

**Fakulta managementu Vysoké školy ekonomické
v Praze
Jindřichův Hradec**

Katedra managementu informací

Diplomová práce

Bc. Tomáš Novotný

2011

**Fakulta managementu Vysoké školy ekonomické
v Praze
Jindřichův Hradec**

Katedra managementu informací

**Řízení životního cyklu
softwarového produktu: teorie a
praxe**

Vypracoval:

Bc. Tomáš Novotný

Vedoucí diplomové práce:

doc. Ing. Dr. Jan Voráček, CSc.

V Jindřichově Hradci, Prosinec 2011

Poděkování:

Rád bych poděkoval vedoucímu mé práce **doc. Ing. Dr. Janu Voráčkovi, CSc.** za jeho vstřícnost a cenné rady

a

společnosti **J.K.R., s.r.o.** za cenné zkušenosti a poskytnuté materiály.

Prohlášení:

Prohlašuji, že diplomovou práci „Řízení životního cyklu softwarového produktu: teorie a praxe“ jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v přiloženém seznamu literatury.

V Jindřichově Hradci, Listopad 2011

.....
podpis studenta

Anotace

Řízení životního cyklu softwarového produktu: teorie a praxe

Práce seznamuje se současnými trendy řízení vývoje softwarových řešení (procesními modely). Ve zvolené aplikační oblasti analyzuje aktuálně aplikované metodiky. V souladu s nastudovanou teorií identifikuje úzká místa stávajících procesů a navrhuje jejich zlepšení.

Prosinec 2011

Obsah

Abstrakt	1
Abstract	1
Úvod	2
1. Historie vývoje software	3
1.1. Vývoj informačních systémů	3
1.1.1. Definice ERP.....	5
2. Informační systém a projektování IS	6
2.1. Metodika vývoje IS	7
2.2. Rozdíl mezi modelem a metodikou vývoje IS	7
2.3. Potřeba vývojových procesních modelů	7
3. Životní cyklus	8
3.1. Obecně o životním cyklu.....	8
3.1.1. Před-analytická fáze	9
3.1.2. Analýza	9
3.1.3. Návrh	9
3.1.4. Vývoj systému.....	9
3.1.5. Implementace systému	9
3.1.6. Správa systému	9
3.1.7. Údržba systému.....	9
3.2. Manažerský pohled na životní cyklus	10
3.2.1. Inicie.....	11
3.2.2. Konstrukce	11
3.2.3. Dodání	12
3.2.4. Provoz.....	12
4. Modely životního cyklu	13
4.1. Rigorózní a agilní metodiky	14
4.2. Tvorba IS v kontextu podnikového managementu.....	14
4.3. Procesní modely životního cyklu	15
4.4. Sekvenční modely	15
4.4.1. Vývojový cyklus „vodopád“	15
4.4.2. Prototypový přístup	17
4.5. Iterativní modely	18
4.5.1. „Přírůstkový“ (inkrementální) vývojový cyklus	18
4.5.2. Spirálový model	18
4.5.3. Rational Unified Process.....	20
4.6. Evoluční modely	21
4.6.1. Inkrementální programování	21
4.6.2. Extrémní programování	21
4.6.3. Agilní programování	22
4.7. Agilní přístup k programování a SCRUM v praxi	22
4.7.1. Proč vyvíjet agilně?	22
5. SCRUM.....	24
5.1. Role v týmu.....	24
5.2. Sprint.....	25
5.3. User story	25

5.4.	Product backlog.....	26
5.5.	Poker planning.....	26
5.6.	Sprint backlog.....	26
5.7.	Daily meeting.....	26
5.8.	Workflow.....	27
5.9.	Body složitosti.....	28
5.10.	Velocity týmu	28
5.11.	Burndown chart	29
5.12.	Planning session	31
5.12.1.	Demo	31
5.12.2.	Retrospekce (retrospektiva).....	31
5.12.3.	Plánování náplně dalšího sprintu	31
5.13.	Aktivita – plánování	32
6.	Software maintenance	33
6.1.	Software maintenance a náklady společnosti.....	35
6.2.	Software maintenance jako proces.....	37
7.	Normy a standardy.....	38
7.1.	Český standard životního cyklu	38
7.2.	ISO/IEC 12207	38
7.3.	ISO/IEC 15504	39
8.	Charakteristika podnikatelského subjektu, společnosti J.K.R. spol. s r.o.	40
8.1.	Základní charakteristiky společnosti	40
8.2.	Profil společnosti	41
8.3.	Organizační struktura	42
8.4.	Produkty	43
8.4.1.	ByznysDOS.....	43
8.4.2.	ByznysWin	44
8.4.3.	ByznysVR	44
8.4.4.	Byznys EVO.....	45
9.	Procesy a metriky	46
9.1.	Procesní přístup	46
9.2.	Posloupnost procesů	48
9.3.	Procesy týkající se zákazníka.....	51
9.3.1.	Určování požadavků týkajících se produktu.....	51
9.3.2.	Přezkoumání požadavků týkajících se produktu	51
9.3.3.	Komunikace se zákazníkem	51
9.4.	Měření, analýza a zlepšování	52
9.4.1.	Monitorování a měření	52
9.4.2.	Řízení neshodného produktu	54
9.4.3.	Analýza dat	55
9.4.4.	Neustálé zlepšování.....	56
10.	Vývoj a údržba v minulosti	57
10.1.	Postup při řešení uživatelského požadavku.....	57
10.2.	Návrh a vývoj.....	59
10.2.1.	Plánování návrhu a vývoje	59
10.2.2.	Vstupy a výstupy návrhu a vývoje	59
10.2.3.	Přezkoumání a ověřování	59
10.2.4.	Validace	60
10.2.5.	Prezentace nových funkcí.....	60

10.2.6. Řízení změn návrhu a vývoje	60
10.3. Řízení procesu údržby	61
10.4. Kategorizace chyb	62
10.5. Organizace zadání úkolu	63
10.5.1. Obecné povinnosti zadavatele úkolu	63
10.6. Ostrá a distribuční verze	64
10.6.1. Ostrá verze	65
10.6.2. Distribuční verze	65
10.7. Shrnutí	65
10.7.1. Vytíženost programátorů	66
10.7.2. Vytíženost konzultantů	66
10.7.3. Spokojenost zákazníka	66
10.7.4. Týmovost	66
11. Vývoj a údržba v současnosti	67
11.1. Agilní vývoj v J.K.R.	67
11.1.1. Délky iterací v jednotlivých týmech	69
11.2. Byznys tým	71
11.3. Shrnutí	73
11.3.1. Vytíženost programátorů	73
11.3.2. Vytíženost konzultantů	73
11.3.3. Spokojenost zákazníka	73
11.3.4. Týmovost	74
11.4. Scrum v Byznys týmu - praxe	75
11.4.1. IKO – přehled úkolů	75
11.4.2. TO - BTScrum	77
12. Případové studie	79
12.1. Zdokonalovací údržba	79
12.1.1. Vstupní data	79
12.1.2. Metriky hodnocení	80
12.2. Vodopád	81
12.2.1. Případová studie 1 – Spotřební daň 2 – kolky	81
12.2.2. Případová studie 2 – Komunikace s insolvenčními rejstříky	81
12.2.3. Případová studie 3 – Datové schránky – fáze I	82
12.2.4. Případová studie 4 – Oddělení Osvobození od daně a Bez daně	82
12.2.5. Případová studie 5 – CreditCheck vs. Insolvenční rejstřík	83
12.2.6. Zhodnocení	83
12.3. Scrum	84
12.3.1. Případová studie 1 – Fakturace ISDOC a PDF	84
12.3.2. Případová studie 2 – Nové formuláře pro Mzdy	84
12.3.3. Případová studie 3 – Období pro DPH	85
12.3.4. Případová studie 4 – Rozdělení období	85
12.3.5. Případová studie 5 – Dobropisy ke skontu	86
12.3.6. Zhodnocení	86
12.3.7. Celkové vyhodnocení případových studií	87
12.4. Oprava chyb	87
12.4.1. Vstupní data	87
12.4.2. Metriky hodnocení	87
12.4.3. Výsledky metody vodopád	88

12.4.4. Výsledky metody scrum	88
12.4.5. Vyhodnocení opravy chyb.....	89
13. Závěr.....	90
Zdroje	92
Příloha č. 1	94
Příloha č. 2	94
Příloha č. 3	96
Příloha č. 4	97

Abstrakt

Práce se zaměřuje na životní cyklus produktu (softwaru) ve společnosti od jeho vývoje, ladění, přes jeho distribuci a následnou údržbu. V práci jsou vymezeny jednotlivé procesy, které v tomto životním cyklu vznikají, jsou popsány a analyzovány. Porovnává praxi, jak je software vyvíjen, laděn, distribuován a spravován ve firmě, kde jsem pracoval a teorii, která je čerpána z teoretických studií.

V praktické části je práce zaměřena na software maintenance, hlavně na srovnání teorie a praktické aplikace. Pro tyto účely jsem využil zkušenosti ze společnosti J.K.R., s.r.o. V závěru práce věnuji část pro zhodnocení získaných poznatků a návrhy na zlepšení životního cyklu tak, aby v případě jejich využití, mohl tento proces fungovat efektivněji.

Abstract

The work focuses on the life cycle of a product (software) in the company since its development, debugging, through its distribution and subsequent maintenance. The work defined by the individual processes within the life cycle occur, are described and analyzed. Compares the practice, as the software is developed, debugged, distributed and managed in the company where I worked and theory, which is drawn from theoretical studies.

In the practical part of the work is focused on software maintenance, especially on the comparison of theory and practical applications. For this purpose I used the experience of JKR, Ltd. In conclusion I devote a section for evaluation of lessons learned and suggestions for improving the life cycle so that in case of their use, could this process work efficiently.

Úvod

Jako téma mé diplomové práce jsem si zvolil řízení životního cyklu softwarového produktu. Absolvoval jsem střední školu s hlavním oborem výpočetní technika a nyní pracuji jako IT specialista ve společnosti CPI Group, kde mám na starosti správu informačních systémů, jejich implementaci a project management. Dva roky jsem působil jako aplikační konzultant ve společnosti, která se zabývá vývojem a distribucí podnikových informačních systémů – ERP (Enterprise resource planning).

Problematika vývoje softwaru a životního cyklu je velice diskutovaným problémem a zabývá se jím mnoho publikací. Existuje mnoho různých přístupů jak aplikovat procesní modely životního cyklu.

Má diplomová práce je rozdělena na teoretickou a praktickou část. V teoretické části, která je ohraničena prvními sedmi kapitolami, si ukážeme jaká je historie vývoje softwarových produktů. Práci bych chtěl aplikovat na podnikové informační systémy, proto si dále v této části vysvětlíme co to informační systém je a jaká je jeho struktura. Z blízka se zaměříme na trendy řízení vývoje softwarových řešení – procesní modely a na samotný životní cyklus informačních systémů. Ukážeme si, jaké jsou jednotlivé fáze životního cyklu a také výhody a nevýhody různých přístupů. Neopomeneme si představit, jaká je podpora v českých a mezinárodních standardech.

V praktické části představím společnost, ve které jsem působil a podnikový informační systém, který společnost vyvíjí a distribuuje. V praktické části jsem si zvolil dva cíle. Prvním je identifikace procesních modelů, které firma při vývoji softwaru používá a jeho srovnání teoretické a praktické použitelnosti. V nedávné době prošla firma změnou v přístupu k vývoji svého produktu. Druhým cílem tedy je srovnání minulého a současného přístupu.

Cílem práce je tedy dojít k výsledkům srovnání teorie a praxe ve využívání různých procesních modelů a navržení doporučení, která by vedla ke zlepšení využití procesního modelu.

1. Historie vývoje software

Cílem této kapitoly je popsat chronologický vývoj softwaru a podnikových informačních systémů s přihlédnutím k souvislostem managementu znalostí a modifikací při vývoji podnikových informačních systémů.

1.1. Vývoj informačních systémů

Podnikání v dnešní době si bez informačních technologií snad už nelze ani představit. Pro velké podniky jsou stále více klíčové komplexní informační systémy známé pod zkratkou ERP (Enterprise Resource Planning), které umožňují rychlejší a přesnější zodpovězení otázek managementu, či vedení podniku souvisejících s vedením jak celé firmy, tak i s jednotlivými a dílčími procesy. Vzhledem ke stále rychlejším změnám ve vnějším prostředí podniku se i s časem postupně měnily požadavky na tyto informační systémy. První takové systémy se dříve zaměřovaly pouze a zejména na výrobní procesy, případně účetnictví. Postupem času, rozšířením informačních technologií a snížením jejich cen, došlo k rozšíření informačních systémů, jak do počtu instalovaných aplikací, tak i do jejich komplexnosti. V současnosti informační systémy soustřeďují a poskytují informace z a do všech procesů v podniku podle aktuální potřeby a plní tak jejich funkce: monitorovat, analyzovat a plánovat podnikové procesy. Dnes je zřejmé, že úspěch podniku je z velké části ovlivněn i výkonností a kvalitou informačního podnikového systému.

První podnikatelské organizace, které začaly společně s půdou, kapitálem a prací využívat informace jako zdroj k podnikání, se začaly objevovat na americkém kontinentu ve dvacátých až třicátých letech dvacátého století. Příkladem těchto firem mohou být korporace Henryho Forda, Gerarda Philipse či Tomáše Bati. [16] Úspěch těchto firem byl založen zejména na efektivním řízení informačních toků, řízení inovací a řízení kontinuálního procesu učení - tyto tři faktory jsou pak označovány jak podstata systému řízení znalostí, tedy základem pro učící se organizace, které se snaží podílet na vytváření svého okolí a ovlivňovat jeho změny ve svůj prospěch, případně na změny rychle a pružně reagovat. Tyto společnosti již tehdy začaly systémově využívat zpracovávání informací pro vyhledávání souvislostí mezi zpracovávanými informacemi a pro vytváření predikcí budoucího vývoje. Dnes jsou významným nástrojem pro včasné vyhodnocení informací podnikové informační systémy, ať už se jedná o systémy komplexní, nebo otevřené systémy zaměřené na konkrétní oblast řízení.

První podnikové informační systémy se začaly objevovat v šedesátých letech minulého století. Jednalo se o programované aplikace v tehdy rozšířených programovacích jazycích (zejména COBOL, FORTRAN či ALGOL), připravované přesně pro danou organizaci a pro daný účel.[18] Tyto první systémy sloužily výlučně ke kontrole zásob a skladů, či k řízení jednotlivých výrobních procesů. Později se z těchto aplikací vyvinuly komplexnější systémy, které pokrývaly materiálové plánování výroby a byly známy pod zkratkou MRP (Material Requirements Planning). V roce 1981 představil Oliver Wight¹ tzv. systém MRP II (Manufacturing Resource Planning), který sloužil k řízení a optimalizaci dodávek materiálů a výroby. Ten zasahuje hlouběji do výrobního procesu a je integrován s činnostmi celého podniku.[4] Během osmdesátých let byly podnikové aplikace označovány termínem Automatizované Systémy Řízení a vžila se pro ně zkratka ASŘ.

V devadesátých letech začíná v důsledku nových informačních technologií a klesající ceny hardwarového vybavení velký rozmach podnikových systémů. V této době se začaly objevovat první systémy ERP, které k výše uvedené oblasti rozšiřovaly a přidávaly další moduly či funkce z širších oblastí fungování podniků.

Tyto systémy vznikaly třemi cestami[2]:

- rozvojem již existujícího řešení podnikového systému

Tato varianta byla z krátkodobého hlediska nejlacinější a nejrychlejší, na druhou stranu nemuselo toto řešení odpovídat požadavkům na podnikový systém v budoucnosti.

- vývojem nového informačního systému na míru

Cenově a časově náročné řešení, které ale plně odpovídalo potřebám podniku

- nákupem hotového softwarového systému

Toto řešení značně závisí na dodavateli, cenově i časově je příznivější než vývoj vlastního systému a je zaručen další vývoj a funkcionalita systému.

Na vývoj informačních systémů má značný vliv i masový rozmach využívání internetu, který je používán jako zdroj pro sběr informací i jako komunikační médium pro spojení jednotlivých filiálek velkých společností či nadnárodních korporací. Internet jako takový se vyvíjí také neustále a pro využití podnikových informačních systémů se používají stále nové komunikační protokoly, formy datových souborů i samotný způsob využití internetu.

¹ Zakladatel významné konzultační společnosti

1.1.1. Definice ERP

Definice pro podnikové informační systémy ERP není jednoznačná a existuje více možností jak systémy ERP popsat. Příkladem může být tato definice:

ERP systém představuje softwarové nástroje sloužící k řízení podnikových dat. ERP systémy pomáhají organizovat oblasti řízení dodavatelského řetězce, příjmu, skladového řízení, řízení zákaznických objednávek, plánování výroby, expedice, účetnictví, lidských zdrojů a dalších podnikových funkcí.[2]

Z uvedené definice vyplývá, že dnes už víme, co lze od systémů ERP očekávat.

V současné době podnikový informační systém podporuje všechny vyjmenované důležité podnikové funkce, které se s novými distribučními cestami dají rozšířit o e-business a m-business. Nynějším trendem ve vývoji podnikových systémů je jejich otevřenost jak v možnosti využití informací z jiných aplikací či informačních systémů partnerů uživatele, tak ve zpracování výstupů informačního systému. To je oproti dosavadně dodávaným systémům změna – tyto systémy byly uváděny spíše jako jeden komplexní systém, který se snažil obsáhnout všechny funkce spíše v uzavřeném systému od čerpání informací až po jejich zpracování přímo v prostředí aplikace informačního systému.

2. Informační systém a projektování IS

V této kapitole bychom si měli nejprve objasnit některé základní pojmy, se kterými se v této práci budeme setkávat. Jedná se o pojmy informační technologie a informační systém.

