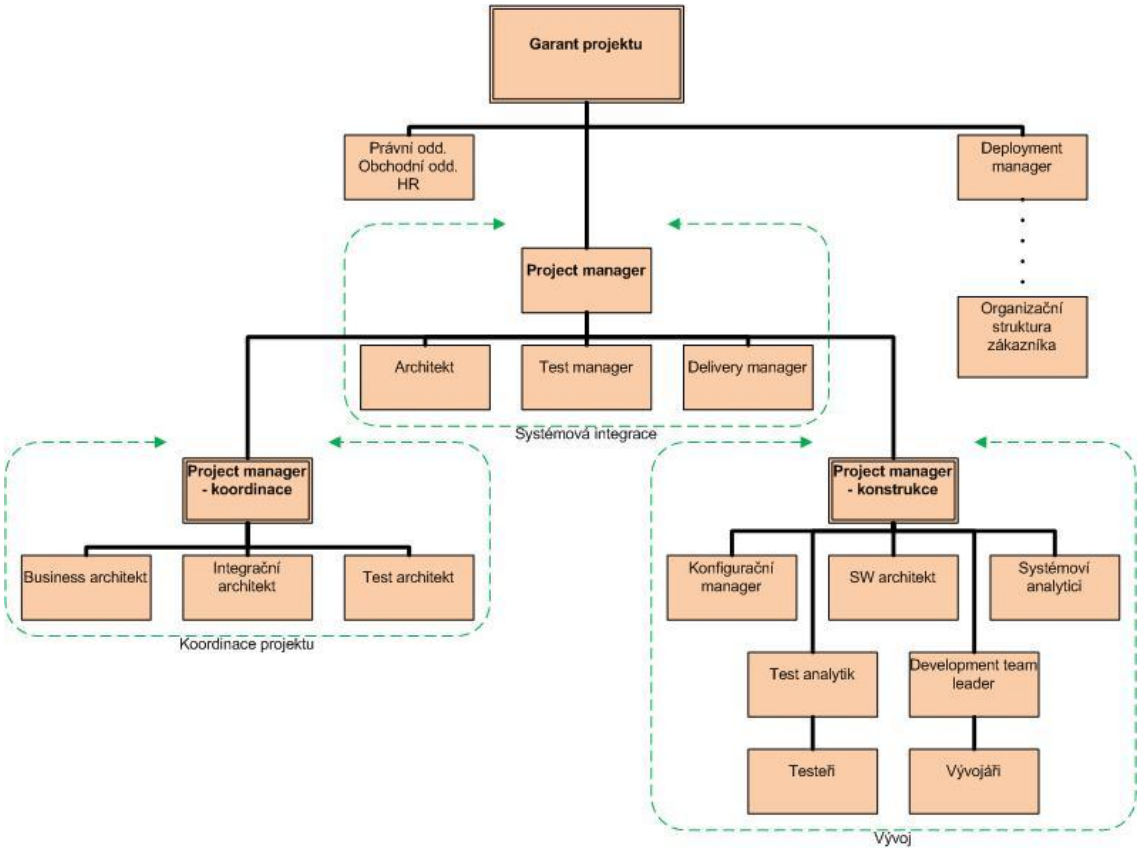
V tuto chvíli již byl vývojářský tým proškolen a byl plně k dispozici pro vývoj. Testeři se do procesu zapojili zhruba o 14 dní později. Management podle nastalé situace upravil podobu organizační struktury (viz. Obrázek 13).

Projektoví manažeři představili plán, v němž je nastíněn průběh kompletního vývoje. V souladu s metodikou RUP se vývoj má odehrávat v rámci několika iterací (plánovaný průběh hovoří o čtyřech iteracích).



Obrázek 12 - Plán průběhu fáze konstrukce

Plán byl vytvořen tak, aby se během první iterace pracovalo na těch částech systému, které měly spojitost s integrací. Druhá iterace se nesla v duchu vývoje komponent, jež byly pro celý systém klíčové. Třetí naplánovaná iterace se zaměřovala na případy užití se střední prioritou a čtvrtá měla za cíl vytvořit menší, z pohledu zákazníka nejméně důležité moduly. Během každé iterace probíhaly systémové testy, podle nichž bylo možné určovat kvalitu odvedené práce a rychlost progresu. Položka testování se tak týká plánů integračních, end-to-end (E2E) a uživatelsky akceptačních testů (UAT). Jak bylo řečeno již dříve, tak termín Go Live byl stanoven na 17. 10. 2011.



Obrázek 13 - Doplněná organizační struktura

Hlavní vedoucí vývojářů předvedl návrh na vytvoření tří týmů developerů, z nichž každý tým měl na starosti jinou část aplikace (jiné činnosti).

Byly identifikovány moduly, které budou implementovány a zároveň bylo určeno v jakém pořadí (s jakou prioritou).

Na základě těchto informací začal testový analytik tvořit první postupy, podle nichž se bude moci ověřit správnost funkcionality částí systému. Testovací scénáře

se se píší především za použití use case specifikací a konzultací se systémovými analytiky. Zároveň probíhají také první přípravy na E2E a UAT. Vyjasňují se jejich termíny zahájení, které závisejí také na dodávkách potřebných komponent od samotného zákazníka. Tyto body má plně v kompetenci testový manažer, který rovněž řeší jaké prostředky a technologie budou pro testování a následný reporting chyb použity.

Vývojáři tvoří jednotlivé komponenty – začíná se komponentami, které se nejvíce týkají architektury (jsou přímo propojeny s databází) a které jsou zároveň stěžejní pro úspěšnou integraci. Jeden z týmů se již také zabývá front – end komponentami (části aplikace, které jsou viditelné pro uživatele). Každý svůj výsledek práce před odevzdáním k integraci musí vývojář otestovat sám (unit testy). Teprve po tom, co je přesvědčen o správné funkčnosti dané části, může ji předat do dalšího zapracování.

Součásti, odevzdané vývojáři jsou pomocí jejich rozhraní spojovány do jednoho celku – subsystému. Touto činností se zabývají zkušení senioršti vývojáři – integrátoři.

Jednotlivé subsystémy, jež jsou postupně vytvářeny, se musejí průběžně testovat, což je systematický proces prováděný podle scénářů systémových testů. Tvorba testovacích scénářů byla předána do kompetence testerů. Zároveň se vytvářejí nové testovací scénáře pro integrační a E2E testování, která budou také vykonávána dodavatelem. Jediným druhem testování, které vykonávají sami zaměstnanci zákazníka je UAT.

Ve druhé polovině července si vývojáři i testeři ztěžují na nedostatky a nejasnosti v analýze. Proto se musí use case specifikace neustále upravovat a odlaďovat tak zachycené chyby (konzultace se zákazníkem stále probíhají). Po každém zanesení změny do specifikací případů užití se tyto změny musí zanést do již vytvořené komponenty a stejně tak do testovacích scénářů. V tomto okamžiku nastává problém v komunikaci uvnitř projektu. Analytici nepředávají vývojářům žádnou informaci o tom, že byla provedena změna v některé ze specifikací. Tyto informace se k vývojářům dostávají až od testerů, kteří stále pracují na testovacích scénářích a jejich aktualizacích.

Testový manažer se po dohodě na řídícím výboru rozhodl pro variantu hlášení chyb nalezených při testování pomocí komplexního defektního nástroje JIRA (bude

využíván i zákazníkem v době UAT). Ve spojení s testovacími scénáři tvořenými v tabulkovém procesoru (typu MS Excel), však tento postup není příliš efektivní.

Ve druhé třetině července přichází zákazník s prvním změnovým požadavkem, který po zpracování dopadové analýzy a odhadů pracnosti odsouvá termín Go Live na 31. 10. 2011. Zpracovávání zmíněných dokumentů má vliv i na dodávku UC specifikací, která se tak začíná opožďovat. Změnový požadavek si rovněž vyžádal prodloužení fáze vývoje na pět iterací. To do značné míry souvisí i se zpožďováním dodávek hardwaru, na kterém by mělo probíhat uživatelsky akceptační testování. Vedle termínu Go Live se tak posunuly i termíny pro zahájení UAT a E2E testů, přičemž E2E testy by navíc měly trvat o týden déle oproti původnímu plánu.

Na konec srpna bylo dohodnuto zahájení školení zástupců zákazníka na dosud vytvořené verzi aplikace. Školení by mělo mít za cíl umožnění přístupu zaměstnanců z řad zákazníka a následně ověřit jejich pochopení funkčností celé aplikace. Školení má probíhat přímo na pobočce dodavatele aplikace, tudíž je zapotřebí připravit několik školitelů a vyhradit odpovídající prostory (učebny). Zákazník doposud nezačal pracovat na scénářích pro UAT, jehož testeři by měli být rovněž přítomni na školení. Tato informace v době tří týdnů před plánovaným zahájením školení není nijak povzbudivá.

Integrační testování zatím probíhá podle plánů, ale vyskytuje se velké množství neprůchozích testů. Vedle toho jsou E2E testy pro první tři iterace vývoje zahájeny.

Celý projektový tým plně soustředí svou činnost na problematiku plánovaného školení, protože z vyjádření zákazníka vyplynuly jeho pochybnosti nad současnou kvalitou aplikace a její celková připravenost. Tyto pochybnosti byly podloženy především nízkou průchodností systémových testů. Pro systémové testy je v tuto chvíli připraveno zhruba 75 % testovacích scénářů, přičemž všechny testy se provádějí přes jednotný uživatelský účet (nejsou zohledněny uživatelské role).

I přes vysoké nasazení celého projektového týmu se stále nedaří uvést aplikaci do takového stavu, aby bylo možné zahájit školení. Tento problém vznikl jak na straně zákazníka, který včas nedodal potřebný hardware, tak i na straně dodavatele, kterému se neustále vracejí blokující chyby (cca 25 % testovacích scénářů systémových testů), nebo jsou úspěšnými opravami zavlečeny chyby nové. Projektový management tak musel oznámit zákazníkovi zpoždění systémových a E2E testů minimálně o týden.

Zákazník v této nelehké situaci navíc oznámil změnu v architektuře, která by měla být dodržena. Změna se týkala toho, že jedna databáze nebude sloužit všem integrovaným systémům najednou, ale že každý ze systému bude postaven na své vlastní instanci databáze. Z toho důvodu bylo potřeba změnit strukturu datového modelu, což má za následek velmi výrazný rozpad dosavadní funkčnosti aplikace.

Skutečnosti z několika předchozích odstavců mají také za následek odkládání zahájení UAT. Tomu nakonec bude předcházet jakási fáze preUAT (pro zabezpečení zprůchodnění kriticky důležitých scénářů), která je nyní naplánována na polovinu září. Do té doby by měly být implementovány i vybrané změnové požadavky.

Školení, které mělo podle původních plánů trvat 5 dní, bylo nakonec přesunuto až na 10. října, přičemž by mělo trvat celých 10 dní. Navíc se situace změnila natolik, že zaměstnanci (budoucí uživatelé) nepřijedou na pobočku dodavatele, ale školitelé poletí do zahraničí, konkrétně do Indie. Na stejný termín bylo potvrzeno i zahájení fáze UAT. K tomu, aby mohlo školení a UAT začít bylo zapotřebí naplnit databáze relevantními testovacími daty, jejichž parametry zákazník do poslední chvíle nedodal. Jednalo se o poměrně zdlouhavý proces, protože bylo rozhodnuto, že data se nemohou do databáze nalévat dávkově, ale že budou vkládána již pomocí formulářů aplikace. Sice se jednalo o časově náročnější variantu, ale na druhou stranu díky opakovanému přístupu do aplikace se mohla nalézt řada dosud nezjištěných chyb.

Zástupci dodavatele pracují na přípravách školících materiálů a plány pro průběh školení, které je následně spuštěno.

Další úprava časového harmonogramu se týkala mezního termínu pro spuštění zkušebního provozu na zahraniční pobočce zákazníka. Tento termín byl garanty projektu stanoven na 28. 11. 2011. Pro zajištění dostatečné rezervy projektoví manažeři stanovili datum pro nasazení aplikace do zkušebního provozu na 21. 11. 2011. Z toho vyplývá, že akceptace zákazníkem musí proběhnout do tohoto data, tzn., výzva pro akceptaci musí být předložena alespoň 7 dní předem.