Informační technologií můžeme označit prostředky založené na výpočetní technice a postupy pro sběr, uchování, zpracování a přenos informací. Informační technologie sestává z komponent hardware, software a komunikační technologie. [7]

Informačním systémem můžeme označit účelově využitě informační technologie pro zpracování určitých úloh neboli aplikací. Informační systém sestává z následujících komponent: účel, lidé, pracovní postupy, informace a informační technologie. [7]

Pro lepší pochopení těchto dvou termínů si můžeme uvést příklad. Pokud budeme hovořit o informační technologii, tak tímto termínem můžeme označit např. osobní počítač, na kterém bude nainstalován operační systém Windows a tabulkový kalkulátor Microsoft Excel. Pokud budeme chtít mluvit o informačním systému, tak k tomuto osobnímu počítači musíme přidat i další složky informačního systému. Musíme osobnímu počítači dát nějaký účel – např. zpracování plánu investic podniku. Počítač musí obsluhovat člověk – např. investiční referent. Pracovním postupem bude postup pro zpracování plánu investic a informacemi rozumíme informace potřebné pro stanovení požadovaného plánu. Pokud si takto definujeme jednotlivé komponenty, můžeme již hovořit o informačním systému pro zpracování plánu investic. [7]

Výše uvedený příklad je velice zjednodušený model sestavení informačního systému. V současné době jsou informační systémy vyvíjeny v rámci projektů. Činnosti probíhající v rámci projektu, jejichž cílem je vytvoření informačního systému, nazýváme projektováním IS. [7]

Jako projekt lze definovat množinu vzájemně souvisejících činností, které sledují splnění společného cíle, jsou omezeny určitými zdroji (čas, peníze, materiál...), jsou založené na strategickém plánu projektu a jsou navrženy, organizované a realizované pod řízením projektového manažera v zájmu zadavatele. Výsledkem projektu je výsledný informační systém. [7]

Pod pojmem projektování IS rozumíme tedy činnosti prováděné v rámci projektu vývoje IS, tj. činnosti pokrývající celý proces vývoje informačního systému od specifikace zadání informačního systému až po jeho předání zákazníkovi k rutinnímu provozování. [7]

„Informační systémy v podnicích pokrývají široké pole technologie zpracování podnikových dat. Zahrnují problematiku sběru dat, přenosů, uchování, zpracování a informačního využití dat.“ [7]

„Informační systémy v podniku rovněž mohou zastávat různou roli z hlediska podpory podnikatelské strategie podniku a z hlediska významu pro provoz firmy. V podniku mohou být provozovány IS s významem podpůrným, provozním, či s významem strategickým. V posledním případě jde o IS, bez jejichž správné funkčnosti nemůže podnik plnit své obchodní a výrobní úkoly. To může vést ke značným finančním i morálním ztrátám podniku nebo dokonce i k jeho likvidaci úspěšnější konkurencí.“ [7]

„Uvědomíme-li si šíři pojmu IS a význam informačních systémů v moderních podnicích, je zřejmé, že vývoj IS nemůže probíhat chaoticky (ad-hoc). Vývoj IS musí být postaven na předem definovaných, optimalizovaných a řízených postupech. Jen tak lze vytvořit potřebné předpoklady pro požadovanou kvalitu výsledného IS. V současné době existuje množství více či méně standardizovaných postupů vývoje IS, tj. existují metodiky a metody projektování IS.“ [7]

2.1. Metodika vývoje IS

Pod pojmem metodika vývoje IS si lze představit soubor doporučených etap, přístupů, zásad, postupů, pravidel, metod, technik, nástrojů, dokumentů, metod řízení, který pokrývá celý proces vývoje IS. Metodika stanoví kdy, kdo, co a proč má dělat během procesu vývoje IS a je základním standardem a návodem k dílčím postupům spojeným s tvorbou IS.

Metodika vývoje IS zahrnuje organizaci práce vývojového týmu, metody práce s informacemi o vyvíjeném IS, ekonomické otázky vývoje IS, vedení projektové a provozní dokumentace k IS a také způsob řízení v jednotlivých fázích vývoje IS.

2.2. Rozdíl mezi modelem a metodikou vývoje IS

V následujících kapitolách se budeme bavit o jednotlivých modelech a metodikách vývoje informačních systémů a software obecně. V této chvíli je potřeba pochopit správně rozdíl mezi těmito dvěma pojmy.

Budeme se mimo jiné například bavit o modelech spirály a vodopádu. V obou těchto případech se stále jedná o modely životního cyklu softwarového produktu. Rozdíl mezi metodikou je ten, že model životního cyklu popisuje fáze, kterými softwarový produkt prochází, přičemž metodika určuje, co se v které fázi má přesně dělat, jaké postupy se mají použít a k jakému výsledku se má nakonec dojít.

2.3. Potřeba vývojových procesních modelů

Na konec této části jsem zařadil hlavní důvody, proč jsou vývojové procesní modely tolik důležité při vývoji a údržbě software.

Výhodou při využití některé z dále popsanych metodik je fakt, že při jejím využití dochází k vhodnému zadání algoritmů chodu životního cyklu od analýzy aplikace, přes její programování až po její údržbu. Pro implementační tým též dochází k výraznému zamezení efektu bobtnání projektu. Možné odhady pracnosti na projektu a jeho řízení jsou specifikovány daným modelem. Dále dochází k efektivnější a snadnější tvorbě dokumentace. Při řízením se postupy, které jsou dány daným modelem, dochází k vytváření podkladů pro funkcionální testování vyvinuté aplikace. Poslední výhodou je také efektivní tvorba dokumentu funkční specifikace produktu jako přílohy smlouvy mezi odběratelem a dodavatelem software.

3. Životní cyklus

V následujících částech mé práce se již budeme zabývat pojmy jako životní cyklus softwarového produktu, procesními modely či software maintenance. Následující kapitoly by měly být nápomocny správně čtenáře uvést do problematiky řešené v praktické části.

3.1. Obecně o životním cyklu

V této kapitole se budeme zabývat pojmy metodologie vývoje softwaru a životní cyklus. Vysvětlíme si, jak spolu tyto dva pojmy souvisí, jak se v průběhu času vyvíjely různé metodologie vývoje software a v návaznosti na to i životní cyklus softwaru.

Pokud se budeme bavit o životním cyklu softwaru, budeme se bavit o metodikách vývoje software, ze kterých se skládá ucelený model životního cyklu. Podle [12] je metodika softwarového vývoje pojmem softwarového inženýrství, kterým označujeme tzv. framework používaný pro návrh, plánování a řízení procesu vývoje informačního systému. Pod pojmem metodika se také rozumí využití takového frameworku pracovním týmem či celou organizací při vývoji informačního systému.

První formalizovaný metodický Framework pro vývoj informačních systémů se nazýval SDLC (Systems development life cycle) a byl představen v roce 1960. Jako hlavní myšlenka SDLC bylo „*nastavit vývoj informačních systémů velmi účelným, strukturovaným a metodickým způsobem, což vyžadovalo, aby všechny fáze vývoje od nápadu až po dodání finálního systému byly provedeny postupně a vždy v nezměnné formě*“[3]. Hlavním cílem bylo „*vyvinout rozsáhlé funkční obchodní systémy pro velké obchodní konglomeráty. Typickou aktivitou informačních systémů bylo hromadné zpracování dat a složité aritmetické výpočty, tzv. number crunching*“[3]

Životní cyklus informačního systému se tedy skládá z jednotlivých etap neboli fází, které na sebe navazují. Tyto etapy pokrývají kompletní vývoj IS od jeho počátku – prvotního nápadu vývoje IS, přes vymezení jeho požadavků, tvorbu konceptuálního a implementačního modelu, vlastní implementaci, zavedení, testování až po údržbu systému a jeho provoz. Poslední fáze může být označena jako stažení IS z užívání. [20] Jedná se tedy v podstatě o časové období životnosti a vývojových změn systému od jeho zavedení přes projektování, realizaci, zavedení a provozování (včetně údržby a rozvoje) až k jeho zániku, případně nahrazení jiným systémem. [10]

Životní cyklus může být tedy chápán jako označení pro etapizaci života IS a metodika vývoje IS. Je to určitý model vývoje systému nebo také speciální metodika pro postup práce na projektu, který IS vyvíjí. [10]

Jednotlivé fáze se dle různých autorů různí a přizpůsobují se danému konkrétnímu případu. Sami vývojáři si životní cyklus přizpůsobují podle jejich vlastních potřeb. Tím může dojít k tomu, že ne vždy se shodují názvy jednotlivých etap a činnosti, které do daných etap patří. Změnou životního cyklu dochází ke zvýšení efektivity a produktivity, jelikož si je vývojáři přizpůsobí dle svých potřeb [20].

Je nutno dodat, že tyto fáze jsou popisovány z hlediska vývojářů softwaru. Důležitý manažerský pohled je popsán v následující části této práce. Za klíčové etapy životního cyklu informačního systému lze však podle [10] označit následující.

3.1.1. Před-analytická fáze

V první fázi se uplatňuje byznys modelování, specifikace požadavků na činnost a funkcionalitu systému. V této fázi jsou na systém kladeny požadavky – základní podmínky, které musí systém splňovat. Provádějí se zde tzv. studie proveditelnosti (tzv. feasibility study), které jsou určeny k identifikaci a analýze problémů spojených s předběžným návrhem na vývoj systému s cílem ukázat jeho realizovatelnost a účelnost. V neposlední řadě se v této fázi specifikují požadavky na systém. Mělo by jít o podrobnou formulaci ve tvaru dokladu poskytující definitivní popis systému z pohledu uživatele pro potřeby jeho vývoje a ocenění jeho správnosti. [10]

3.1.2. Analýza

V této fázi jde o modelování budoucího systému na konceptuální úrovni. [10]

3.1.3. Návrh

V této fázi jde o modelování budoucího systému na technologické úrovni. Zde se převádějí specifikace na hierarchii stále detailnějších schémat, která definují požadovaná data a rozkládají procesy, jež se mají s daty provádět, až na úroveň, kdy mohou být vyjádřeny v podobě instrukcí pro počítačový program.[10]

3.1.4. Vývoj systému

Zde se již jedná o psaní a testování počítačového softwaru – programování. Je zde i vývoj vstupních a výstupních formulářů a konvencí.[10]

3.1.5. Implementace systému

Při implementaci je vlastní IS uveden do provozu. Jedná se o instalaci fyzického systému a aktivity s tím související, jako je například školení operátorů a uživatelů. [10]

3.1.6. Správa systému

Při správě systému se jedná o další vývoj funkcí a struktury systému, jež vyplývají ze změněných požadavků a technologií, zkušeností s užíváním systému a dolaďováním jeho výkonu. [10]

3.1.7. Údržba systému

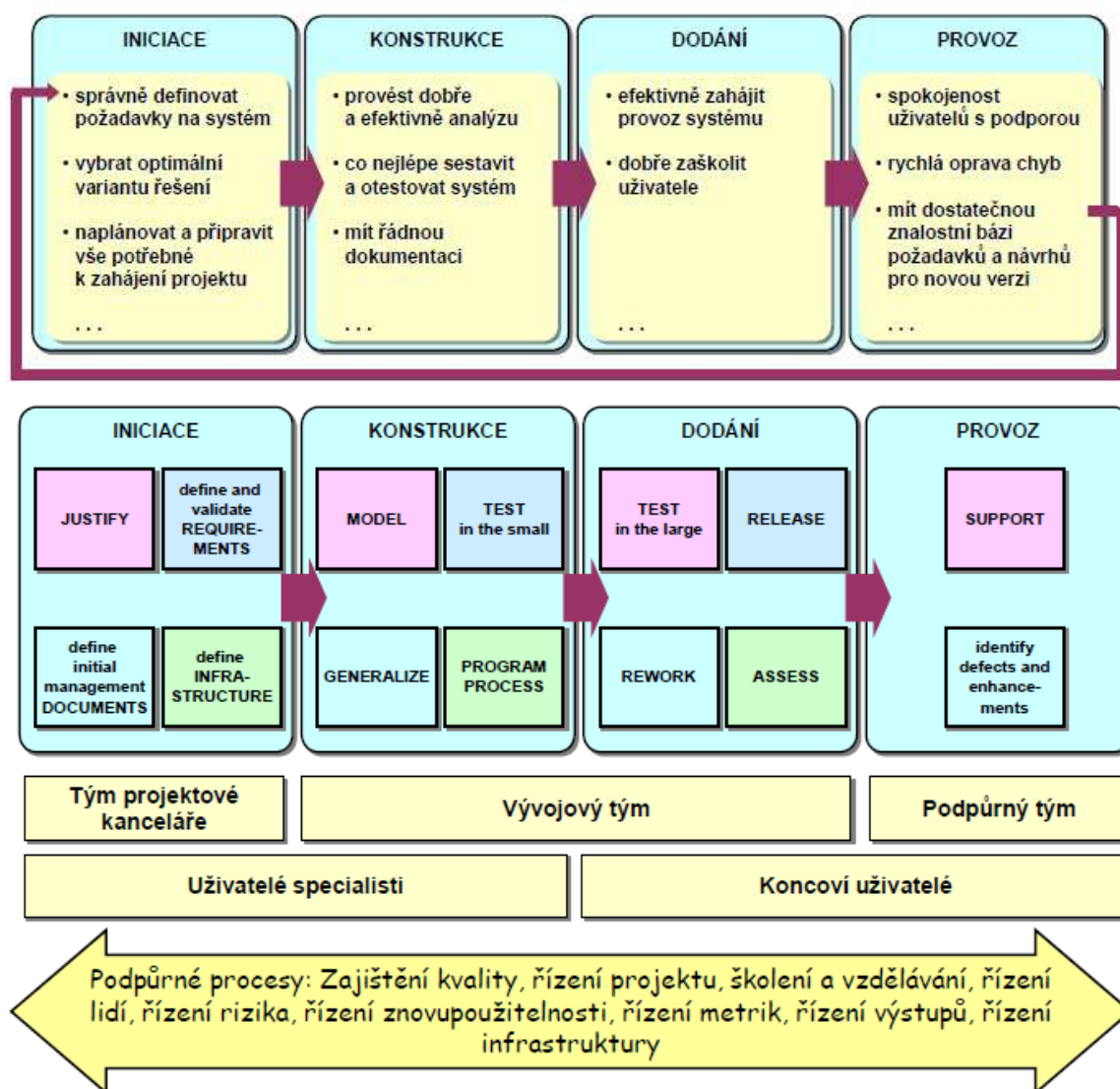
Jedná se především o úpravu systému při jeho provozování podle nově vzniklých požadavků uživatele, resp. pro odstranění závad. [10]

3.2. Manažerský pohled na životní cyklus

Výše jsem se zmínil, že předchozí fáze modelu životního cyklu jsou popisovány z pohledu vývojáře software. Pro účely této práce a následné aplikace v praktické části je důležité nahlížet na jednotlivé fáze z manažerského pohledu. O toto se snaží tato část.

V první řadě je nutné si uvědomit, že přípravu a realizaci software je nutné pochopit jako proces. Manažerský přístup pohlíží na tvorbu softwaru z perspektivy projektového řízení očima manažera. To má za následek jiný pohled na jednotlivé fáze než předchozí model.

Z manažerského pohledu se celý životní cyklus skládá z fází „iniciace“, „konstrukce“, „dodání“ a „provoz“, které se dále dělí na dílčí procesy. Nejlépe to vyjadřují následující dvě schémata.



Obrázek 1: Manažerský pohled na jednotlivé fáze

„Čtyři hlavní fáze by se měly provádět sekvenčně po sobě. Protože na konci každé fáze dochází ke změnám týmu a platform, tak je doporučováno, aby manažer projektu neopomněl svolat pracovní setkání, kde prezentuje dosavadní postup projektu, provede kontrolu dokumentace a dalších dosud vytvořených výstupů a seznámí pracovní skupinu s novými členy týmu. Dílčí procesy uvnitř fází je možné provádět opakovaně – iterativně nebo evolučně. Tento model v sobě kombinuje výhody všech hlavních přístupů: Umožňuje plánování a projektování ve velkém (= celý projekt) a zároveň se nebrání experimentování v malém (= uvnitř fází).“[11]

Nyní si popíšeme hlavní fáze životního cyklu podrobněji.

3.2.1. Iniciace

Cílem iniciace je připravit vše potřebné k zahájení projektu. Musí se zde zajistit všechny přípravné práce (definice požadavků, dokumentace, příprava infrastruktury, proces zmocnění).

Definice požadavků (define and validate requirements) znamená prvotní sběr požadavků na nový informační systém. Většinou se tento sběr informací provádí na prvotní schůzce účastníků projektu.

Následuje příprava manažerské dokumentace (define initial management documents). Jedná se o přípravu plánu čerpání zdrojů a přípravu časového plánu celého projektu. Definují se zde požadavky na kapacity, technické a finanční zdroje a sestavení harmonogramu.

Další dílčí částí fáze iniciace je příprava infrastruktury (define infrastructure). Jedná se o přípravu podkladů, informačních zdrojů a případnou instalaci dílčích systémů, které jsou nezbytné pro tvorbu informačního systému.

Poslední dílčí částí je proces zmocnění či narovnání (justify). Během tohoto procesu dochází k vypracování rizik celého projektu, právnímu zabezpečení a počítačové bezpečnosti. Případně v této fázi může být vypracována alternativa k řešení a výběr optimální varianty.

3.2.2. Konstrukce

Teprve v této fázi začíná vlastní tvorba informačního systému. Může docházet k iterativnímu opakování dílčích procesů. Může k tomu docházet z důvodu, že teprve nyní dochází ke zpřesňování zadání a správnému pochopení.

Prvním dílčím procesem je proces modelování (model). V tomto procesu dochází k podrobné definici zadání a úplnému získání podkladových materiálů. Nedílnou součástí tohoto procesu je modelování a simulování cílových uživatelských procesů.

Následuje proces zvaný sestavování (program process). Až v této chvíli dojde k vlastnímu naprogramování software. Pro účely budoucích úprav systému by měla být vytvořena systémová dokumentace a také uživatelská dokumentace – manuál.