Většina vývojářů se v této fázi již orientuje na opravu nalezených chyb. Zbývající členové vývojového týmu se zaměřují na implementaci změnových požadavků. Testování se provádí denně na několika interních verzích aplikace. Zákazníkovi

jsou nové verze poskytovány pro UAT na denní bázi. Během listopadu probíhaly už i performance testy, které simulují používání aplikace větším množstvím uživatelů.

I když se do konce listopadu podařilo nasadit aplikaci do produkčního prostředí, do poloviny prosince se nepodařilo podepsat ani jeden kontrakt (stěžejní funkcionalita aplikace), což ale nebylo způsobeno chybou dodaného softwaru.

Zákazník dodává nové změnové požadavky, které očekává v pilotní verzi aplikace.

To, že školení na dodané verzi aplikace proběhlo úspěšně a že se následně podařilo nasadit ji i do produkčního prostředí na zkušební provoz lze považovat za dosažení milníku pro ukončení fáze konstrukce. Sice nebyla doposud dodána dokumentace ani žádný uživatelský návod, ale to vzhledem k podstoupenému průběhu této fáze zákazníkovi nevadí.

4.1.4 Předání

Poslední fáze projektu se zabývá dokončením vyvíjeného systému a jeho předání zákazníkovi do provozu. Projekt se v této fázi nese hlavně v duchu neustálého testování a oprav nalezených chyb.

Analýza všech případů užití je kompletní a dochází již jen k minoritním zásahům, které nemají žádné radikální následky při vývoji a testování. Všechny vývojářské týmy se soustředí na tzv. bugfixing. Senioršti vývojáři se začínají soustředit na code review.

V polovině prosince 2011 bylo dohodnuto datum pro Go Live v produkčním prostředí na 16. 1. 2012. Stejně tak byly schváleny další změnové požadavky, které však budou implementovány až v rámci dalších verzí aplikace.

Zákazníkem prováděné testování se postupně mění z UAT na testy přejímací. Vzhledem k potřebě celkové stabilizace verze pro Go Live a vyhnutí se zanášení nových defektů do již stabilizovaných modulů, budou v první polovině ledna opravovány pouze ty chyby, které mají kritickou nebo vysokou prioritu. Ostatní chyby budou opravovány spolu s implementací změnových požadavků až do další verze.

V tomto období se rovněž pracuje na dalších výstupech projektu, které byly sjednány před začátkem projektu. Jde o různé návody a dokumentace k jednotlivým částem systému.

Dne 16. ledna 2012 dodavatel nasazuje aplikaci na reálném prostředí zákazníka a je spuštěn pilotní provoz.

Během následujících několika týdnů bude probíhat tzv. aplikační podpora, v rámci níž budou opravovány zbylé nalezené defekty pomocí hotfixů (záplat). Všechny provedené opravy jsou sepisovány v servisních zprávách, které zákazník potvrzuje. Pro zdárné dokončení projektu zbývá provést úspěšné performance testy a odevzdat dokumentaci.

Na konci února zákazník provedl akceptaci celého projektu a potvrdil tak jeho úspěšnost.

Po skončení veškerých činností přechází projekt do fáze servisní údržby a opravování záručních závad.

5 Vlastní návrhy

Vlastní návrhy budou rozděleny na dvě hlavní části, z nichž první bude věnována vypozorovaným nedostatkům, které se na projektu během doby jeho trvání vyskytovaly. Ve druhé části bude nastíněna možnost určitého vylepšení použité metodiky vývoje, která by mohla pomoci při dalších projektech.

5.1 Nedostatky projektu

Kapitola se bude týkat nedostatků analyzovaného projektu. Jde o nedostatky, které jsou posuzovány především subjektivně jedním ze členů projektového týmu. To znamená, že jiní členové mohou mít na věc jiný názor a nemusí se s tímto ztotožňovat.

Po poukázání na fakt, v čem nedostatek spočívá, je následně uveden návrh řešení, který by mohl daný problém eliminovat (ať už v tomto projektu, nebo při některém z dalších).

5.1.1 Organizační struktura

Typ organizační struktury jako takové byl zvolen správně a byl naprosto vyhovující pro daný projekt.

Nepřehlédnutelným problémem, však byl fakt, že ačkoliv měl projekt hlavní sídlo v Brně, podílelo se na něm i mnoho pracovníků vzdáleně z centrály společnosti Unicorn Systems, a.s. v Praze. Z Prahy přicházela spolupráce několika systémových analytiků a vývojářů.

Sice v dnešní době není nejmenší problém v komunikaci na dálku (mobilní telefony, e-maily, instant messaging), ale při tak vypjatém projektu, jako byl tento, by bylo vhodné, kdyby buď byli všichni členové projektového týmu na jednom místě, anebo se alespoň jednou za čas uskutečnila schůze všech. Jednak by bylo možné na takové schůzi prokonzultovat aktuální nedostatky a problémy a jednak by mohlo psychologicky pomoci i při komunikaci např. přes telefon (je dokázáno, že telefonní hovory probíhají lépe s člověkem, kterého už osobně známe).

Jako velmi podstatný nedostatek se jeví počet pracovníků bez předchozí praxe – juniorů. Junioři byli obsazeni do pozic na jednu stranu nejnižších, na druhou stranu velmi podstatných. Jednalo se o pozice vývojářů a testerů.

Co se týče testerů, tak jejich tým čítal v průměru 6 členů. Pro každého z nich byla tato pozice nová a tak se museli s používanými postupy seznamovat až na místě. V případech, kdy vznikaly nejasnosti v některých postupech, byla možnost konzultací s testovým manažerem, architektem nebo analytikem. Protože nebylo pevně stanoveno, na koho se v takových situacích obracet, často vznikalo i zdvojené zadání, které bylo protichůdné. Přesto ale není disciplína testování tolik náročná, aby se úkoly nedaly zvládnout, případně upravit.