Dalším procesem je proces zvaný testy v malém (test in the small). Tento proces slouží k testování funkčnosti. Testování se neprovádí na provozní platformě, ale na testovací, což znamená, že se ho účastní pouze speciálně určení členové vývojového týmu – nikoli skuteční budoucí uživatelé informačního systému.

Posledním procesem této fáze je proces generalizace (generalize). Tento proces si klade za cíl optimalizaci vytvořeného informačního systému a identifikaci potenciálně znovupoužitelných a nahraditelných částí systému. Jedná se v podstatě o „vyčištění“ zdrojového kódu, aby následující úpravy a opravy systému byly co nejjednodušší.

3.2.3. Dodání

Tato fáze slouží k uvedení vytvořeného informačního systému do provozu u zákazníka. Následující dílčí procesy nemusí běžet sekvenčně, ale mohou být prováděny souběžně.

Dílčí proces vydání (release) zahrnuje instalaci informačního systému u koncového uživatele, řešení problémů s instalací a přípravu provozní infrastruktury. Neoddělitelnou součástí tohoto procesu jsou i podpůrné a pomocné činnosti, jako například propagace systému koncovým uživatelům a školení uživatelů.

Následuje proces testy ve velkém (test in the large). Proces je podobný jako testy v malém, ale testy jsou prováděny zástupci koncových uživatelů.

Po dokončení testů je možné zařadit proces přepracování (rework). Jedná se o opravný proces, který vychází z výsledků testů ve velkém. Jsou zde realizovány změny a úpravy, které koncovým uživatelům nevyhovují.

Posledním procesem fáze dodání je zhodnocení projektu (assess). V tomto procesu se vyhodnocují předchozí procesy. V této chvíli je možné ocenit iniciativu účastníků projektu, vyhodnocení chyb, kvality, použité metriky apod.

3.2.4. Provoz

Tato fáze pokrývá používání systému v praxi. Skládá se ze dvou důležitých dílčích procesů.

Prvním důležitým dílčím procesem je podpora (support). Do supportu lze zařadit činnosti help desku. Jedná se o podporu uživatele při užívání informačního systému týkající se ovládání a instalace systému, péče o průběžné doškolení nových uživatelů systému. Nedílnou součástí je řešení problémů uživatelů spojených s poruchami hardware, sběr námětů ke zlepšení systému atp.

Druhým důležitým dílčím procesem je údržba (maintenance). Údržba systému zahrnuje operativní odstraňování chyb v systému. Je důležité rozlišovat podněty vedoucí ke zlepšení systému a chyby, které zapříčiňují nemožnost práce se systémem.

Tomuto tématu – Software maintenance – je věnována celá kapitola a její poznatky budou použity v praktické části této práce, která se bude zabývat procesy při údržbě informačního systému.

4. Modely životního cyklu

Účelem této části je seznámit čtenáře s různými typy modelů životního cyklu. Tato část by měla čtenáři přiblížit přehled o tom, co jednotlivé modely vnášejí do procesu vývoje – kvalitu, agilitu, rychlost a spokojenost. V následující kapitole poté bude čtenář seznámen s jednotlivými procesními modely, které jsou v této části zmíněny.

Životní cyklus se dá rozdělit na různé modely. Modely životního cyklu se dají rozdělit do tří základních kategorií, podle toho, jaké metodiky se v něm používají. Tyto kategorie jsou: sekvenční, iterativní a evoluční.

Sekvenční model životního cyklu je založen na tom, že existuje systematický postup, jak se dá dojít od počáteční fáze (zadání) až k řešení pomocí na sebe navazujících činností, které je možné předem naplánovat a definovat. Nejznámější varianta toho modelu je tzv. vodopádový model, který byl v minulosti velice rozšířen a používal se nejen v oblasti vývoje informačních systémů. Tento model je například užíván v jiných inženýrských disciplínách, jelikož jiné řešení neexistuje – například stavebnictví. [11]

Mezi jeho přední výhody lze zařadit jeho jednoduchost, dobrou možnost řízení a sledování postupu řešení. Jeho základní nevýhodou, díky níž je od tohoto modelu upouštěno, je fakt, že vyžaduje mít na začátku vývoje úplně přesně definovány všechny požadavky, které musí systém zahrnovat.

Iterativní model životního cyklu je založen na tom, že je možné se vracet do předchozích fází vývoje projektu za účelem zpřesnění či změnou zadání. Nejznámější varianta iterativního přístupu je prototypový model a spirálový model.

Hlavní výhodou tohoto přístupu je možnost zahájit vývoj dříve, než jsou známy všechny požadavky na informační systém. Pro další fáze je také možno využít předchozí zkušenosti z prvotní implementace pro zpřesnění zadání. Nevýhodou tohoto modelu je horší možnost řízení projektu a sledování postupu řešení.[11]

Třetí skupinou jsou **evoluční modely** životního cyklu. *“Ty jsou založeny na myšlence provádění projektu postupně po menších částech, přičemž celý systém vzniká postupně tak, jak se i během vývoje mění požadavky na něj.”* [11] V poslední době získávají tyto modely velkou oblibu. Nejznámějšími variantami tohoto přístupu je inkrementální programování, komponent based development a agilní metodiky, kterými je například extrémní programování či SCRUM. Touto skupinou modelů a hlavně metodice SCRUMu se budeme zabývat v dalších částech mé práce a hlavně v praktické části.

Nespornou výhodou tohoto přístupu je možnost postupného vývoje a údržby, stejně tak i krátká doba mezi zadáním a řešením dílčích požadavků na informační systém. Na druhou stranu je potřeba zmínit jeho velkou nevýhodu – velmi obtížné řízení a sledování projektu. Nejvýrazněji se tato nevýhoda projevuje u velkých projektů.

4.1. Rigorózní a agilní metodiky

Vedle rozdělení typů modelů životního cyklu, který je popsán v předchozí kapitole (sekvenční, iterativní a evoluční), je možné modely dále rozdělit podle metodiky rigorózní a agilní.

Rigorózní metodiku je možno chápat jako označení přístupu založeného na sekvenčním modelu od autorů agilních metodik [11]. Pod tuto metodikou lze zahrnout jakoukoliv metodiku, ve které se objevují různé fáze během vývoje a pracuje se s formální dokumentací – a nezáleží na tom, že metodika může být iterativní nebo evoluční. Nevýhodou rigorózních metodik je jejich přílišná složitost, malá účinnost a požadavek na velké množství dokumentace. Tím je zkomplikován samotný proces vývoje a následné implementace.

Na druhé straně stojí agilní metodiky. Agilní metodiky umožňují vytvořit řešení velmi rychle a umožňují řešení operativně přizpůsobovat. Mezi nejpoblárnější patří opět extrémní programování a scrum.

Agilní přístup má v praxi dobré výsledky. Má velké přednosti v malých týmech, které pravidelně komunikují se zákazníkem.[11]

4.2. Tvorba IS v kontextu podnikového managementu

„Při práci na velkých projektech se analytici informačních systémů setkávají s problémem, kdy funkčnost budovaných rozsáhlých systémů má vliv na vlastní organizační a řídicí strukturu podniku nebo organizace, kam se systém zavádí – jsou to například nové či pozměněné pracovní funkce, změna řízení, nová oddělení, nová potřeba legislativní podpory, ... Proto je žádoucí se při práci na informačních systémech zabývat i změnou těchto souvisejících struktur. V praxi se však bohužel tato otázka podceňuje a problém zavádění a fungování informačních systémů se řeší „od opačného konce“, tedy například se provádějí výběrová řízení na konkrétní technologie (např. čipové karty nebo jiná koncová zařízení) aniž by došlo ke správnému pochopení a nastavení business procesů, související legislativní podpory atp.“[11]

4.3. Procesní modely životního cyklu

V současné době existuje spousta metodik vývoje software. V této části mé práce si představíme ty nejdůležitější, kterým se budu věnovat i v praktické části mé práce. Představíme si nejdříve tradiční (sekvenční) metodiky životního cyklu software, jakými jsou např. vodopádový model a prototypový přístup. Dále se budu věnovat iterativním modelům, kterými jsou např. inkrementální přístup a spirálový model. Nakonec se budu věnovat agilnímu přístupu - evolučním modelům jako například extrémní programování a scrum. Agilnímu pojetí vývoje software se budu věnovat i v následující kapitole, která je důležitá pro praktickou část této práce.

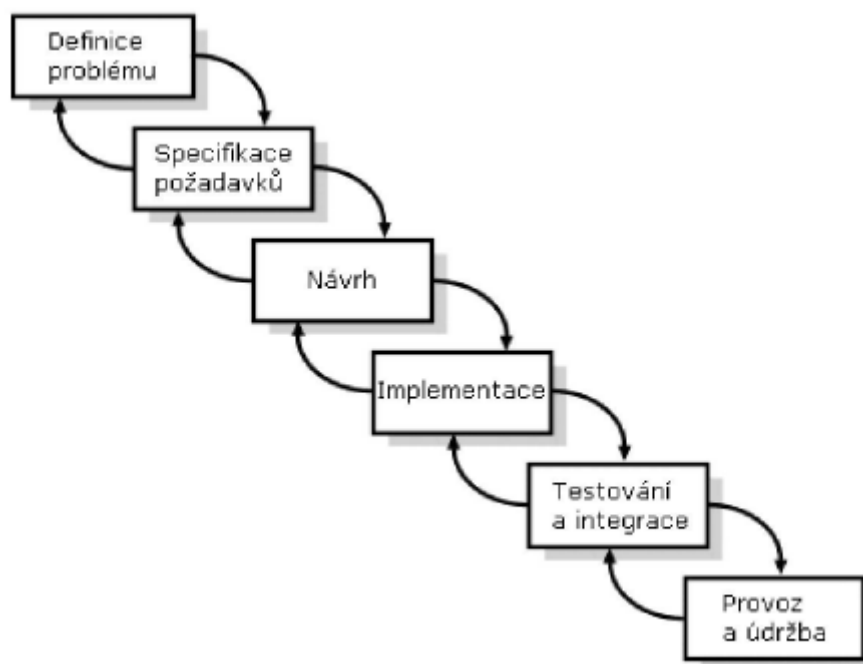
4.4. Sekvenční modely

4.4.1. Vývojový cyklus „vodopád“

Vodopádový model je považován za první ucelenou metodiku vývoje software. Je to sekvenční vývojový proces bez iterací, ve kterém je vývoj nahlížen jako neustále se svažující tok jednotlivými fázemi. Mezi jednotlivými fázemi je schvalovací proces, přes který musí všechny dokumenty projít, aby vývoj mohl pokračovat v další fázi.[5], [12] Základními fázemi vodopádového modelu jsou:

- ❖ Definice problému, poznání zákazníka a cílové oblasti
- ❖ Analýza a specifikace požadavků
- ❖ Návrh
- ❖ Implementace
- ❖ Testování a integrace
- ❖ Provoz a údržba

Schéma podle [19] je vidět na následujícím obrázku. Schémat znázorňující vodopádový model je nespočet. Vybral jsem to, které mi přišlo jako nejvhodnější.



Obrázek 2: Schéma životního cyklu typu Vodopád

Model vodopád není striktně sekvenční, jak se z popisu výše může zdát. V modelu je možnost pohybovat se vždy o jednu fázi dopředu nebo zpět. To nám umožňuje vrátit se např. z fáze implementace do návrhu, ten upravit dle současných požadavků a pak se opět vrátit zpět do fáze implementace. Je možné se pohybovat pouze o jednu fázi. Teprve až ve fázi Provoz a údržba je možné se vracet do jakékoli předchozí fáze a provádět požadované úpravy. [5]

Mezi hlavní výhody tohoto modelu lze jistě zařadit přehlednost a jednoduchost. Model je jednoduchý jak na pochopení a práci podle něj, tak také na řízení. Každá fáze je zakončena zpracováním předem daných dokumentů, přechody mezi jednotlivými fázemi zajišťuje schvalovací řízení. Je možné tedy dobře kontrolovat, v jaké fázi se projekt právě nachází a jaká část specifikace požadavků je již splněna.[5]

Nevýhod tohoto modelu je mnohem více než výhod. Je to dáno paradoxně jednoduchostí modelu a také jeho stářím. Pro velké projekty je nevýhodou, že je tento model takto jednoduchý – model nedokáže pokrýt veškeré aspekty rozsáhlých projektů. Další nevýhodou je to, že uživatel zpravidla nedokáže v počátečních etapách formulovat přesné požadavky na systém. To má za následek to, že pokud obdržíme během vývoje nový požadavek od uživatele, musíme znovu projít fázemi specifikace požadavků, analýzy a návrhu, včetně schvalování všech dokumentů. Další nevýhodou je to, že systém je dodáván celý najednou. Zákazník má tedy dlouho pocit, že se nic neděje.[5], [10]

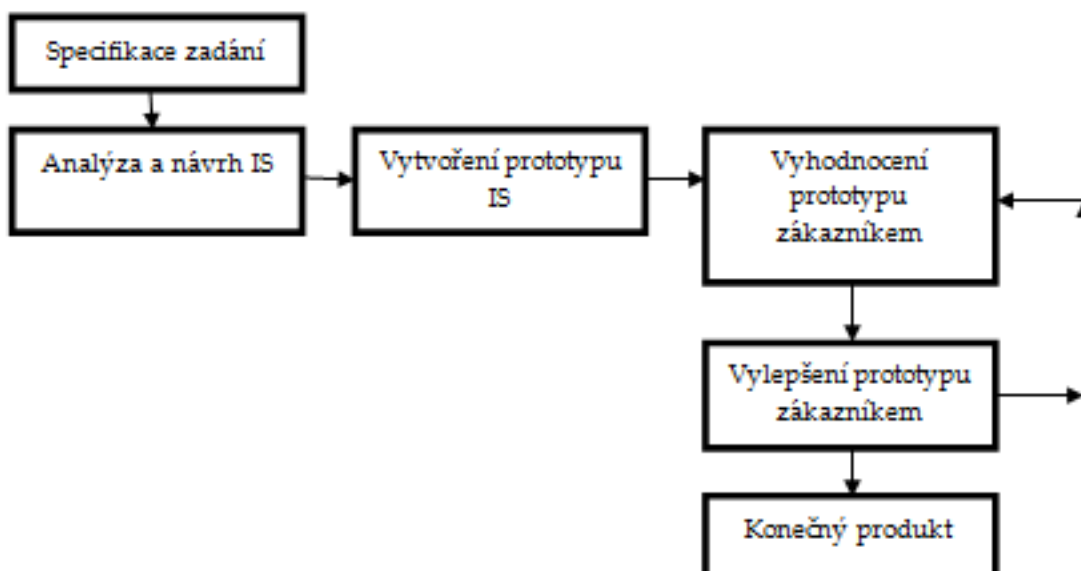
Ve své době byl tento model revoluční a vznikla podle něj spousta softwarových produktů. Pro současné projekty je ovšem lepší používat některou z metodik popisovaných dále v této práci, která následuje soudobé trendy ve vývoji software.

4.4.2. Prototypový přístup

Jedná se o vývoj pomocí vytváření prototypů, tzn. dochází k vývoji neúplných verzí software. Prototypový přístup podle [12] není samostatným a kompletním přístupem metodiky vývoje, ale spíše přístup k jednotlivým částem větších tradičních metodik vývoje software (tj. přírůstková metoda, spirála, nebo RAD - Rapid application development).

Jedná se o snahu snížit nebezpečí projektových rizik rozdělením projektu na menší části a zjednodušit tak možnost změn v průběhu procesu vývoje.[12]

Uživatel je zapojen do procesu vývoje, tudíž vidí, jak vývojový tým postupuje a zvyšuje se tím pravděpodobnost přijetí konečného produktu uživatelem. V iterativním procesu jsou vytvářeny malé ukázky systému, dokud prototyp není vyvinut dle požadavků uživatele. Nevýhodou je, že většina prototypů bude vyřazena.



Obrázek 3: Prototypový přístup

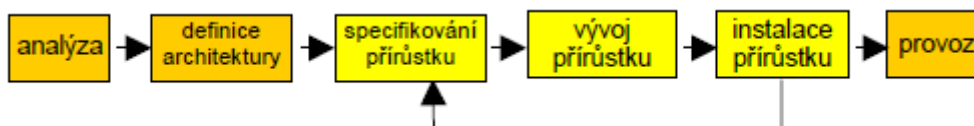
4.5. Iterativní modely

4.5.1. „Přírůstkový“ (inkrementální) vývojový cyklus

„Inkrementální přístup je vhodný pro kombinaci sekvenčních a iteračních metodik softwarového vývoje. Cílem je omezit projektová rizika rozdělením projektu na menší segmenty a zjednodušuje možnost zavedení změn během procesu vývoje. Základní principy inkrementálního přístupu:“ [20]

V modelu jsou prováděny série malých vodopádů (viz předchozí kapitola), přičemž každý malý vodopád je prováděn jen pro malou část systému. Jakmile je tato malá část dokončena, pokračuje se na další. Požadavky na software jsou definovány dříve než se zahájí vývoj prvního malého vodopádu. [20]

Model iterativního cyklu s přírůstkem rozkládá celý proces na iterace, kde každá z nich předpokládá prototyp. Každá další iterace (přírůstek) přidává nové funkce softwaru. Výhodou tohoto principu je to, že krátké iterace jsou lépe kontrolovány a plánovány než celý vodopád najednou. [20]



Obrázek 4: Inkrementální vývojový cyklus

Mezi klady tohoto přístupu patří fakt, že celý systém je specifikován a navržen na začátku, ale zákazníkovi je dodáván v sérii přírůstků – tudíž dalšímu vývoji napomáhá zpětná vazba od koncového uživatele.[20]

Mezi hlavní zápory patří fakt, že je obtížné sledovat a kontrolovat postup práce na vývoji celého systému. [20]

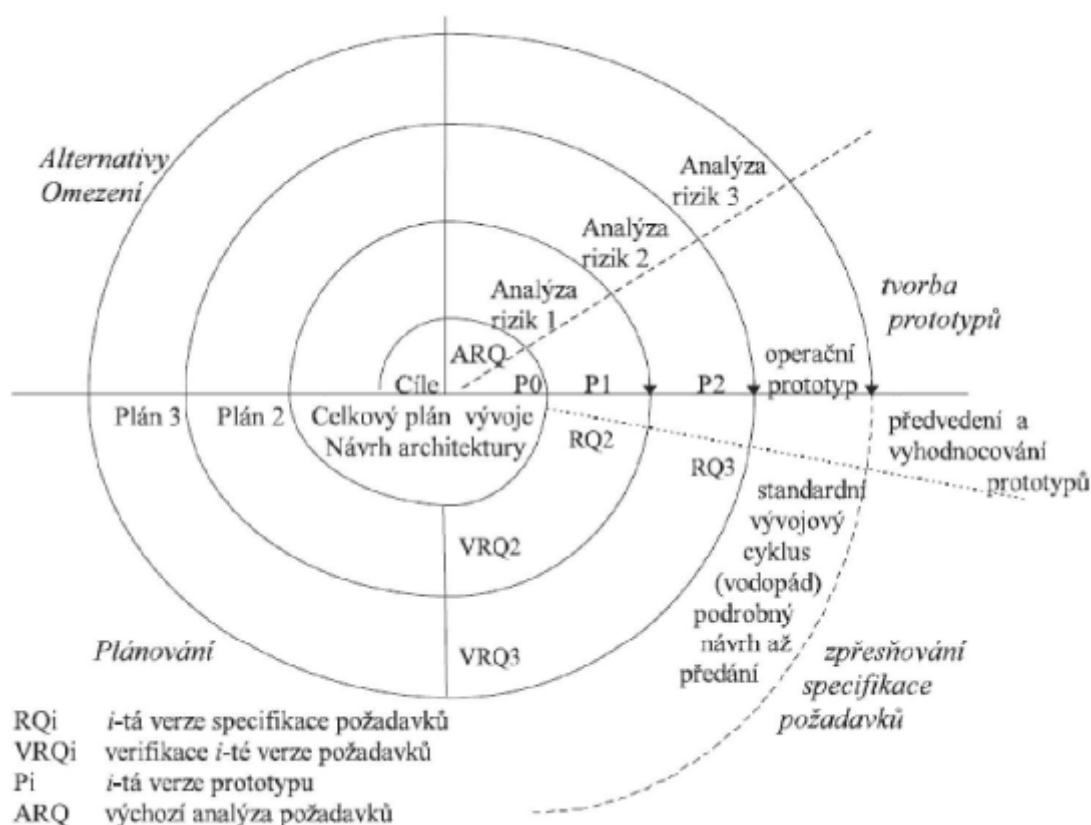
4.5.2. Spirálový model

Spirálový model vychází z vodopádového modelu. Vznikl v roce 1985 (Barry Boehm), kdy vodopádový model brzy po svém vzniku začal být nedostačujícím. Spirálový model přináší dvě nové vlastnosti – iterativní přístup a podrobnou analýzu rizik.

Ve vodopádovém modelu byl problém to, že u spousty projektů se nepodařilo stanovit přesnou a úplnou specifikaci požadavků, což činilo vodopádový model skoro nepoužitelný.[5] To vyústilo v cyklické opakování jednotlivých kroků vývoje, kdy se na začátku projektu pouze určil rámec architektury a funkčnost celého systému se postupně rozpracovávala do detailů v tzv. iteracích.

V každém cyklu je navíc prováděna analýza rizik, která určuje další směr vývoje projektu. Riziko je tedy klíčovým pojmem, pod kterým se rozumí jakákoliv událost nebo situace, která může ohrozit projekt. Projekt se rozdělí na menší segmenty, ve kterých je umožněno provádět změny během procesu vývoje. V průběhu vývoje je také možné vyhodnocovat rizika a zvažovat další pokračování projektu v průběhu životního cyklu.[12]

Schéma spirálového životního cyklu je na obrázku níže přejato z [9].



Obrázek 5: Schéma životního cyklu typu Spirála

Hlavní výhodou je nezávislost na konkrétní metodice vývoji software, která se při samotném vývoji použije. [5] Model se hodí i pro větší projekty, jelikož je velice komplexní. Díky neustálé analýze rizik se zde výrazně snižuje možnost nevhodného řešení projektu.

Díky své komplexitě je model vhodný pro velké projekty, svou výhodou ale ztrácí, budeme-li se bavit o malých projektech. Pro malé projekty je model příliš složitý. Z vodopádového modelu si spirálový model zachoval jednu nevýhodu – výsledný produkt je zákazníkovi předán až po dokončení posledního cyklu. Model spoléhá na metodiku, která je při vývoji použita a proto nijak dopodrobna nezpracovává jednotlivé činnosti, které by se měli v průběhu projektu provádět.[5]

Model spirály byl používán v řadě projektů. Stal se z něj důležitý milník v minulosti v softwarovém inženýrství. V současnosti jeho použití opadává. Existují již daleko více propracované metodiky a pro současné typy aplikací je tento model nepružný.[5]

4.5.3. Rational Unified Process

Rational Unified Process (RUP) je komerčním produktem firmy Rational, která byla později koupena firmou IBM. Jedná se o velice rozsáhlou a detailně propracovanou metodiku vývoje software. Vývoj podle RUP probíhá v iterativních cyklech. V prvním cyklu, který se nazývá „úvodní vývojový cyklus“, je výsledkem funkční softwarový produkt, který obsahuje nejpodstatnější části funkcionality. V dalších iteracích, které se nazývají „rozvíjející cykly“ se postupně dopracovává zbylá funkcionality. Počet iterací je závislý na velikosti projektu.

Každý cyklus se dle [5] skládá ze čtyř fází, kde v závorkách je uvedeno typické rozdělení doby pro jednotlivé fáze úvodního vývojového cyklu:

- Zahájení (10%)
- Projektování (30%)
- Realizace (50%)
- Předání (10%)

Podle [5] je RUP dodáván ve formě internetových stránek, které slouží jako online instruktor, který vývojáře vede celým průběhem vývojového procesu.

Mezi největší výhody RUP patří jeho robustnost, obecnost a přizpůsobivost pro nejrůznější typy projektů. V úvodní fázi při použití RUP je modifikována samotná metodika a to na míru konkrétního projektu. Není tedy problém upravit metodiku pro jakkoli velký tým.

„Díky své rozsáhlosti je RUP přece jen určen spíše větším týmům pro realizaci velkých projektů. Zvládnutí metodiky vyžaduje hlavně v úvodních fázích poměrně hluboké studium a dobře trénovaný vývojový tým“.[5]

4.6. Evoluční modely

Problémem tradičních metodik začala být jejich přílišná složitost a malá flexibilita, takže nestačily reflektovat požadavky nově přicházejících projektů. Tradiční metodiky přestaly být vhodné pro malé projekty a malé týmy kvůli přílišné „byrokratické zátěži“. Příliš často vyžadovaly striktní dokumentaci a nejrůznější revize, inspekce a schvalování. Ani postupné vylepšování těchto metodik nebylo schopno vyřešit zmíněné problémy. Bylo tedy nutné udělat v koncepci vývoje software radikální změnu. Tou změnou bylo právě agilní pojetí vývoje software, kterému se budu v této kapitole věnovat.

4.6.1. Inkrementální programování

Inkrementální programování zavádí nový pohled na programování jako na postupný proces. Mnoho programátorů se snaží psát celý program najednou v jednom kroku. Výsledkem většinou bývá to, že po spuštění programu je vykázano mnoho chyb a program je prakticky nepoužitelný. [1]

Díky tomuto faktu, bylo vyvinuto inkrementální (přírůstkové) programování, které říká, že celý program je lepší rozdělit do několika samostatných funkčních celků. Tyto funkční celky budou tvořit jakousi kostru pro finální program. Jakmile jsou tyto jednotlivé celky otestovány a funkční, je přidán kód, který tyto části spojuje a výsledkem je finální program. [1]

V inkrementálním programování je využíváno iterací. Po každé iteraci je program spouštěn a testován. Díky těmto přírůstkům je finální systém funkční. [1]

4.6.2. Extrémní programování

Principem extrémního programování je to, že se zákazníkovi dodává nejjednodušší a nejzákladnější verze systému, která funguje. Následně vývojový tým provádí revize programového kódu, unit testy, všichni z vývojového týmu mohou měnit návrh a vylepšovat architekturu. V podstatě kdokoli z týmu může měnit cokoli. [6]

Následující pojmy jsou spíše metaforické, ale vystihují činnosti, které se pod nimi v rámci extrémního programování dělají. [6]

RefaktORIZACE – Při psaní kódu je buďto přidávána nová funkčnost, čili programátor přidává nové řádky programového kódu, ale žádné původní nemaže ani neupravuje. Druhou možností je tzv. refaktORIZACE, kdy naopak programátor přepisuje, upravuje a vylepšuje stávající kód, aniž by přidával novou funkčnost. [6]

PÁROVÉ PROGRAMOVÁNÍ – tento pojem popisuje činnost, kdy jeden programátor zpracovává aktuální novou funkčnost a druhý současně přemýšlí v globálním měřítku, jak tato nová funkčnost zapadne do celkového návrhu a požadavků na systém. Zároveň kontroluje, zda nenaruší architekturu či integritu systému. [6]

PLÁNOVACÍ HRA – vývojový tým sepíše User story (slovní popis požadavku zákazníka), z nichž všech se sepíše závazek na nějakou skupinu funkcností (karty zadání), následují iterace vývoje. [6]

Extrémní programování předepisuje 40-ti hodinový pracovní týden, předepisuje oslavu odevzdání hotového projektu a také předepisuje oslavu v případě zániku projektu. [6]

4.6.3. Agilní programování

Agilnímu přístupu se bude věnovat následující část práce, tudíž v této části jsem zařadil pouze stručné shrnutí základních faktů, kterými se agilní programování řídí.

V agilním programování jde o rychlé dodávky funkčních částí systému zákazníkovi. Zákazník rychle zjišťuje, co vlastně chce. V agilním programování hraje významnou roli komunikace mezi zákazníkem a vývojovým týmem. Funkcionalita se může během procesu vývoje měnit. Čas a zdroje na projekt jsou fixní. Rozdíl je v tom, že tradiční metodiky tyto zásady vnímaly opačně. [6]

4.7. Agilní přístup k programování a SCRUM v praxi

4.7.1. Proč vyvíjet agilně?

Při použití neagilních metod vývoje se může stát, že Váš softwarový projekt trvá mnohem déle, než se původně předpokládalo nebo stál dvakrát tolik peněz, než se čekalo. Dalším nepříjemným zjištěním může být to, že po dvou letech práce přijdete s výsledným řešením a klient řekne: „No jo, ale takhle už to nepotřebujeme!“, případně: „Díky, to je moc pěkné ... hmm ... A nešla by tady ta drobnost udělat jinak?“ a ve výsledku můžete téměř celou svou práci zahodit a začít znovu.

Problém je v tom, že klient ze začátku projektu vlastně vůbec neví, co doopravdy chce. Klient ví, že chce mít nějaký software, který bude plnit nějaké jeho požadavky, ale vlastně ještě nemá představu, jak to bude vypadat, on jenom ví, že mu onen software má ulehčit práci.

Na začátku bych měl asi vysvětlit pojem agilita a vysvětlit, co to vlastně agilní programování je. Podle odborné terminologie je agilita použití time-boxovaného iterativního a evolučního vývoje, adaptivního plánování, evolučních dodávek a dalších hodnot a technik, které podporují čilost – rychlou a pružnou odezvu na změny. Tento termín sám o sobě většině lidí nic neřekne. Tuto definici bych shrnul do dvou vět, podle kterých se dá agilní programování pochopit snáze. První věta je: „Motto: změna je vítána!“ – tato věta v normálním programování, při využití např. vývojového procesu vodopád, není příliš vítána. Programátor dostane přesně nalinkovaný úkol, který má splnit, na nic se již nikoho neptá a úkol splní. U agilního programování to ovšem tak není. U agilního programování programátor není jen člověk, který přepisuje příkazy do zdrojového kódu dle zadání. Není to stroj, který také potřebuje změnu k tomu, aby se z programování nestala stereotypní záležitost.

Druhou větou je: „Strategie: co největší manévrovatelnost!“. Tato věta sama o sobě nic nevypovídá. Jedná se ale o snahu o co největší rychlost změny o rychlou změnu vývoje. Potřebuje vývojář rychlou změnu vývoje? Samozřejmě, že ano! Stále se pracuje s tím, že se něco mění. Nic není „nalinkováno“ na pevně s tím, že to a to bude hotovo přesně za 14 dní, pak se tento proces ukončí a jde se dál.

Zkusím to přeložit ještě do jiných pojmů a vysvětlit agilní programování trochu jinak. Agilní programování klade důraz na průběžnou komunikaci mezi vývojovým týmem a zákazníkem. Jedná se přitom o řízenou komunikaci. Zákazník je vtažen do vývoje. Zákazník přitom nemusí chápat, co se přesně programuje, či jaký kód se skrývá pod tím či oním tlačítkem. On pouze vidí, že nyní je hotovo políčko, do kterého např. zadá státní poznávací značku automobilu, které právě projede vrátnicí.

Druhou věcí je důraz na tvorbu kvalitního kódu a funkcí, které mají přímou obchodní hodnotu pro zákazníka. Zákazník si řekne – „Potřebuji udělat toto a toto“ – no a vývojový tým jde a rovnou dělá tu a tu funkci, kterou může rovnou zákazníkovi prezentovat.

Agilní programování je založeno na týmové spolupráci a samo-organizaci týmů. Není to o tom, že si každý programátor dělá svoji část produktu, a když udělá chybu, tak je to pouze jeho problém, jelikož do dané části kódu vidí pouze on. Agilní přístup je to o tom, že všichni programátoři dělají vše a týmová spolupráce je na prvním místě.

Velmi důležitou částí je co nejčastější předávání hotové práce. Nikdy se nemůže podařit, že celková práce bude hotova za 8 měsíců. Nebude! Vždy se někde něco nevydaří a termín se prodlužuje. K této části se vrátím dále v této práci.

V agilním programování jsou změny vítány jako příležitost být lepší. Narážím nyní na přílišnou zkosnatělost tradičních modelů, která nevede k tomu, aby každý programátor do projektu dával to nejlepší ze sebe sama.

Poslední definicí agilního programování je důraz na výslednou hodnotu pro zákazníka před dokumenty a papírováním. To je dobrá zpráva pro programátory, protože programátoři se ne tolik věnují dokumentaci, ale spíše výslednou hodnotou pro zákazníka.

5. SCRUM

SCRUM je agilní metodika, která se hodí do týmů od čtyř do patnácti lidí. V ideálním případě by měli být na jednom místě (v jedné místnosti), ale vyskytují se i případy, kdy se Scrum provozuje napříč světem. Samotná metodiky se rozšířila na začátku devadesátých let a u jejího vzniku stáli Ken Schwaber a Jeff Sutherland, kteří základně nastavili délku iterace na 30 dní a zavedli backlog a další artefakty, které budou zmíněny dále v textu.

Scrum je opředen mnohými mýty. Prvním mýtem je to, že scrum urychluje vývoj. To není pravda. Scrum vývoj nikdy neurychlí. Vývoj se musí provést ať se jedná o tu či onu techniku. Jedinou výhodou je to, že scrum vývoj zprůhledňuje a tím zvyšuje efektivitu. Ale každopádně žádný programátor zdrojový kód nenapíše rychleji použitím této metodiky.

Druhým mýtem je to, že se SCRUM nehodí zrovna na „můj“ projekt. Druhá nepravda! Scrum se hodí na jakýkoli projekt a dále dokážu proč. Třetí mýt je ten, že scrum je vhodný pouze tam, kde lze rychle předvést něco v GUI. Tento mýt pozbývá smysl, protože se nikdy nepředvádí GUI, ale již hotová funkčnost. Poslední častým mýtem je to, že se spousta času proschůzuje. Scrum je hodně založen na komunikaci. Každý den se prování patnácti minutová schůze, kde se prezentuje, kdo co za minulý den udělal a co bude dělat dnes. To dává dohromady za týden hodinu a čtvrt. Cena hodiny a čtvrt času za to, že každý v týmu ví, co kdo dělá je přijatelná.

Scrum se poprvé objevil v roce 1983 a používá se při vývoji zakázkového software, krabicového software, interních nástrojů, aplikací certifikovaných ISO 9001, finančních aplikací, informačních nástrojů. Milion critical aplikací, software pro mobilní zařízení a handheldy, webové stránky a videohry. Používají jej společnosti jako Microsoft, Google, Yahoo, Lockheed Martin, Phillips, Nokia, BBC, Time Warner a mnohé další.

5.1. Role v týmu

V týmu scrumu se musí objevit všechny profese - analytici, programátoři, testeři a QA (Quality Assurance) – tito členové jsou nazýváni jako Scrum team member (STM). Je to z toho důvodu, aby všichni věděli, co kdo dělá a tým tak funguje jako celek. Tým by měl být dále ucelený po celou dobu vývoje. Není doporučováno, aby v týmu působili externisti. Je lepší mít v týmu jednoho člověka naplno než dva na půl. Další důležitou rolí je pozice Scrum Mastera. To je člověk, který působí jako manažer projektu, ale není to hlavní šéf projektu. Scrum Master dbá na hodnoty a postupy. Je to takový firewall týmu, který odstraňuje překážky. Poslední a podstatnou částí týmu je tzv. „product owner“. Je to osoba, která je buď reálná nebo fiktivní a dohlíží na projekt a prosazuje požadavky zákazníků. Či je product owner reálný či fiktivní záleží na typu společnosti a produktu, který vyvíjí. Například ve společnosti vyvíjející informační systém pro tisíce zákazníků není zákazník jenom jeden a proto jej zastupuje osoba z týmu či z vedení společnosti – fiktivní zákazník.

PRODUCT OWNER - je osoba, která zodpovídá za priority, za to, co se bude v příštím sprintu (viz dále) implementovat a určuje implementační detaily.