Horší situaci představovali junioršti developeři. Vyskytovalo se několik jedinců, kteří již s používanou technologií zkušenosti měli. Ale mnoho členů mělo možnost setkat se s postupy a přístupy teprve až na tomto projektu. Nebyl tedy výjimečný stav, že vývojář trávil nad svým úkolem mnoho hodin bez jakéhokoliv hmatatelného výstupu. Zde doslova čas plynul mezi prsty. Dá se říci, že než se vůbec tito pracovníci dostali do efektivního tempa, mohlo uplynout na dvacet dní práce (ne-li více).

Ve chvíli, kdy se na projektu podílelo okolo padesáti pracovníků, bylo až dvacet z nich juniorů. Tento počet naprosto nekorespondoval s důležitostí projektu pro dodavatelskou společnost, která měla v případě úspěchu tohoto projektu zaručenou spolupráci se zákazníkem i na dalších plánovaných projektech.

Je sice naprosto pochopitelné, že i když společnost Unicorn Systems, a.s. zaměstnává na 750 lidí, tak nemůže všechny vývojáře alokovat pouze na jeden projekt do Brna, ale s přesunem např. deseti seniorských developerů by neměl být takový problém. Druhou možností by bylo vypsání výběrového řízení na pár zkušených jedinců.

Potřeba snižování celkových nákladů ve formě výplaty poměrně nižších mezd pro větší počet juniorů oproti několika seniorům se podle mého názoru velmi podepsala na době trvání celého projektu.

5.1.2 Použité nástroje

Během vývoje natolik rozsáhlé aplikace, jakou ta uvedená bezpochyby je, by se mělo využívat moderních nástrojů, které dokážou jednak ulehčit práci, ale hlavně

ušetřit mnoho času. V dnešní době jsou již k dostání nástroje, které jsou svým zaměřením velmi úzce zacílovány a podporují celou řadu rozhraní, pomocí kterých je lze spojit s dalšími nástroji jiných odvětví.

Vývoj

Velikým problémem se během projektu ukázal být verzovací nástroj, který měli využívat především vývojáři. Každá změna kódu by měla být zaznamenána a vedle nové verze dané části kódu by měla existovat i ta původní. V každý okamžik je tak možné dohledat, kde, kdy a kým byla změna provedena.

Na projektu byl použit verzovací systém CVS (Concurrent Version System), který se řadí mezi tzv. centralizované systémy. Tyto systémy jsou charakteristické tím, že svá data ukládá na jeden server. To znamená, že veškerá aktivita s jakoukoliv verzí kódu si vyžaduje přímé spojení pracovní stanice developera se serverem. Výsledkem tak není jenom možná nedostupnost serveru, ale především doba potřebná k tomu, aby mohla být nějaká nová verze vůbec uložena (otevřena atp.). Zejména na projektu s velkým rozsahem, kde kódu denně přibývají tisíce řádků, může nastat problém s rychlostí přístupu k datům uloženým na serveru.

Už před zahájením projektu měl být zajištěn odpovídající verzovací systém, který by pokryl celý rozsah projektu. Místo centralizovaného CVS, by určitě bylo mnohem vhodnější použít některý z nástrojů, který využívá distribuovaný systém správy verzí. Distribuovaný systém se liší tím, že každý pracovník, který má měnit kód, si může kompletní historii verzí uložit na svůj pevný disk a následně provedené změny pouze synchronizovat se serverem. Větší množství času by tak mohlo využíváno efektivněji (kódováním) než čekání na ukládání na server s CVS.

Stejně jako CVS, tak i mnoho nástrojů s distribuovaným systémem je volně k dostání jako open-source. Náklady na projekt by tak nemusely být nijak navýšeny o licence. Jedním z dostupných nástrojů je např. Git, který nabízí i plug-in pro Eclipse, v němž byl veškerý kód na projektu psán.

Testování

Podobné problémy se vyskytovali i v oblasti testování, kde mělo být od samého počátku projektu jasně dáno, jaké nástroje (nejlépe osvědčené samotnou firmou) se budou během celého trvání projektu používat.

Prakticky celá fáze konstrukce (v oblasti testování) byla postavena na testovacích scénářích napsaných pomocí tabulkového procesoru Open Office Calc. K těmto scénářům se v případě provedení testů zapisovali i jejich výsledky, ze kterých se následně vytvářeli reporty o počtech chyb aplikace. Neustálé změny prováděné v UC specifikacích si vyžadovali i zásahy do testovacích scénářů, jejichž označení verzí bylo plně na uvážení jednotlivých testerů. Stejně tak vlastní podoba testovacích scénářů nebyla nijak sjednocena (kromě vzhledu šablony) a při bližším pohledu na některý ze souborů byl po čase poznat „rukopis“ každého z testerů. Často tak docházelo k situacím, kdy jeden tester neporozuměl scénáři psanému některým z jeho kolegů.

Ve chvíli, kdy byla nalezena chyba, zapsal se její nález do defektního nástroje JIRA, který byl už delší dobu používaný i na projektech samotného zákazníka. Nástroj umožňuje zadání chyb a sledování jejího celého životního cyklu (zadaná, opravená, znovuotevřená atd.). Nevýhodou byla nemožnost exportu statistik o chybách do nějakého použitelného formátu (např. xls) pro další zpracování.

Teprve až v době, kdy se začínalo s testováním uživatelských rolí a jejich práv, se začal používat pro tvorbu scénářů nástroj TestLink, pomocí něhož se testování stalo mnohem srozumitelnější a organizovanější. Do TestLinku se postupně přepisovaly i původní testovací scénáře psané v Open Office Calc.

Na pozici test manažera i test architekta byly zkušení lidé, kteří se už dříve museli setkat s podobnými projekty. Proto by bylo ideální, kdyby již na začátku stanovili a podpořili ten systém vytváření testovacích scénářů a evidence chyb, který se již osvědčil na některém z dřívějších projektů. Určitě není moudré udržovat testování rozsáhlé aplikace pomocí tabulkového procesoru, který na takovou práci není vůbec vhodný.

Naprosto jednoduchým a elegantním řešením by bylo od začátku používat adekvátní nástroj typu TestLink, který měl už v té době kladné ohlasy mezi pracovníky dodavatelské společnosti. Mohlo být ušetřeno nemalé množství času vyplýtvané na neustálé přepisování testovacích scénářů (včetně jejich šablon). Práce mohla probíhat systematičtěji a pohodlněji při zachování nastavené kvality testování.