SCRUM MASTER - pracuje jako ten, kdo má programátory odstínit od okolního světa. Řídí vývojáře, ale zároveň se stará o to, aby jim fungovaly počítače, měli dostupný software, který potřebují, řeší spory apod. Scrum Master a manažer se mohou krýt, zároveň může být Scrum Master i analytik. Není vhodné, aby byl Scrum Master zároveň programátor, v takovou chvíli by nebyl schopen odfiltrovávat programátory od rušivých vlivů – byl by sám zranitelný.

SCRUM TEAM MEMBER (STM) - člen vývojového týmu. Může se jednat o vývojáře, testera, analytika, dokumentaristu,...

5.2. Sprint

V této části vysvětlím, co to je sprint. Každý projekt se dělí na časové jednotky – tzv. sprinty. Sprint trvá typicky 1 – 4 týdny. Délka sprintu by měla být konstantní kvůli časovým odhadům. V rámci jednoho sprintu je malá část produktu specifikována / navržena, naprogramována a otestována v rámci 1 sprintu. Čím je sprint kratší, tím je větší možnost na změnu. Během sprintu může úkoly přidávat / ubírat pouze tým a nikdo zvenku.



5.3. User story

Jedná se o slovní popis požadavku zákazníka. US jsou vytvářeny Product ownerem. Po zahrnutí US do konkrétního Sprintu by se zadání US nemělo měnit. Všechny sepsané User Stories se sepíší pod sebe do takzvaného Product Backlogu (zásobníku).

5.4. Product backlog

Obsahuje všechny scénáře, které systém zahrnuje. Následně se Product Backlog seřadí podle priority (tuto činnost provádí Product Owner). Jednotlivým úkolům se přiřadí časové náročnosti (používá se bezrozměrná jednotka, tzv. body složitosti). To proto, že na začátku projektu není známa rychlost týmu. Do prvního sprintu se pokusí programátoři odhadnout, kolik úkolů dovedou z Product Backlogu stihnout během jedné iterace (tzv. sprintu). Úkoly se odebírají od nejdůležitějších. První jednu až dvě iterace se vývojáři samozřejmě netrefí – není totiž známa rychlost týmu. Na to se využívá metodika Poker planningu.

5.5. Poker planning

Jednotlivé úkoly se ohodnocují body podle náročnosti. K tomuto účelu využíváme klasických hracích karet, kdy každý z členů měl k dispozici karty v hodnotě A, 2, 4, 6, 8, 10, K, JOKER. Karty (kromě K a JOKER) značí bodovou hodnotu, Král znamená „nekonečná pracnost“, JOKER znamená "nejsem schopen odhadnout". Členové týmu u každého požadavku naráz ukáží kartu s jejich předpokladem (naráz proto, aby se nemohli ovlivňovat). Důležité je, aby ve výsledku všichni členové týmu souhlasili s výslednou bodovou hodnotou. Tyto hodnoty se zapisují ke každému požadavku a stávají se závaznými.

5.6. Sprint backlog

Jedná se o seznam jednotlivých User Stories, vlastností, které se mají implementovat a přesouvají se z Project Backlogu. Není možné za žádnou cenu umožněno změnit zadání. Změny jsou možné pouze v čase mezi iteracemi. Existují týmy, které Product Ownerovi dovolí vyměnit úkol ve Sprint Backlogu za jiný s identickou obtížností. Motiv toho, že není dovoleno měnit zadání za běhu je naprosto klíčovou podmínkou Scrumu. Programátor se jen díky tomu může soustředit na svou práci, na dosahování cílů a odevzdávání práce v termínu. Pro vývoj v praktické části byla stanovena perioda iterace na 14 dní.

5.7. Daily meeting

Denní střetnutí celého týmu (každý den od 8:30, max. 15 minut). Tato střetnutí jsou pro metodiku klíčová. Na střetnutí každý ze členů prezentuje, co dělal předešlý den, co dělá dnes a s jakými problémy se setkal. Pokud se vyskytnou nějaké problémy, které zabraňují dalšímu pokračování v práci, Scrummaster nebo ostatní členové týmu udělají potřebné kroky k jejich odstranění. Setkání se může zúčastnit více lidí (třeba Product Owner), není jim ale dovoleno mluvit – to smí pouze tým a Scrum Master. Jak sprint pokračuje, ubývají úkoly ve Sprint Backlogu. Důležité je, že na této schůzce se případné problémy pouze identifikují, jejich řešení probíhá mimo scrum. Důvod je logický – nebudeme zdržovat ostatní členy týmu záležitostmi, které se jich přímo nedotýkají. Scrumy se provádějí ve stoje. Důvod je jednoduchý, relativně snadno se tím zabrání tomu, aby se účastníci scrumu zbytečně rozpovídávali.

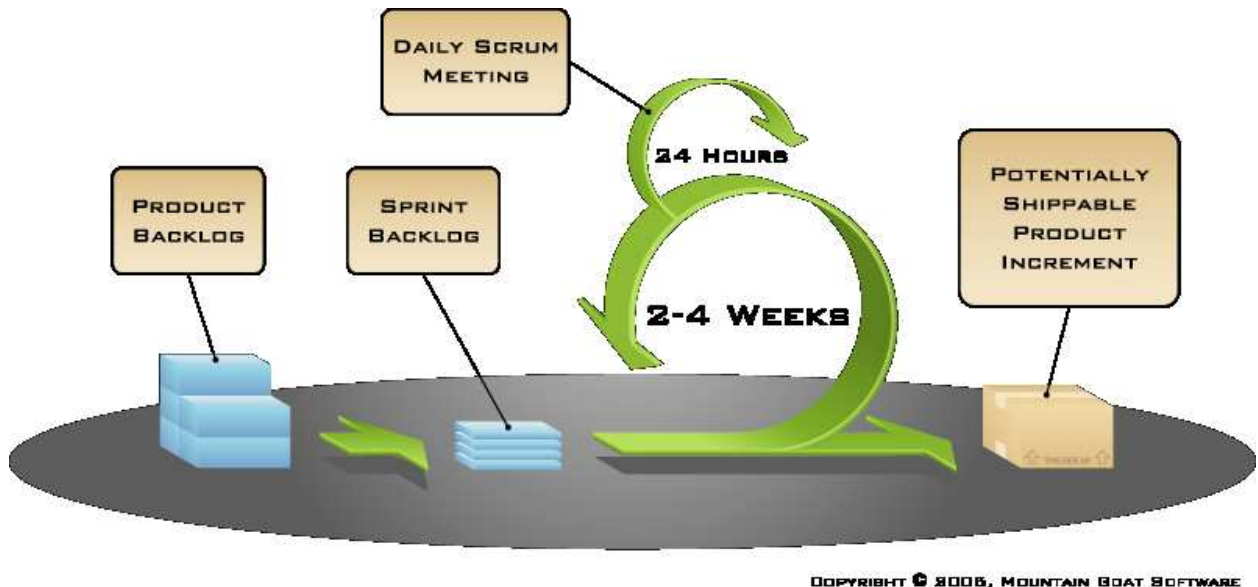
Jedná se o každodenní setkání týmu a je moc důležité z důvodu, aby každý v týmu věděl, co kdo dělá. Jedná se empirický proces, komunikaci týmu, zpětnou vazbu a zprůhlednění činnosti týmu. Denní schůzka musí mít jasná pravidla, co se týká místa a času konání. Například každý den na stejném místě a ve stejný čas. U nadnárodních společností se může stát problémem časový posun. Například v Evropě se koná schůzka v osm hodin ráno, ale v Americe se již jedná o polovinu pracovního dne. Je důležité provádět tuto schůzku ve stoje, jelikož člověk, který stojí a má něco povídat se vyjadřuje lépe. Cílem této schůzky je odpovědět na následující otázky. Co jste dělali od poslední denní schůzky (daily scrum)? Co budete dělat do dalšího? Co stojí v cestě k cíli iterace? - jaké mají jednotliví členové týmu problémy. K poslední otázce se vztahují i dvě nepovinné podotázky. Jsou nové položky do backlogu? Naučili jsme se něco nového? K otázce backlogu se dostaneme dále. Jedná se tedy o stručné shrnutí minulého a následujícího dne.

5.8. Workflow

Projekt je rozčleněn do jednotlivých User Stories, bude zvoleno, které User Stories jsou předmětem sprintu a bude stanoven termín dokončení. Tento termín je označen jako Time Box. Do konce Time Boxu musí být požadované vlastnosti implementovány. Odhady se stanovují tak, aby byla práce dokončena přesně v den odevzdání, tedy nenechávají se ani rezervy, ani se nečeká rychlejší práce. Tento postup vyplývá z myšlenky “dáte-li týmu jakýkoliv termín, tým čas vždy celý vyčerpá.”.

Pokud existuje nutnost práce i na jiných projektech, je možné to řešit tak, že se vyhradí například 6 hodin denně pro práci na sprintu a 2 hodiny na práci na jiných úkolech. Nezasahování do vyhrazeného času vývojáře, vedou k tomu, že lze velmi zpřesnit časové odhady toho, co vývojář stihne, jaká je jeho aktuální rychlost a zda situace nasvědčuje tomu, že své úkoly stihne odevzdat do konce sprintu.

Další podmínkou agilních metodik (a tedy i Scrumu) je zákaz práce přesčas. Odpočínutý zaměstnanec pracuje lépe, dělá méně chyb a dokáže se snáze soustředit na svou práci.



Obrázek 6: Workflow

5.9. Body složitosti

Body složitosti nebo Story Point (SP) je označení základní jednotky pro odhady pracnosti. V praktické části bylo pro prvotní odhady stanoveno $1 \text{ SP} = 4 \text{ hodiny}$.

5.10. Velocity týmu

Jedná se vlastně o výkonnost týmu. Ta se počítá v bodech složitosti. V případě praktické části se přepočítává na body, které reflektují složitost úkolu, která se určí na plánovacím meetingu pomocí hodnotícího pokeru. Vlastní velocita pak říká, kolik těchto bodů jsme schopni zrealizovat během jedné iterace (sprintu).

5.11. Burndown chart

Graf závislosti počtu SP v období daného Sprintu. Graf má sloužit pro Scrummastery k tomu, aby dokázal lépe sledovat průběh vývoje.



Obrázek 7: Burndown chart

V tomto ukázkovém Burndown Chartu bylo dodrženo naprosto přesně zadání a tým celou dobu pracoval tak, jak se od něj očekávalo. Takový Burndown Chart ale nebývá obvyklý. Většinou dochází v průběhu práce k různým odchylkám, některé úkoly se daří řešit rychleji, jiné pomaleji.

Ukažme si burndown chart, kdy se nejdříve pracovalo rychleji, pak ale došlo k výraznějšímu zpomalení a tým nestihl odevzdat práci tak, jak očekával:

Většinou se burndown chart zobrazuje ve 2D, ale klíčový je pouze pro určení, jak rychle se vyvíjí. Je z něho patrné, zda je potřeba upravit Project Backlog, nebo zda nastaly nějaké problémy. Lépe je to vidět na následujícím grafu (zdroj: Wikipedia)



Obrázek 8: Sample Burndown chart

Takovou situaci by měl Scrum Master řešit tak, že vypočítá, jaká je rychlost (Velocity) týmu, dále by pak vrátil některé úkoly ze Sprint Backlogu do Project Backlogu tak, aby byly na konci hotovy všechny úkoly.

Pokud tým pokračuje příliš rychle, přidá Scrum Master další úkoly z Product Backlogu do Sprint Backlogu. Pokud je tým příliš pomalý, řeší Scrum Master důvody, proč se nedaří pracovat stanovenou rychlostí. Nelze-li zrychlit, Scrum Master přesune úkoly zpět ze Sprint Backlogu do Product Backlogu tak, aby rychlost práce vycházela přesně na termín Sprintu. Tyto přesuny nesmí dělat nikdo jiný – zejména ne Product Owner, tedy zákazník!

5.12.Planning session

Po dokončení sprintu se provádí tzv. planning session (většinou v pátek). Jedná se o celodenní meeting všech lidí z týmu, který se rozděluje na 3 části:

5.12.1.Demo

V první části se předvádí produkt „klientovi“. Vlastnosti, které se nestihly dokončit, jsou skryty, klient tedy uvidí pouze dokončenou práci. Na jejím základě se rozhodne, co se bude dělat dál. Klient ví, které úkoly se nestihly a může se rozhodnout, zda se dokončí (což se stává většinou, protože tyto úkoly bývají obvykle rozpracovány a jejich opuštění by znamenalo smazat práci na nich strávenou).

5.12.2.Retrospekce (retrospektiva)

V této části se každý vývojář v týmu vyjádří jaké shledává na uplynulém sprintu plusy a mínusy při vývoji. Scrum Master provádí okamžité vyhodnocování, reaguje, případně si zapisuje věci k dodatečnému řešení a provádí jejich realizaci.

Jedná se o vyhodnocení celého procesu uvnitř celého týmu. Zkoumá se co funguje, co nefunguje, jak co dělat či nedělat. Tým je v podstatě samo-řídící. Tým funguje na bázi demokracie v tom, že v týmu není klasický šéf, který určuje, co kdo bude kdy dělat. Jedná se v podstatě o tým bez vlastníka a zkoumá projekt z hlediska metodiky jednou za 2-4 iterace. Každý v týmu se může vyjádřit k tomu, co funguje a co nefunguje a zda to ponechat či vyřadit. Jedná se též o sběr nových nápadů na zlepšení a jak je uvést v život.

5.12.3.Plánování náplně dalšího sprintu

Jedná se o zaplánování všech User Stories požadované Product Ownerem a dalších k naplnění nadcházejícího Sprint Backlogu. Tuto činnost provádí SCRUM MASTER ve spolupráci s týmem. Na závěr si každý člen v týmu vybere podle priorit konkrétní User Story, který bude řešit.

5.13. Aktivita – plánování

Aktivity, které probíhají v rámci scrumu si nyní přiblížíme. Plánování probíhá tak, že se plánuje v tzv. bodech složitosti. Neplánuje se v čase! Toto je docela zásadní. Důležité je zjistit rychlost vlastního týmu. Není možné říci, že pokud dám 5 programátorům stejný úkol, tak ho všichni splní za stejnou dobu. Není přitom podstatné, zda programátor udělá úkol za dvě či tři hodiny, je důležité ale vědět, kolik je tým schopný odevzdat práce za určitý časový úsek. Tyto údaje se zprůměrovávají. Čas je konstantní. Například dva týdny jsou deset pracovních dnů krát 8 hodin je 80 hodin. Když se toto vynásobí počtem lidí v týmu, tak dostaneme časový fond, který máme k dispozici. To ale nic nevypovídá o tom, kolik se udělá práce. Víme pouze, že máme nějaký čas. Do toho fondu něco může zasáhnout – školení, nemoc, cokoli. Ve výsledku čas není tolik podstatný jako body složitosti. Dle výkonnosti a efektivnosti týmu se každému jednotlivému úkolu přiřadí body složitosti. Plánování probíhá na začátku projektu a poté na začátku každého sprintu. K plánování se dá užít tzv. plánovací poker. Jedná se o způsob odhadování pracnosti. Každý programátor má na stejný úkol jiný pohled. Jeden úkol přeceňuje, druhý podceňuje. Ve výsledku tým odhaduje nezávisle na sobě pracnost jednotlivých úkolů. Pro jednotlivý sprint by nemělo plánování trvat déle než 30-60 minut.

6. Software maintenance

Software maintenance se dá přeložit jako softwarová údržba. V životním cyklu softwaru zaujímá závěrečnou část. Údržba bývá často podceňována, přestože podle některých zdrojů pokrývá téměř 80% času v životě informačního systému. V této fázi se zhodnocuje úsilí vývojářů vytvořit specifikaci, návrh a dokumentaci softwarového produktu.

Software maintenance je hlavní proces změny systému poté, co byl naimplementován a zprovozněn u zákazníka. Změny mohou být jednoduchého charakteru – oprava chybného programování, složitějšího charakteru – oprava návrhu softwaru a nakonec mohou být změny zásadní – oprava specifikace softwaru nebo začlenění nových požadavků na informační systém. Samotné změny obvykle nenaruší hlavní jádro softwaru, běžně jsou aplikovány úpravy existujících komponent či přidáváním nových komponent.

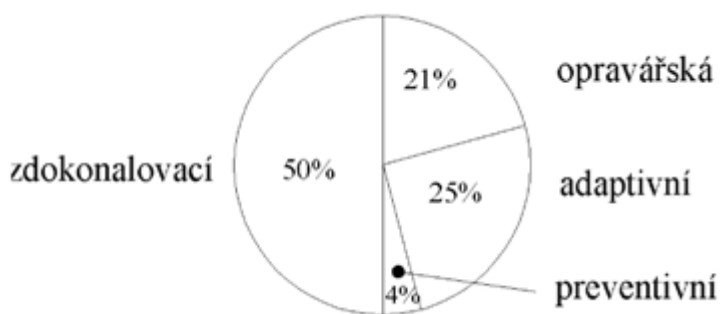
Existují různé typy softwarové údržby[17].

1. Opravování softwarových chyb. Programové úpravy jsou většinou levnější než opravy designu. Ty jsou většinou nákladné, protože se musí zasáhnout do více programových komponent najednou. Nejvíce nákladné na opravu jsou opravy vzniklé na základě požadavků na systém, jelikož je potřeba přestavit základní stavbu informačního systému. [17]
2. Adaptivní údržba. Tento typ údržby je nutný v případě, kdy se změní prostředí, kde informační systém pracuje. Může se jednat o změny v hardwarové konfiguraci, změnu operačního softwaru nebo změna dalších podpůrných aplikací. Tato údržba žádným způsobem nemění funkčnost či chování informačního systému. [17]
3. Zdokonalovací údržba. Údržba zaměřená na změnu funkčních požadavků ze strany zákazníka/uživatelé. Vede k funkčním zlepšením systému. Tato údržba je vyžadována při změně organizačních či business požadavků na systém. Zásah do systému je v tomto případě největší ze všech typů údržby. [17]
4. Preventivní údržba. Tato údržba zahrnuje aktivity zaměřené na zvýšení výkonu systému, jeho udržitelnost a optimalizace. [17]

V praxi bohužel nelze tyto typy údržby tak snadno oddělit. Softwarová chyba může být způsobena například tím, že systém byl použit trochu jiným způsobem, než jakým byl navržen. V této chvíli je jednodušší a méně nákladné pro opravu chyby přidat novou funkcionalitu. Když software přizpůsobujeme novému prostředí, tak při této údržbě může vývojář navrhnout i nové funkcionality, které dokáží toto nové prostředí lépe využít a usnadnit tak práci koncovým uživatelům. Přidání nové funkcionality se například může stát nezbytné v případě, kdy kvůli chybě uživatelé začnou systém používat jiným způsobem.