5.1.3 Školení

Jak bylo uvedeno hned u několika metodik a popisů postupů při vývoji softwarových produktů, tak školení uživatelů ve vytvářené aplikaci by mělo probíhat v samotném závěru všech prací na projektu. V ideálním případě tehdy, když je kompletní aplikace řádně otestována a nasazena u zákazníka. Během analyzovaného projektu bylo však školení již od začátku plánováno ještě v době, kdy měly být veškeré práce v plném proudu. V tu chvíli aplikace nemohla být v žádném případě připravena pro zahájení školení. Sice již byly klíčové činnosti implementovány, ale přesto nebyly zdaleka odstraněny všechny doposud nalezené chyby.

Nepřipravenost aplikace tak vyústila nejen v posun termínů, ale i ve změnu místa konání (ČR vs. Indie).

Chyba týkající se školení vznikla již v době analyzování požadavků na vyvíjený systém. Přesto, že zákazník chtěl, aby toto školení proběhlo v době před zahájením UAT, neměl dodavatel na tento požadavek přistoupit. Velká spousta pracovního nasazení tak byla nárazově použita pro dosažení požadovaného stavu jen u určitých modulů celého systému. Práce tak nepokračovaly plynule, ale v jakýchsi obdobích tlaku a stresu.

Jednoduchým řešením by v tomto případě bylo poskytnutí zákazníkovi dostatečné vysvětlení, proč není možné zahájit školení na systém, který je stále v intenzivním vývoji. A proč je mnohem výhodnější se školením počkat na dobu, kdy by byla aplikace ve stabilním stavu a bez vážnějších nedostatků, tzn. nejdříve v době přejímacích testů.

5.2 Návrh optimalizace použité metodiky

Použitá metodika by v případě, že by byla skutečně dodržena její struktura, mohla vést ke zdárnému cíli bez větších problémů. Na sledujících řádcích bude nastíněna úprava metodiky RUP tak, aby více odpovídala danému projektovému týmu i zadání.

I přes to, že se může zdát jako nesmysl spojovat dohromady metodiku RUP s vodopádovým modelem, mohla by tato spolupráce vyústit ve velmi zajímavé výsledky. Ze všeho nejvíc záleží na správném uchopení navrhovaného řešení.

Jak již bylo zmíněno, tak vodopádový model má svou podstatu v dokonalém dokončování jednotlivých fází, ze kterých vycházejí další fáze. To je i hlavní

myšlenkou, která by se měla alespoň částečně převést do některých disciplín jednotlivých fází metodiky RUP.

Metodika RUP je jako jedna z mála orientována na tvorbu kvalitní dokumentace, která slouží buď jako východisko pro vývoj, testování a další činnosti v rámci projektu, nebo má být předána do rukou zákazníka jako součást dodávky kompletního softwaru. Proto zejména v oblasti návrhu a analýzy architektury a systému by měly být veškeré činnosti prováděny důsledně a do nejmenších detailů. Tento fakt klade nemalé nároky i na samotného zákazníka – zadavatele. Zákazník by měl mít jasnou představu o tom, jaký systém vlastně chce, jak by měl vypadat, jaké jeho stávající systémy by měl podporovat, kde by měla být ukládána data ze systému atd. Všechny zmíněné požadavky jsou nezbytné pro plynulý a systematický vývoj.

Zajištění úplné a tím pádem i kvalitní dokumentace, která zachycuje všechny požadavky, by logicky mělo prodloužit fáze zahájení a přípravy. Na druhou stranu by se ale mohla výrazně zkrátit doba potřebná pro konstrukci.

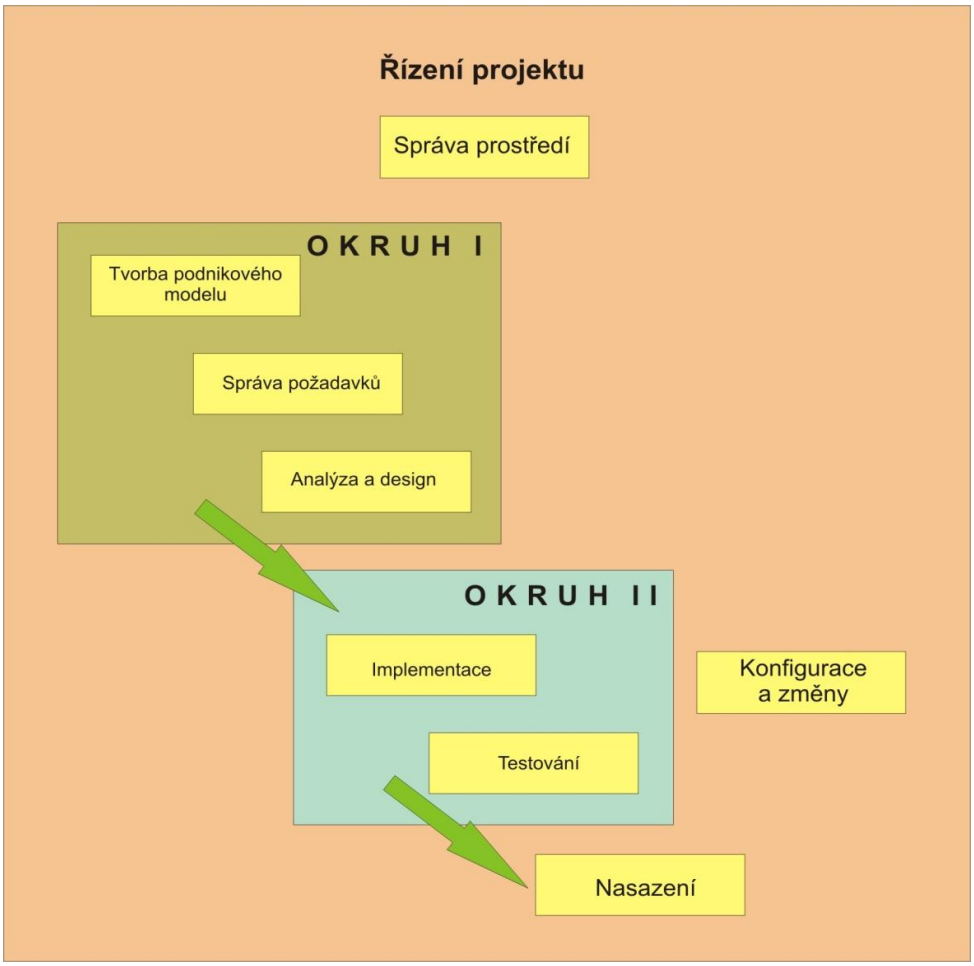
S vytvořením kvalitní a plnohodnotné dokumentace, z níž lze vycházet při vývoji i testování souvisí disciplína metodiky RUP – správa požadavků. Ta je tedy zachována i při upravené verzi metodiky.