Změna prostředí informačního systému stejně jako uživatelských požadavků je nevyhnutelná. V dnešní době informační systém v podstatě modeluje část reality a ta se mění. S postupem času každý software stárne a je potřeba jej dále rozvíjet. Z tohoto pohledu se dá říci, že zapracováváním uživatelských požadavků v rámci údržby můžeme mluvit prakticky o vývoji software.

Na následujícím grafu je znázorněno, kolik ze software maintenance pokrývají jednotlivé typy údržby [17].



Obrázek 9: Podíl jednotlivých typů údržby

Je velice těžké říci, že výše uvedený graf je přesným vyjádřením procentuálního rozdělení software maintenance. Z různých zdrojů je zřejmé, že se procentuální vyjádření liší. Jiní autoři uvádějí, že zdokonalovací údržba pokrývá celých 65%. Dle mého názoru je nejlepším rozdělením poměr 50:21:25:4, který je znázorněn na obrázku. Vycházím z praktických zkušeností, které jsem získal více než dvouletou praxí v oblasti software maintenance, kterou se budu zabývat v praktické části této práce.

6.1. Software maintenance a náklady společnosti

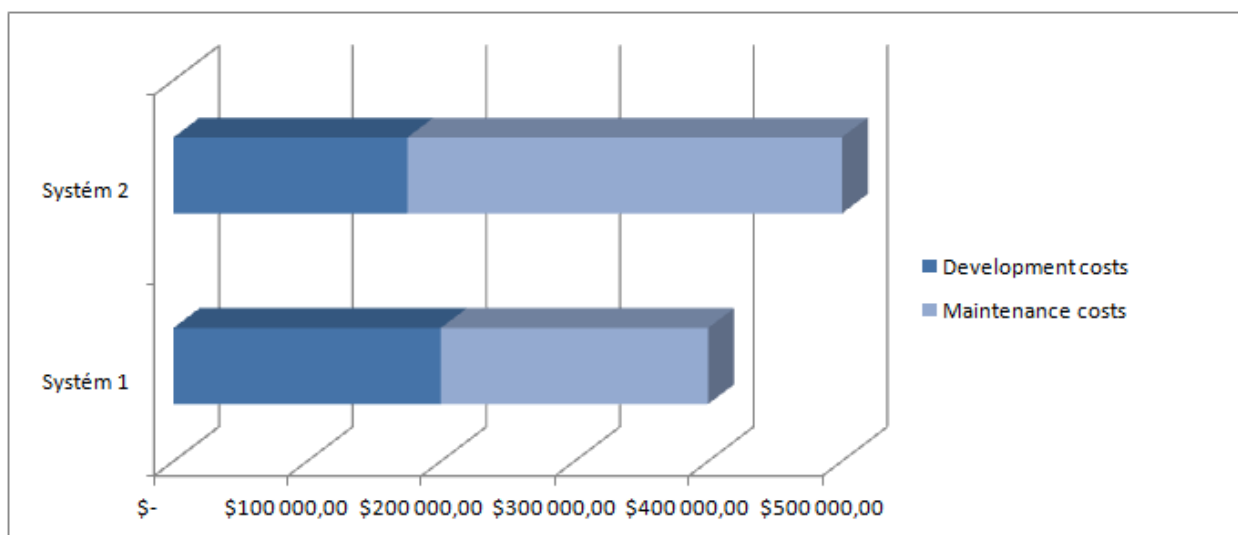
Z obrázku výše lze přejmout i rozložení nákladů společnosti na softwarovou údržbu. Lze vypořádat, že opravárenské činnosti nejsou až tak významné (nákladné) jako náklady na zdokonalovací a adaptivní údržbu. Navíc musíme vzít v úvahu i fakt, že náklady na opravárenskou činnost jsou často kryty smlouvě o údržbě, která deklaruje, že chyby způsobené na straně dodavatele (vývojáře) systému nejsou hrazeny přímo, ale většinou ročním poplatkem za údržbu.

Jak jsem zmínil výše, tak skoro veškeré náklady na maintenance jsou generovány adaptivní a zdokonalovací údržbou. Těchto 75% (50% zdokonalovací + 25% adaptivní) údržby podle některých zdrojů činí přibližně polovinu rozpočtu společnosti vyčleněného na informační systémy – jsou to náklady na rozvoj existujících informačních systémů společnosti. [17]

Zajímavý je fakt, že podle některých zdrojů mohou v průběhu života informačního systému ve společnosti dosáhnout náklady za maintenance až čtyřnásobku sumy za jeho pořízení a vývoj.

Ve většině případů implementace informačních systémů je moudré a méně nákladné věnovat úsilí přesně definovat požadavky na systém a definování funkcí systému již při výchozí analýze. Je totiž většinou více nákladné přidávat funkcionalitu poté, co je informační systém plně naimplementován a používán ve společnosti. [17]

Na následujícím obrázku je dobře znázorněno, jak se vyplatí investovat úsilí při samotném vývoji informačního systému než následně platit za zdokonalovací údržbu. Obrázek vychází z faktu, že následná údržba může být až čtyřikrát více nákladná než náklady při samotném vývoji. Přitom u obou případů dosáhneme úplně stejného výsledku. [17]



Obrázek 10: Náklady společnosti

Z obrázku je vidět, že u Systému 1 jsou náklady na vývoj o 25 000 \$ dražší, ale celkové náklady na vývoj a maintenance jsou o 100 000 \$ nižší než u Systému 2. Zde se sice ušetřilo 25 000 \$ na vývoji, ale celkové náklady vzrostly o 100 000 \$.[17]

Důležitým faktem tedy je, že náklady na následnou zdokonalovací údržbu jsou mnohem dražší než zapracování té samé funkcionality při samotném vývoji. Klíčové faktory, které odlišují vývoj od údržby, a které jsou důvodem k vyšším nákladům na údržbu, se pokusím popsat v následujících pár odstavcích.

1. Stabilita vývojového týmu. Běžně se stává, že tým, který určitý informační systém vyvíjí na straně dodavatele, je po dokončení tohoto projektu rozpuštěn a jeho jednotliví členové se již věnují jiným projektům. Bývá také časté, že lidé, kteří systém vyvíjeli, nejsou ve finále i ti, kteří ho následně udržují. Nový tým nebo jednotlivci, kteří jsou následně zodpovědní za údržbu, zcela nerozumí tomu, jak je systém naprogramován či tomu, proč se při jeho vývoji postupovalo tak či onak. Mnoho zbytečných nákladů na údržbu se dá tedy ušetřit, pokud se „údržbář“ napřed dokonale seznámí se systémem předtím, než začne implementovat změny. [17]
2. Smluvní odpovědnost. Smlouvu o údržbě lze uzavřít dvěma způsoby. Prvním je ten, že smlouva o údržbě je uzavřena s dodavatelem samotného systému. Druhým způsobem je uzavření smlouvy o údržbě s jinou, externí, společností, která u samotného vývoje nebyla. Pokud původní dodavatel je informačně uzavřen vůči další společnosti poskytující údržbu, tak náklady na zjištění vlastní funkčnosti a fungování informačního systému „uvnitř“ mohou být velice nákladné, jelikož zákazník neplatí pouze za vlastní údržbu, ale i za čas, který nová firma stráví analýzou již funkčního systému. [17]
3. Zkušenosti zaměstnanců. Máme tým na mysli zkušenosti zaměstnanců dodavatele informačního systému, případně zkušenosti externí firmy na údržbu. Údržba je bohužel chápána jako podřadná činnost ve srovnání s jeho vývojem. Většinou je tedy práce na údržbě přenechána mladším zaměstnancům dodavatele – mladšími zaměstnanci mám na mysli programátory na juniorských pozicích. Jejich práce je díky jejich nízkým zkušenostem často málo efektivní a hodně chybová. Čas, který by opravě věnoval zkušený programátor je často mnohem nižší než doba, kterou na ní stráví junior programátor. [17]
4. Stáří systému a jeho struktura. S postupem času software stárne. Při následných úpravách systému je tendence jeho zastaralou strukturu nahrazovat moderními trendy v programování a využívá se moderních metod softwarového inženýrství. Tím se brzy stane systém strukturně nepřehledný a je velice náročné do něj implementovat další změny. Dalším faktorem může být i to, že softwarová dokumentace se může ztratit či jeho vypovídací schopnost je vlivem nových změn již nulová. [17]

První tři faktory, které zvyšují nákladnost údržby, jsou způsobeny tím, že hodně organizací vidí rozdíl mezi vývojem software a jeho údržbou. Na údržbu je pohlíženo jako na druhotnou aktivitu a není zde snaha investovat peníze navíc při vývoji k redukování následných nákladů při údržbě. Poslední bod – stáří systému – se dá vyřešit moderními metodami re-inženýringu, ale toto téma přesahuje rozsah této práce.

6.2. Software maintenance jako proces

Již jsme si vysvětlili, že údržbu je nutno chápat jako proces. Tento proces je vysoce závislý na typu systému, na kterém je údržba prováděna. Většina požadavků na údržbu vzniká přímo u uživatelů a jejich komunikace s vývojáři informačního systému. Každá údržba, ať se jedná o opravnou, zdokonalovací, adaptivní či preventivní má fundamentálně stejné procesy a aktivity. Vždy se jedná o změnovou analýzu nového požadavku, plánování, vlastní opravu a implementaci změny + testování a prezentace nové funkčnosti (opravy) koncovému uživateli.

Proces údržby je spouštěn vznesením požadavku na údržbu od uživatelů či managementu společnosti, ve které je informační systém nasazen. Každý požadavek je analyzován za účelem zjištění nákladů (závislé na pracnosti) na údržbu a dopadu na chování systému. Když je ze strany zákazníka údržba schválena, začíná proces plánování. V procesu plánování – ať se jedná o opravnou, zdokonalovací, adaptivní či preventivní údržbu – jsou zvažovány všechny změny, které je nutné udělat pro požadovanou funkčnost. Když se rozhodne o implementaci změn, jsou změny zaplánovány do další verze systému. Poté jsou změny implementovány, otestovány a jsou distribuovány do nové verze systému. V ideálním případě je po implementaci těchto dílčích změn systém testován jako celek. V této fázi se zhodnotí, zda provedené změny neovlivnily chování systému (systém je podroben celkovým testům – často označováno jako kompilační kontrola) a je distribuována nová verze systému zákazníkům. [17]

Požadavky na údržbu většinou přicházejí velice rychle a pochází z těchto tří základních důvodů [17]:

1. Nutná údržba kvůli chybě, která znemožňuje standardní práci se systémem.
2. Změna prostředí, která negativně ovlivnila chování systému
3. Nepředvídatelné změny business prostředí, ve kterém systém působí, které mají nepříznivé důsledky na systém – vznikají z důvodu změny legislativy, či je potřeba změnu implementovat z důvodu zachování konkurenceschopnosti.

V předchozích třech případech je většinou nutné provést údržbu rychle než se držet formálního postupu údržby. Tyto změny jsou většinou implementovány do systému rychle přímo programátorem bez důkladné předchozí analýzy. Údržba je sice provedena rychle, ale bohužel se může stát to, že programový kód a struktura systému se pomalu stává nepřehlednou a každá další změna je komplikovanější než ta předchozí. Dle teorie by každá změna měla být provázena analýzou a důkladným prověřením dopadů na systém – to ale způsobuje zdržení a případné urgentní problémy nejsou řešeny včas. Tento problém je možné řešit udržováním dvou verzí systému. Jedna – distribuční – která je rychle distribuována zákazníkům s rychle provedenými změnami a druhá – záložní, ostrá – kde jsou změny zapracovány pečlivěji a tato verze je distribuována zákazníkům méně často než verze distribuční. Touto problematikou se budu zabývat v praktické části této práce.

7. Normy a standardy

7.1. Český standard životního cyklu

„V České republice byl v prosinci roku 2002 vydán Standard ISVS 005/02.1 pro náležitosti životního cyklu informačního systému, který vydal Úřad pro veřejné informační systémy v Praze. Standard se vztahuje na IS veřejné správy a na projekty akvizice, vývoje, provozu a údržby těchto systémů a věnuje se především dokumentaci a jednotlivým krokům, které musí být provedeny při vývoji IS. Standard definuje základní postupy a náležitosti procesů životního cyklu informačního systému nebo jeho části s hlavními cíli:“[20]

- zajistit řízení a zejména kvalitní organizaci a kontrolu projektů rozvoje IS v rámci organizace správce ISVS [20]
- zajistit kvalitní řízení vývoje, provozu a údržby informačního systému jako celku [20]
- připravit věcný podklad pro atestace informačních systémů nebo projektů jejich rozvoje [20]
- poskytnout správcům informačních systémů veřejné správy možnost efektivněji kontrolovat dodavatele komplexních řešení (zejména externí subjekty) rozšířením počtu a přesnou definicí kontrolních bodů, ve kterých lze zpracováváný projekt ovlivnit nebo v krajním případě zastavit tak, aby nedocházelo k dalším zbytečným finančním a časovým ztrátám [20]
- vést jednotnou strukturovanou dokumentaci jednotlivých projektů tak, aby nebyly ohroženy personálními změnami v řešitelských týmech [20]

7.2. ISO/IEC 12207

„ISO/IEC 12207 je mezinárodní standard určený pro softwarové procesy v rámci životního cyklu (Software Life Cycle Processes). Tento standard byl poprvé navržen v roce 1988 a publikován v roce 1995 a stanoven jako společný mezinárodní rámec pro vývoj, dodávky, podporu a údržbu softwaru. Standard se věnuje především třem zásadním procesům: procesům primárního životního cyklu, podpoře procesů životního cyklu a organizačním procesům životního cyklu.“[20]

Procesy ISO 12207 jsou určeny pro oblasti nákupu, provozu a správy systémů (včetně softwarových) bez ohledu na to, zda je firma sama vytváří nebo je nakoupila od výrobce nebo třetí strany.[14]

Stejně jako všechny současné procesní normy, i standard ISO 12207 nepředepisuje a dokonce ani nedefinuje hotové procesy, které by bylo možné vzít a použít (jak to dělají komerčně vytvářené metodiky), ale vytváří procesní rámec, který slouží jako předloha pro vytvoření vlastních procesů. Procesy musí být upraveny pro podmínky organizace i konkrétního projektu.[14]

7.3. ISO/IEC 15504

„ISO/IEC 15504 Software Process Improvement and Capability Determination, známý pod zkratkou SPICE, je mezinárodní rámec pro hodnocení procesů, který byl vyvinut Joint Technical Subcommittee v období mezi standardem ISO (International Organization for Standardization) a IEC (International Electrotechnical Commission).“[20]

„Dimenze procesů jsou rozděleny do pěti kategorií procesů na dimenzi zákazník/dodavatel, inženýrství, podpora, řízení a organizace. Vyspělost procesů (capability of processes) je měřena pomocí atributů procesu a jsou definovány mezinárodním standardem devíti atributy jako je výkonnost procesů, řízení procesů, dále definice, rozmístění, měření, kontrola, inovace a optimalizace procesu. Každý z těchto atributů je hodnocen pětibodovým hodnocením od nedosaženo (not achieved) až po plně dosaženo (fully achieved). Standard poskytuje návod na to, jak provádět hodnocení procesů, obsahuje model pro toto hodnocení a doporučené nástroje používané pro hodnocení.“[20]

„Sestrou či spíše dítětem standardu ISO 15504 je známý Referenční model softwarového procesu ISO 12207, který se od normy oddělil při její poslední revizi v roce 2004. Samotný model ISO 15504 se tak „oprostil“ od definice procesů a lze jej univerzálně použít pro posouzení jakéhokoli procesního modelu. Jeho největší část, ISO 15504-6, definuje univerzální cíle kritéria v podstatě stejně, jako před oddělením referenčního modelu.“[16]

8. Charakteristika podnikatelského subjektu, společnosti J.K.R. spol. s r.o.

8.1. Základní charakteristiky společnosti

Název společnosti:	J.K.R. spol. s r.o.
Sídlo:	Příbram 2, Pražská 14
Pobočky:	Praha, Teplice, Plzeň
IČO:	186 08 001
Zapsáno:	17. května 1991
Právní forma:	Společnost s ručením omezeným
Počet zaměstnanců:	90
Předmět podnikání z OR:	➤ Nákup a prodej výpočetní techniky a elektroniky ➤ Poradenská a školicí činnost v oblasti hardware a software ➤ Nákup a prodej firemních a aplikačních softwarových produktů ➤ Analýza systémů ➤ Tvorba a prodej programového vybavení

8.2. Profil společnosti

„Společnost J.K.R. je s produkty BYZNYS ERP předním českým dodavatelem podnikových informačních systémů ERP. Byla založena v roce 1991 v Příbrami, dnes působí po celé České republice i na Slovensku. V příbramské centrále a v pobočkách v Praze a v Teplicích zaměstnává přes 90 lidí, má externí servisní střediska v České republice a na Slovensku, a disponuje rozsáhlou distribuční sítí. Roční obrat společnosti se pohybuje kolem 80 milionů korun.“ [21]



„J.K.R. se zabývá vývojem a dodáváním podnikových informačních systémů BYZNYS ERP, kde ERP je zkratka pro „Enterprise Resource Planning“, nebo-li plánování podnikových zdrojů. Řešení Byznys ERP jsou určena především pro střední a větší organizace, ale díky vysoké flexibilitě plně vyhoví i potřebám malých firem.“ [21]

„Své kvality v oblasti nasazování a provozu podnikových informačních systémů J.K.R. úspěšně prokazuje více než 20 let. Má tudíž rozsáhlé zkušenosti se zaváděním procesů v oblasti řízení ekonomiky, logistických systémů, zakázkově orientovaných řešení, výroby a dalších oborových řešení spolu s vytvořením uživatelských úprav dle specifických potřeb zákazníků. Důležitými atributy jsou také spolehlivost, solidnost a stabilita společnosti, což v kombinaci s kvalitními produkty a špičkovými službami a servisem zákazníkům zaručuje jistotu a úspěch.“ [21]

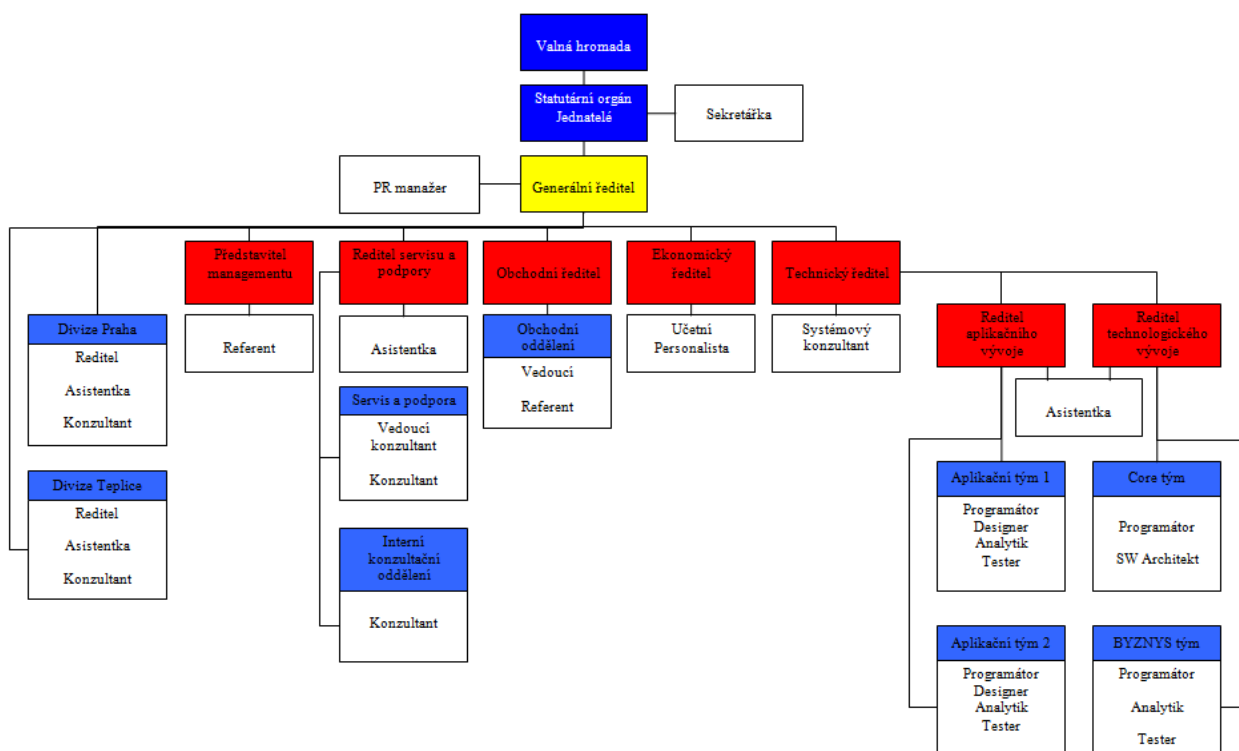
„Základním a stěžejním cílem společnosti je maximální spokojenost uživatele jak s možnostmi systému, tak i s uživatelskou konzultační a metodickou podporou. Vzhledem k udržení vysoké výkonnosti informačních systémů, jsou shromažďovány a vyhodnocovány náměty uživatelů a v pravidelných periodách je společnost nabízí v nových verzích.“ [21]

„Vizi společnosti J.K.R. je vyvíjet a nabízet kvalitní podnikové informační systémy s ohledem na potřeby středních a větších společností. Samozřejmostí jsou špičkové a spolehlivé služby a servis. O úspěchu takového přístupu svědčí více než 1 250 spokojených zákazníků z celé země, kteří rozvíjejí svoje podnikání s podporou podnikových informačních systémů Byznys ERP.“ [21]

„Společnost J.K.R. si je plně vědoma veškerých úskalí života člověka a neopakovatelné možnosti být součástí celospolečenského dění. Z tohoto titulu společnost přispívá na zvláštní školní a výchovná zařízení a podporuje kulturní a sportovní dění ve svém regionu. Vzhledem ke svému přednímu postavení na trhu a vzhledem ke společenské potřebě výchovy nových ekonomických odborníků, společnost J.K.R. podporuje a nasazuje ekonomické informační systémy na středních a vysokých školách technického i humanitního směru.“ [21]

8.3. Organizační struktura

Personální obsazení společnosti J.K.R. s.r.o. zastřešuje generální ředitel, jemuž přímo podléhají projektový, obchodní a technický ředitel. Každý z těchto ředitelů má na starost konkrétní oddělení. Největší zodpovědnost má technický ředitel, jemuž jsou přímo zodpovědní vedoucí analytického oddělení (zabezpečuje analýzu a vývoj nových funkcionalit informačního systému), vedoucí technického oddělení (samotné programování systému) a vedoucí systémového oddělení, který zabezpečuje bezproblémový chod veškerého hardware a síťových prvků. V případě potřeby je možné též i konzultace a instalace u uživatele, který zakupuje dodávaný informační systém. Neméně důležitý je i ředitel servisu a podpory, který má na starost oddělení servisu a podpory, kde se řeší dotazy a problémy uživatelů s již zakoupeným systémem.



8.4. Produkty

8.4.1. ByznysDOS

Po svém založení v roce 1991 začala společnost J.K.R. vyvíjet svůj první softwarový produkt Byznys pro operační systém MS DOS. Zpočátku tento software nebylo možné nazývat jako plnohodnotný ERP systém, ale jednalo se spíše o účetní program, protože první modul, který byl naprogramován a samostatně prodáván byl modul Finanční účetnictví. Zaměření systému bylo spíše na malé a menší firmy, které potřebovali zefektivnit svůj chod a převést evidenci z „papírové podoby“ do elektronické. Finanční účetnictví od společnosti J.K.R. začali využívat např. i různé školy a vzdělávací instituce. V roce 1993 začal vývoj dalších důležitých modulů jako je fakturace nebo skladové hospodářství, což mělo za důsledek možnost prodeje do dalších ekonomických odvětví. Též bylo potřeba přijmout nové programátory, protože při svém založení měla firma pouze okolo 8 zaměstnanců. V dalších obdobích dochází k dalšímu rozšiřování a vývoji nových modulů přes pokladnu, bankovní operace až po evidenci majetku. Do značné míry zlomový byl rok 1995, kdy došlo k zavedení systému Byznys do společnosti Ravak, což je největší výrobce sanitární techniky (sprchové kouty, vany...) ve střední Evropě a ve své podstatě to byla první větší zatěžkávací zkouška pro software, co vše je schopen zvládnout. Tento produkt za dobu své existence dosáhl značné uživatelské oblíbenosti, kdy si ho zakoupilo více než 700 firem. Tento počet byl také dobrým odrazovým můstkem pro systém následující (viz. další kapitola), kdy většina firem vlastnících ByznysDOS přešla postupně na produkt následující – ByznysWIN

8.4.2. ByznysWin

Dalším důležitým rokem byl rok 1997, kdy začala práce na novém systému ByznysWin, který již byl programován pro operační systém MS Windows. MS DOS v tuto chvíli již začal být zastarávajícím. Základním principem bylo převzetí úspěšných a ověřených funkcí ze systému ByznysDos a jejich převedení do modernějšího a graficky příjemnějšího prostředí na platformě MS Windows. Základní tvorba nového systému trvala do roku 2000, kdy byl následně nabídnut uživatelům. V této prvotní fázi došlo k převedení již existujících modulů na platformu MS DOS na novou platformu. V dalších letech dochází k programování dalších modulů, tak aby systém byl plně konkurenceschopný. Jako první to byl modul mezd a personalistiky. V roce 2001 a 2002 dochází k implementaci systému ByznysWin ve společnosti Ravak, který nahrazuje původní ByznyDOS. V následujícím roce se vedení společnosti zamýšlelo čím odlišit jejich produkt od produktů konkurence. A následně bylo rozhodnuto o vytvoření nových, značně specifických modulů, které by bylo možné využít při obchodních jednáních a prezentacích. Jednalo se o CRM a Workflow. CRM – customer relationship management neboli řízení vztahu se zákazníky, kde byly evidovány všechny způsoby kontaktů s partnery a zákazníky a tyto kontakty byly následně vyhodnocovány. Workflow by se volně nechalo přeložit jako řízení procesů, kdy je definována struktura jednotlivých firemních procesů, jejich činností a vazeb a jsou určeni zodpovědní pracovníci za jednotlivé činnosti. V roce 2005 dochází ještě k programování části zvané OLAP neboli Business intelligence. O dva roky později dochází také k ukončení uživatelské podpory systému ByznysDOS.

8.4.3. ByznysVR

Dalším přelomovým rokem je rok 2007, kdy začal vývoj nového produktu a to systému Byznys VR (Vista ready). Nejednalo se, ale o zcela nový plnohodnotný systém, ale o jakési vylepšení již existujícího ERP softwaru ByznysWin. Vylepšení se netýkalo ani tolik funkčních vlastností, jako spíše vzhledu a komfortu uživatele. Byly například přidány ikony, kterými bylo možné spustit jednotlivé nabídky nebo menu v ribbonu (menu, které je např. v MS Office 2007 v podobě jednotlivých „záložkových karet“). Do nového systému byl přidán i jeden nový modul, kterým se měl také odlišit od předcházejícího a zvýšit jeho prodejnost. Jednalo se o projektové řízení, které umožňovalo členit různě složité projekty od stavby rodinného domu až po obchodní centrum na jednotlivé dílčí činnosti, jejich zdroje a časovou a finanční náročnost.

Základní charakteristiky ByznysVR:

- vývoj na technologii .NET společnosti Microsoft,
- podpora nových trendů (menu v ribbonu, podpora formátování, uživatelská plocha)
- optimalizace pro MS Windows Vista a MS Windows 7 (sada Gadgetů, Windows Aero),
- komunikace s MS Office systém 2007 (plastické grafy, komponenta BVR Office),
- podpora nejnovějších formátů – DOCx, XPS.

8.4.4. Byznys EVO

Ovšem ani toto nadstavbové vylepšení systému ByznysWin nemohlo zvrátit jeho postupnou zastaralost, vždyť jádro bylo vytvořeno již v roce 1997. A proto došlo k rozhodnutí vytvořit zcela nový, s novými technologiemi pracující systém. Jeho vývoj začíná v roce 2009 a má pracovní název Byznys EVO. Bude se jednat o třívrstvou aplikaci, jejíž jednotlivé části budou striktně odděleny a na sobě nezávislé. Jsou zde využívány nové technologie jako např. Microsoft Silverlight, dochází přijímání nových programátorů, jejich počet již dosahuje 30, analytiků i testerů. V současné době je hotovo jádro nového systému a předpokládané uvedení na trh pro první zákazníky je rok 2012.

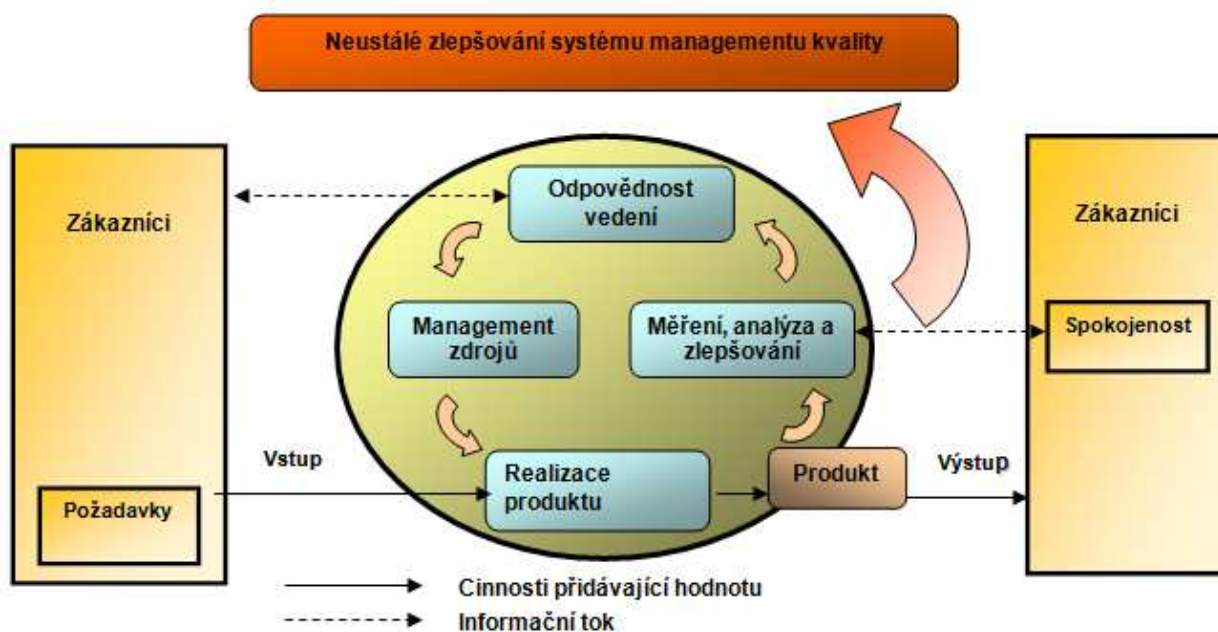
9. Procesy a metriky

Cílem této části mé diplomové práce je ukázat, jak v minulosti probíhaly či doposud probíhají klíčové procesy řízení životního cyklu produktu. V této části se již nebudeme zabývat životním cyklem jako celkem, ale zaměříme se na fázi údržby informačního systému, který společnost produkuje.

9.1. Procesní přístup

Veškeré činnosti v organizaci jsou realizovány procesním přístupem s cílem zajištění neustálého zvyšování spokojenosti zákazníků. Procesní přístup umožňuje nepřetržité řízení jednotlivých procesů a jejich vazeb a vzájemného působení. Procesní přístup zdůrazňuje důležitost pochopení požadavků zákazníka a jejich plnění, zvažování procesů z hlediska přidané hodnoty, dosahování zvýšení výkonnosti a efektivnosti procesů a neustálého zlepšování procesů.

Na následujícím obrázku je zobrazen systém procesně orientované organizace. Na začátku stojí požadavky zákazníka a na konci požadovaný produkt jako výstup. Zákazník prostřednictvím informací, které se týkají vnímání organizace, ovlivňuje neustálé zlepšování produktů a procesů:



Obrázek 11: Procesní přístup

Za dobu mého působení v J.K.R. jsem posbíral mnoho zkušeností a poznatků o tom, jak procesní přístup ve společnosti funguje a pokusím se nyní vyvodit několik závěrů.

Společnost je silně zaměřená na zákazníka. Organizace funguje pro své zákazníky a snaží se uspokojit jejich potřeby, aby dlouhodobě obstála v konkurenci ostatních společností v oboru. Organizace je velice závislá na svých zákaznících a proto se snaží rozumět jejich současným a budoucím potřebám. Snaží se plnit jejich požadavky a předvídat jejich očekávání. Jako aplikační konzultant jsem prošel mnohými školeními, jak komunikovat se zákazníky a jak efektivně uspokojovat jejich požadavky.

Vedení a řízení zaměstnanců je ve společnosti na vysoké úrovni. Osobní zkušenost mám s ředitelem servisu a podpory, který byl mým přímým nadřízeným. Vedení vytváří soulad mezi účelem a směřováním organizace jako celku. Ředitel servisu a podpory se snaží konzultanty plně zapojit při dosahování cílů organizace. Dle mého názoru se toto nejvíce projevuje obousměrnou komunikací při řešení problémů a také při sladění individuálních potřeb zaměstnanců se zájmy organizace. Jako konzultanti jsme nesli velkou zodpovědnost za vztah se zákazníkem a na „oplátku“ nebyl problém ze strany organizace vyjít vstříc například při individuálním plánu dovolené pro ty z nás, kteří studovali. Plán dovolené byl díky povaze oboru organizace pevně daný v souladu s celopodnikovou dovolenou a omezeným provozem podpory pro zákazníky během deklarovaných dvou týdnů v roce.

Na vysoké úrovni je také vysoké procento zapojování zaměstnanců na všech úrovních do různých školení a rozvoje zaměstnanců. Za dva roky mého působení jsem prodělal spoustu školení, které se hodila nejenom pro výkon práce v J.K.R., ale také pro můj další profesní život.

Procesní pojetí organizace má, dle mého názoru, ve společnosti jednu značnou výhodu. Za tu bych označil fakt, že veškeré pracovní činnosti jsou velice podrobně popsány. Byla velická výhoda na začátku mé kariéry v J.K.R., že jsem si mohl tyto procesy nastudovat a řídit se podle nich. Když to srovnám se svou současnou funkcí – IT aplikačního specialisty v CPI Group, kde vedu aplikační část IT oddělení, tak jsem si musel tyto procesy tzv. vzít s sebou a definovat je – byla to pro mě nezbytnost, abych mohl efektivně řídit své podřízené.

Současně je management ve společnosti pojat systémově – což je dle mého názoru dobře. Identifikace, porozumění a řízení vzájemně souvisejících procesů jako systému zvyšuje efektivnost a účinnost naplňování cílů organizace.