Další z disciplín metodiky RUP je správa změnových požadavků (řízení změn). Tato disciplína by zavedením navrhovaného opatření musela striktně rozlišovat mezi změnovými požadavky, které je možné provést během probíhající konstrukce a změnami v architektuře nebo v základní vizi vyvíjeného softwaru. Změny, které by měly mít velký dopad na stabilitu celého systému, by se měly řešit až během dalšího projektu.

Další disciplína, která zaznamenává drobnou úpravu, je dodržení iterativního vývoje. Ten by se od dosavadního pojetí lišil tím, že jednotlivé iterace by se netýkali všech činností současně (viz Obrázek 8 - Iterativní přístup), ale iterace by se týkala pouhých několika vybraných činností (iterace během okruhu I, iterace během okruhu II, iterace při zavádění atd.).

Zbylé „best practices“ by měly zůstat víceméně beze změn, protože žádná z nich není v přímém rozporu s navrhovanou optimalizací RUP.

Průběh projektu podle metodiky RUP by po aplikaci zmíněných úprav mohl vypadat tak, jak je znázorněno na následujícím obrázku.



Obrázek 14 - Návrh pro optimalizaci metodiky RUP

Jednotlivé disciplíny metodiky RUP jsou seskupeny do dvou souvisejících okruhů (okruh I a okruh II). Každý z těchto okruhů by se měl navenek chovat jako jedna činnost vodopádového modelu (dokončení před pokračováním). K tomu aby mohla být zahájena práce v druhém okruhu, musí být dokončeny veškeré činnosti v okruhu prvním. Dále pak pro zahájení činnosti označené podle RUP jako *nasazení*, musí být kompletně dokončen druhý okruh. Nade všemi podruženými činnostmi se realizuje *řízení projektu* a *správa prostředí*. Činnost označená jako *konfigurace a změny* by měla probíhat souběžně s činnostmi okruhu II, přičemž, jak již bylo řečeno, změny by se neměly týkat základních stavebních kamenů vyvíjeného softwaru.

Obsahem prvního okruhu by měly být činnosti *tvorba podnikového modelu, správa požadavků* a *analýza a design*. Tyto činnosti by měly probíhat od samého

počátku projektu a jejich úkolem je sestavení dokonalého přehledu funkčností, vlastností a technických požadavků na navrhovaný software. Jde o fázi projektu, kdy je kontakt se zákazníkem naprostou nezbytností. Zároveň je velmi důležité, aby na pozicích analytiků byli zkušení a schopní pracovníci, kteří donutí zákazníka stanovit veškeré požadované informace a nároky. Kvalitně provedená analýza a celkový návrh je základem pro úspěšnost celého projektu a proto musí být veškeré specifikace vytvořeny tak, aby podle nich bylo možné provádět vývoj a zároveň by informace v nich obsažené plně odpovídali budoucí podobě softwaru.

Tím se první soubor činností stává náročným jak pro analytiky dodavatele, tak i pro zákazníka, který je vystaven mnoha otázkám a sám musí mít jasnou představu o celém softwaru. Jím sdělené informace má v této chvíli možnost zkontrolovat pouze pomocí navržených modelů a specifikací. V rámci okruhu I může proběhnout několik iterací, během nichž by měly být vysvětleny a předány veškeré detaily.

Po dokončení okruhu I by se projektový tým měl zaměřit na činnosti okruhu II – *implementace* a *testování*.

Vzhledem k očekávané kvalitě předložených podkladů z okruhu I, by měl bez problémů začít vývoj jednotlivých částí softwaru. Vyvinuté moduly (po unit testech vývojářů) je potřeba integrovat do jediného celku a následně otestovat i ten. Stejně jako v předchozím případě, tak i zde je vhodné projít postup několika iteracemi, na jejichž koncích bude výsledkem stále kvalitnější aplikace s narůstajícím počtem splněných požadavků.

Členové týmu, kteří se podíleli na zpracování okruhu I se v této době věnují konzultacím s ostatními členy projektového týmu, případně vytvářejí další specifikace pro změnové požadavky. Stejně tak v tuto dobu vytvářejí nejrůznější dokumenty pro zákazníka (manuály, dokumentace, školící materiály atd.).

Teprve ve chvíli, kdy je naimplementovaný systém kompletně otestovaný a neobsahuje žádné závažné chyby, může se projekt posunout do úkolů spojených s *nasazením* aplikace na reálné prostředí.

Projekt je podporován a organizován za pomoci *správy prostředí*, které zajišťuje nasazení kvalifikovaných, schopných a ochotných pracovníků a jejich pracovních nástrojů a pomůcek. Správa prostředí rovněž obsahuje komunikaci se zákazníkem a zajištění veškerých vstupů od něj. Kdyby nastala situace, že by některý modul

aplikace byl připraven k integraci se systémem zákazníka a ten by dosud nebyl připraven, mohlo by docházet ke zdržování, které by však neměl na svědomí projektový tým dodavatele.

Činnost, která musí probíhat neustále během celé doby trvání projektu je *řízení projektu*. Projektoví manažeři nejprve komunikují se zákazníkem a vyjednávají podmínky jejich spolupráce. V dalších fázích se starají o běh projektu správným směrem, plány, rizika atd. Na schopnostech projektových manažerů většinou stojí úspěch a kvalita poskytnutého softwaru (příp. služby).

Obrovskou výhodou, která vyplývá z uvedeného návrhu, je fakt, že samotný vývoj je postaven na hotové specifikaci požadavků a tím pádem může být veškeré úsilí směřováno k tvorbě kvalitního a robustního kódu a pečlivému testování. To je zárukou, že výsledná aplikace bude mít optimalizovaný chod a bude mnohem spolehlivější. Zároveň ve spojení s frameworkem ITIL by byl zajištěn i následný servis a údržba dané aplikace.

Výhoda navrhované metodiky se může snadno stát i její největší slabinou. V případě, že se analýzou zadavatelových požadavků zabývá méně zkušený pracovník, může docházet k tomu, že požadavky na systém nebudou kompletní nebo jednotné vývoj tak nebude moci prakticky vůbec fungovat.

6 Přínos vlastního návrhu

Během analyzování průběhu projektu vyšlo najevo několik poměrně zásadních faktů, které svým způsobem ovlivnily nepříliš příznivou dobu dokončení. Je otázkou zda některé ze zmíněných problémů mohly být vyřešeny dříve a lépe než se tak ve skutečnosti stalo.

Proto byla vedle opatření k jednotlivým nedostatkům navržena i optimalizace metodiky RUP, která by mohla tomuto konkrétnímu projektu pomoci s vypořádáním se s některými nástrahami. Samozřejmě ve chvíli, kdy již projekt skončil, je pozdě něco měnit na jeho průběhu, ale protože se společnost Unicorn Systems, a.s. zabývá projekty s podobným zaměřením poměrně často, mohla by být uvedená optimalizace převedena do praxe na některém z budoucích kontraktů.

Navrhovaná optimalizace by měla pomoci při předcházení problémům projektu, které mohou vycházet ze zneužívání některých pouček metodiky RUP (možnost zadání zásadního změnového požadavku před koncem fáze konstrukce apod.).

Závěr

V úvodu práce byly popsány nejrozšířenější a nejznámější metodiky a postupy, podle nichž lze vytvořit kvalitní a funkční produkt na poli IT. Některé jsou vhodné pouze pro ilustraci toho, jak vůbec může koordinovaný vývoj probíhat, některé jsou vhodné především pro menší tým lidí, kteří pracují na projektu se společným cílem a některé se používají ve velkých komerčních software housech, v nichž vznikají více či méně úspěšné podnikové aplikace a systémy.

Uvedené metodiky a postupy byly využity při analýze současného stavu tvorby informačních systémů a aplikací firmou Unicorn Systems, a.s. Současným stavem je myšlen průběh prací na jednom konkrétním projektu, který společnost řešila během roku 2011. Prvním krokem bylo předvedení, jak by projekt měl podle metodiky RUP probíhat – návaznost činností, intenzita činností apod.

Ve druhém kroku byl popsán skutečný stav projektu, tak jak se odehrával. Jeho průběh byl rozdělen na jednotlivé fáze v souladu s metodikou RUP. Celý postup byl zachycen včetně chyb projektu, které se podepsaly na jeho termínu dokončení.

Další část se zaměřila na návrh na odstranění zmíněných nedostatků a ze zjištěných skutečností byl proveden návrh nové, lépe vyhovující metodiky vývoje softwaru.

Výsledkem práce je tedy představení optimalizované metodiky, která by mohla být v budoucnu využita při řízení podobných projektů, jakým byl ten analyzovaný. V návrhu jsou zmíněny jednotlivé postupy a přístupy k metodice tak, aby mohla být snadno převedena do praxe. Vedle jejích výhod jsou dále vyjmenovány i její nevýhody, které by mohly ovlivnit rozhodnutí, zda metodiku použít či nikoliv.

.

Seznamy

Seznam literatury

[1] Accelerated Ideas [online]. [cit. 2012-05-20]. Dostupné z: http://www.accelerated-ideas.com/free-itil-training/itil-service-lifecycle-11927.aspx

[2] ALDORF, F. Objektová analýza, návrh a programování [online]. 2005 [cit. 2011-12-17]. *RUP – Životní cyklus projektu*. Dostupné z: http://objekty.vse.cz/Objekty/Rup3

[3] ALDORF, F. Objektová analýza, návrh a programování [online]. 2005 [cit. 2011-12-17]. *Základní charakteristiky rup*. Dostupné z: http://objekty.vse.cz/Objekty/Rup2#kap23

[4] AMBLER, S. Ambysoft [online]. 2005 [cit. 2011-12-17]. *The Agile Unified Process (AUP)*. Dostupné z: http://www.ambysoft.com/unifiedprocess/agileUP.html

[5] AMBLER, S. Ambysoft [online]. 2000 [cit. 2011-12-17]. *The Unified Process Inception Phase*. Dostupné z: http://www.ambysoft.com/books/inceptionPhase.html

[6] BOEHM, B. *A Spiral Model of Software Development and Enhancement*. TRW Defense Syst. Group, Redondo Beach, CA, roč. 21, č. 5, August 1988. ISSN: 0018-9162

[7] CDI.cz [online]. 2011 [cit. 2011-12-17]. *Systémová analýza*. Dostupné z: http://www.cdi.cz/sluzba-software.php?sluzba_id=3

[8] CEJTHAMR, V. a DĚDINA, J. *Management a organizační chování: 2., aktualizované a rozšířené vydání.* Praha: Grada Publishing, a.s., 2010. ISBN 978-80-247-3348-7. [cit. 2012-05-15]. Dostupné z: http://www.businessinfo.cz/cz/clanek/management-msp/typy-organizacnich-struktur-cleneni/1001663/59204/?page=7

[9] DOLEŽAL, J.; LACKO, B.; MÁCHAL, P. *Projektový management podle IPMA*. Praha: Grada Publishing, a.s., 2009. 507 s. ISBN 978-80-247-2848-3.

[10] DUDKA, K. *Softwarové inženýrství (IUS)*. Dudka.cz [online]. [cit. 2012-05-15]. Dostupné z: http://dudka.cz/studyIUS

[11] DVOŘÁK, D., MAREČEK, M. a RÉPAL, M. *Řízení portfolia projektů: Nejlepší praktiky portfolia managementu*. Praha: Computer Press, 2011, 200 s. ISBN 9788025130759.

[12] FOTR, J.; SOUČEK, I. *Investiční rozhodování a řízení projektů*. Praha: Grada Publishing a.s., 2011. 408 s. ISBN 978-8-024-73293-0.

[13] HIGHSMITH, J. *History: The Agile Manifesto*. In: Manifesto for Agile Software Development [online]. 2001 [cit. 2012-05-15]. Dostupné z: http://www.agilemanifesto.org/history.html

[14] CHARVAT, J. *Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects*. New Jersey: John Wiley and Sons, Inc., 2003. 264 s. ISBN 978-0-471-22178-4.

[15] *ITIL – výkladový slovník a zkratky v češtině* [online]. v1.0. 2011[cit. 2012-05-18]. Dostupné z: http://www.itil-officialsite.com/nmsruntime/saveasdialog.aspx?lID=1211&sID=242

[16] *ITIL*. ITIL [online]. [cit. 2012-05-15]. Dostupné z: http://www.itil.cz/index.php?id=982

[17] *ITIL*. MODALISA-TECHNOLOGY. [online]. [cit. 2012-05-18]. Dostupné z: http://www.modalisa-technology.com/competencies/maturity-models/itil/

[18] JELÍNEK, J. *Spolupráce metodik ITIL a RUP*. 2008. Dostupné z: http://student.vsmie.cz/~jelij5ap/itil-a-rup.pdf

[19] KOCH, M. et al. *Management informačních systémů*. Vyd. 2., přeprac. Brno: Akademické nakladatelství CERM, 2010, 171 s. ISBN 978-80-214-4157-6.

[20] Koučink firem [online]. 2011 [cit. 2011-12-17]. *Informační řízení*. Dostupné z WWW: http://www.koucinkfirem.eu/informace.html

[21] KRUCHTEN, P. *Going Over the Waterfall with the RUP*. DeveloperWorks [online]. 2004 [cit. 2012-05-15]. Dostupné z: http://www.ibm.com/developerworks/rational/library/4626.html

[22] KŘENA, B. a KOČÍ R. *Úvod do softwarového inženýrství IUS* [online]. Brno, 2006 [cit. 2012-05-15]. Dostupné z: http://www.scribd.com/doc/48636337/Uvod-do-softwaroveho-inzenyrstvi-IUS. Studijní opora. FIT VUT.

[23] Logica: Be brilliant together [online]. 2011 [cit. 2011-12-17]. *Systémová integrace a dodávka projektů*. Dostupné z: http://www.logica.cz/we-do/projektove-a-programove-rizeni/systemova-integrace-a-dodavka-projektu/

[24] *Manifesto for Agile Software Development*. [online]. 2001 [cit. 2012-05-15]. Dostupné z: http://www.agilemanifesto.org/

[25] MARTINÁK, T. *Eliminácia projektových rizík iteratívnym vývojom*. Infoware. 2011, 8-9/2011, s. 47. Dostupné z: http://www.softec.sk/files/prednasky/mr_softec.pdf

[26] NOSEK, J. *Manažerské katastrofy*. Praha: Computer Press, 2011, 256 s. ISBN 978-80-2512-691-2.

[27] OSTERWALDER, A. a PIGNEUR Y. *Tvorba business modelů: Příručka pro vizionáře, inovátory a všechny, co se nebojí výzev*. Praha: Bizbooks, 2012, 278 s. ISBN 978-80-265-0025-4.

[28] PHILLIPS, J. *IT project management: on track from start to finish*. 2. Emeryville, CA: McGraw-Hill Professional, 2004. 535 s. ISBN 978-0-072-23202-8.

[29] POKORNÝ, M. *Vyvíjíme databázový a informační systém I*. Databázový svět [online]. 2004, 05. 05. 2004 [cit. 2012-05-15]. Dostupné z: http://www.dbsvet.cz/view.php?cisloclanku=2004050501

[30] *Principy stojící za Agilním Manifestem*. [online]. 2001 [cit. 2012-05-15]. Dostupné z: http://www.agilemanifesto.org/iso/cs/principles.html

[31] *Projektové řízení*. TELEFÓNICA O2 CZECH REPUBLIC, a.s. ITIL: IT Service Management [online]. 2007 [cit. 2012-05-15]. Dostupné z: http://www.itil.cz/index.php?id=915

[32] *Rational Process Library: The industry's most robust collection of best practices guidance.* IBM [online]. [cit. 2012-05-21]. Dostupné z: http://www-01.ibm.com/software/awdtools/rmc/library/#Practices

[33] ROYCE, W. *Managing the Development of Large Software Systems*. Proceedings of IEEE WESCON. 1970, s. 1-9. Dostupné z: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

[34] *Software Process Models*. Target: The-Software-Experts [online]. [cit. 2012-05-15]. Dostupné z: http://www.the-software-experts.de/e_dta-sw-process.htm

[35] STACKPOLE, C. *A User's Manual to the PMBOK Guide*. New Jersey: John Wiley and Sons, Inc., 2010. 240 s. ISBN 978-0-470-58489-7.

[36] ŠMÍD, V. *Životní cyklus informačního systému*. In: [online]. [cit. 2012-05-15]. Dostupné z: http://www.fi.muni.cz/~smid/mis-zivcyk.htm

[37] *UML® Resource Page*. OMG. Unified Modeling Language [online]. 2007, 2011 [cit. 2012-05-20]. Dostupné z: http://www.uml.org/

[38] VACULNÝ, J. *Agilní metodologie programování*. In: [online]. [cit. 2012-05-15]. Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2003/xvaculny.htm

[39] Interní materiály společnosti Unicorn Systems, a.s.

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

ASP – Application Service Provider

CCTA – Central Communications and Telecommunications Agency

CVS – Concurrent Version System

E2E – End-to-end (typ testování)

HW – Hardware

IPMA – International Project Management Association

ITIL – Information Technology Infrastructure Library

PDCA – Plan, Do, Check, Act (Demingův cyklus)

PMBOK – Project Management Body of Knowledge

PRINCE2 – PRojects IN Controlled Environments

RUP – Rational Unified Process

SW – Software

UAT – User Acceptance Testing

UESPC – Unicorn ES Powered Company

UC – Use Case

UCS – Use Case Specification

UML – Unified Modeling Language