Vedení se snaží neustále zlepšovat procesy a produkty. Vždy, když se vyskytl nějaký problém při zpracování dané problematiky či procesu, tak se vedení sešlo na interní poradě a snažilo se proces zlepšit tak, aby lépe vyhovoval zákazníkům či zaměstnancům. Když při tomto procesu zlepšování bylo nutné rozhodnutí, tak se vždy dělalo na základě faktů – vždy se využilo dat a údajů z monitorování a měření procesů produktů.

Posledním poznatkem k této části, který bych zde chtěl uvést, je skutečnost, že organizace se snaží budovat a udržovat vzájemně prospěšné dodavatelské vztahy. Tím je zajištěna vývojová kontinuita poskytovaných řešení a podpory zákazníků při řešení případných problémů.

9.2. Posloupnost procesů

V organizaci je identifikován hlavní realizační proces:

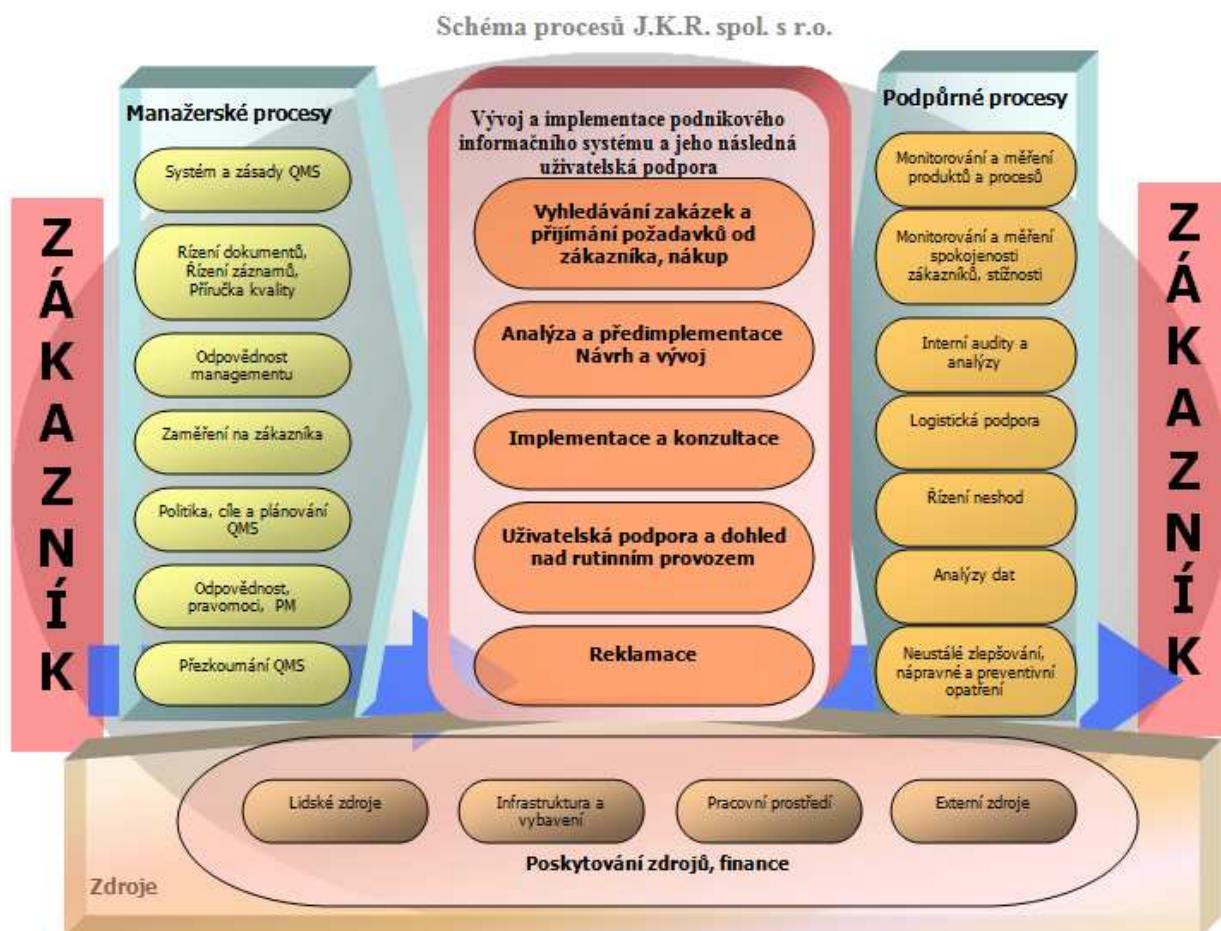
Vývoj a implementace podnikového informačního systému a jeho následná uživatelská podpora.

Hlavní realizační proces je rozčleněn do dílčích procesů, které zajišťují systémové vyřešení všech požadavků, které zákazník vznesl:

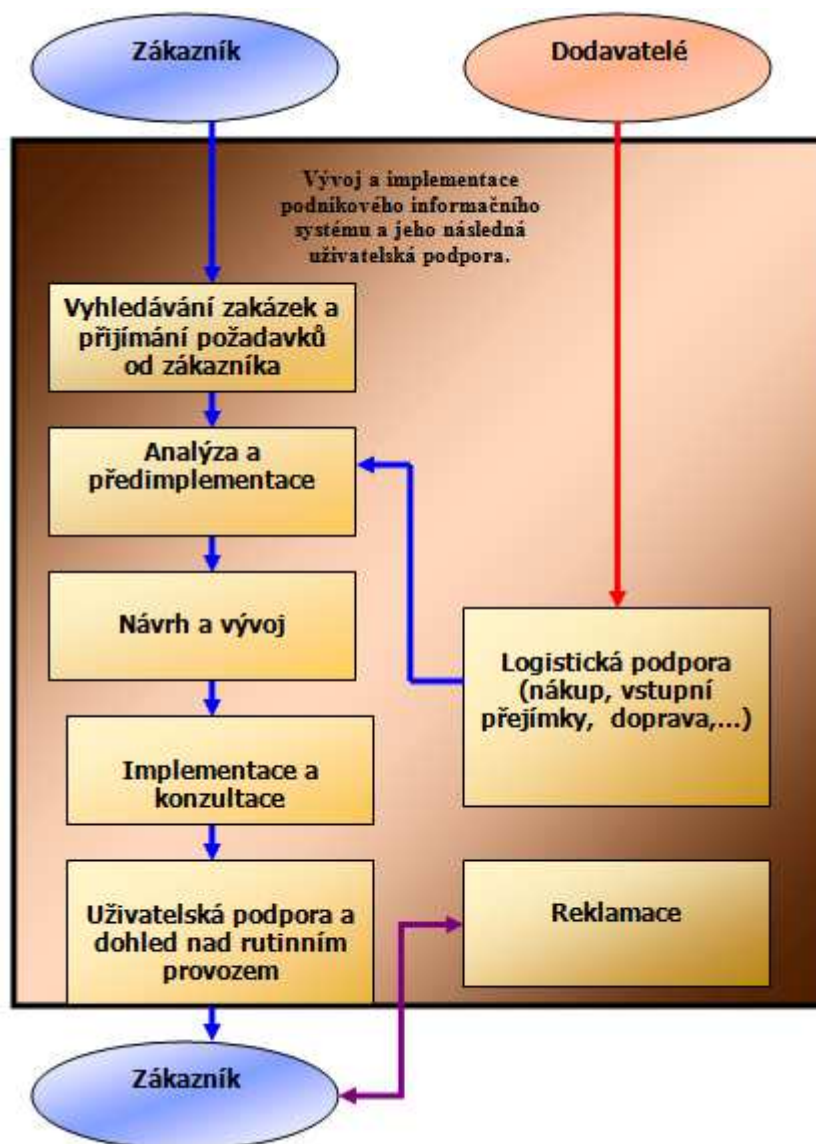
- Vyhledávání zakázek a přijímání požadavků od zákazníka
- Nákup, logistická podpora
- Analýza a předimplementace
- Návrh a vývoj
- Implementace a konzultace
- Uživatelská podpora a dohled nad rutinním provozem
- Reklamace

V oblasti vývoje, implementace podnikového informačního systému a jeho následné uživatelské podpory zadává společnost některé outsourcované činnosti externím smluvním partnerům. Jsou to zejména činnosti spojené s vývojem vybraných modulů systému (např. výroba, doprava, Excellent, eshop, ...), které společnost J.K.R. spol. s r.o. nevyvíjí samostatně. Organizace zajišťuje řízení a kontrolu těchto identifikovaných procesů včetně nastavení kontrolních mechanismů postupem určeným v dané směrnici, kterou má pro tento účel vypracovánu. Požadavky na externí zdroje jsou smluvním partnerům předem zadány a jejich plnění je řízeno a kontrolováno organizací.

Návaznost všech řídicích, podpůrných a realizačních procesů a procesů poskytování zdrojů v organizaci je znázorněna na následujícím schématu "**Matice procesů a činností**". Posloupnosti a návaznosti dílčích procesů v hlavním realizačním procesu jsou podrobně znázorněny na schématu „**Podrobné schéma hlavního realizačního procesu**“. Plánování a řízení návrhu a vývoje nových nebo vylepšených produktů probíhá právě v HRP Vývoj a implementace podnikového informačního systému a jeho následná uživatelská podpora.



Obrázek 12: Schéma procesů J.K.R. spol. s r.o.



Obrázek 13: Vývoj a implementace

9.3. Procesy týkající se zákazníka

Jak jsem se již zmínil výše, tak je společnost vysoce zaměřená na zákazníka. Díky tomuto jsem vypožadoval spoustu procesů, které jsou přímo pro-zákaznický zaměřeny a zaměstnancům jsou tyto základy „vštěpovány“.

9.3.1. Určování požadavků týkajících se produktu

Na základě předpokládaného zájmu zákazníků organizace distribuuje a uvádí na trh výrobky a služby, které splňují požadavky vznesené zákazníkem i požadavky stanovené obecně závaznými právními předpisy, zejména z pohledu daňových, finančních a účetních norem. Produkty jsou vyvíjeny, implementovány a udržovány podle nejnovějších odborných a technických znalostí. V praxi se děje to, že pokud některý zákazník vznes požadavek na systém, tak každý tento požadavek je zvažován na technické poradě společnosti a při jeho přijetí je automaticky implementován do systému. Dle mé osobní zkušenosti mohu říci, že pokud měl zákazník požadavek, který jsme nemohli v dané chvíli uspokojit, tak ze 70 % jsme tyto požadavky zapracovávali do systému.

9.3.2. Přezkoumání požadavků týkajících se produktu

Přezkoumání je prováděno před přijetím požadavku od nového zákazníka, aby se zajistilo, že vznesené požadavky je organizace schopna beze zbytku a včas splnit.

Přezkoumáním požadavků je zajištěno, že jsou podmínky smlouvy nebo objednávky přiměřeně definovány a dokumentovány, jsou vyřešeny všechny rozdíly mezi požadavky zákazníka a možnostmi organizace a požadavky smlouvy nebo objednávky je možné splnit. To zahrnuje přezkoumání smlouvy nebo objednávky včetně zpracování smluv a nabídek.

9.3.3. Komunikace se zákazníkem

Komunikace se zákazníkem probíhá přímo – osobní nebo telefonické jednání, nebo nepřímo - pomocí elektronické pošty, písemností, internetu, katalogových listů, veletrhů, výstav apod. Objednávku je možno přijmout od zákazníka v písemné podobě (dopis, fax, e-mail) nebo telefonicky. O tomto telefonátu je pořízen písemný záznam.

Komunikace se zákazníkem se převážně týká vyřizování poptávek, smluv nebo objednávek, včetně jejich změn. Při komunikaci jsou získávány informace o vnímání organizace zákazníkem včetně informací souvisejících se stížnostmi a reklamacemi.

Zvláštní skupinu tvoří VIP zákazníci, kterým je věnována speciální péče. Nebylo výjimkou, když někteří zákazníci měli tzv. VIP smlouvu, tak jim byla věnována nadstandardní péče a podpora. Dokonce nebylo výjimkou, že konzultant jezdil na osobní konzultace každý týden, tak aby co nejlépe vyhověl velice náročnému zákazníkovi.

9.4. Měření, analýza a zlepšování

Organizace plánuje a uplatňuje procesy monitorování, měření, analýz a zlepšování, které jsou potřebné pro prokázání shody výrobků, zajištění shody managementu kvality a pro neustálé zlepšování efektivnosti.

9.4.1. Monitorování a měření

Spokojenost zákazníka

Organizace jako jedno z měření výkonnosti managementu kvality monitoruje informace týkající se jejího vnímání očima zákazníka. Monitorování spokojenosti zákazníka provádí organizace u zákazníků, kteří jsou určeni obchodním ředitelem jako perspektivní.

Organizace se snaží budovat si se zákazníkem silný vztah. Jako hlavní principy, které mi při práci konzultanta byly „vštěpovány“ patří to, že zákazník je nejdůležitější osobou v organizaci. Zákazník má přednost v každé situaci a zároveň to je ten, kdo přináší do organizace peníze. Mnohokrát se mi stalo, že při komunikaci se zákazníkem došlo k různým konfliktům. Bohužel není možné, aby se každý s každým vždy dohodl, ale díky těmto principům, které je potřeba při této práci brát vysoce profesionálně, se vždy podařilo konflikty vyřešit.

Vztahy k zákazníkům jsou definovány hodnotami, které organizace vyznává a jež jsou definovány v příručce jakosti. Jedná se o partnerství jako vztah vzájemnosti a vstřícnosti, kvalitu jako výraz úcty k zákazníkům a jejich potřebám, zodpovědnost jako přístup k požadavkům zákazníků a pružnost jako reakci na potřeby zákazníků.

K identifikaci spokojenosti zákazníka jsou používány určité nástroje, které nyní shrnu. Obchodní ředitel vybere určité zákazníky a několikrát ročně jim rozešle formulář Spokojenost zákazníka v rutinním provozu, implementaci nebo spokojenost s jednotlivými moduly. Náhled těchto formulářů je přiložen v příloze. Zákazník odpovídá na jednotlivé otázky a tím vyjadřuje svoji spokojenost. Je zavedena stupnice od 1 do 5, kdy nejvyšší počet bodů může být 5. Tyto dotazníky se rozesílají emailem na adresu odpovědné osoby zákazníka. V případě, že na dotazníky není reakce, ověřuje referent spokojenost zákazníka telefonicky a doplní údaje do příslušných formulářů osobně.

Takto získané výsledky spokojenosti jsou zaznamenány do formuláře „Spokojenost zákazníka“, jehož pomocí zjišťuje organizace celkovou spokojenost zákazníků.

Interní audit

Cílem interního auditu managementu kvality je systematické a nezávislé prověření stanovených činností a procesů organizace za účelem zjištění míry jejich shody se stanovenými pravidly a zásadami, uvedenými v odpovídající dokumentaci a stanovení efektivnosti managementu kvality.

Interní audity jsou prováděny v plánovaných termínech vycvičenými interními auditory, externí firmou nebo kombinací obou způsobů. Interní audit jsem za dobu mého působení zažil třikrát a dle mého názoru je tento audit prospěšný. Prověřuje znalost interních směrnic a procesů. Pokud by někdo nebyl schopen odpovědět internímu auditorovi na otázku týkající se práce daného pracovníka, byly nastaveny nápravná opatření a systém pokut danému zaměstnanci. Osobně jsem byl jednou osloven auditorem, abych vysvětlil princip nakládání s daty uživatele.

O výsledku interního auditu zpracuje interní auditor (vedoucí prověřovacího týmu), provádějící audit, bezprostředně po jeho skončení protokol, kde uvede odkazy na všechny zjištěné neshody a důležité údaje. Vedoucí prověřované oblasti navrhne nápravná a preventivní opatření (dále též NOPO). Vedení odpovídá za kontrolu realizace a efektivnosti přijatých NOPO.

Vedení předkládá jedenkrát ročně k přezkoumání zprávu o interních auditech, o plnění NOPO zavedených interními auditry a o celkovém zhodnocení stavu managementu kvality v organizaci.

Monitorování a měření procesů

Monitorování a měření procesů probíhá formou pravidelné řídicí činnosti jednotlivých vedoucích pracovníků, kteří v rámci svých povinností a pravomocí dozírají na to, zda procesy probíhají dle stanovených postupů a zda splňují všechna stanovená kritéria a posloupnosti. Tyto posloupnosti jsou stanoveny v příslušných dokumentovaných postupech pro konkrétní procesy. Měřitelné ukazatele procesů včetně stanovených cílových hodnot jsou ve společnosti určeny následující tabulkou. Údaje v tabulce jsou orientační. Ke každému ukazateli je určeno, kdo za něj zodpovídá, jak často je prováděna analýza a kde je údaj k dispozici (sloupeček záznam). Samozřejmě nesmí chybět sloupeček se stanoveným cílem.

Měřitelný ukazatel	Odpovídá	Analýza	Záznam	Stanovený cíl
Náklady, obrat, zisk	Statutární orgán	1 x měsíčně	Finanční přehled J.K.R.	Viz roční plán
Přepočtené vlastní výkony na jednoho zaměstnance/počet zaměstnanců	Odborní ředitelé	1 x měsíčně	Finanční přehled J.K.R.	1 000 000,- Kč/rok
Počet nových instalací	Obchodní ředitel	1 x ročně	Finanční přehled J.K.R.	Nárůst o 40
Počet reklamací	Obchodní ředitel, Technický ředitel	1 x ročně	Evidence reklamací a stížností	Maximálně 2 za sledované období

Monitorování a měření produktu

Monitorování a měření produktů je prováděno za účelem zajištění důkazů, že byly splněny stanovené požadavky na produkt a že zákazníkovi poskytnutý produkt bude plnit funkci, pro kterou je určen a je při použití, ke kterému je určen, bezpečný a funkční. Způsob provádění monitorování a měření včetně stanovených hodnot je dán příslušnými postupy pro realizaci procesů. Jsou vedeny záznamy s uvedením osob, které uvolnily produkt k dalšímu zpracování.

9.4.2. Řízení neshodného produktu

Neshodný produkt je takový produkt, jehož znaky a hodnoty neodpovídají stanoveným požadavkům. Neshodný produkt nemusí být nutně nepoužitelný.

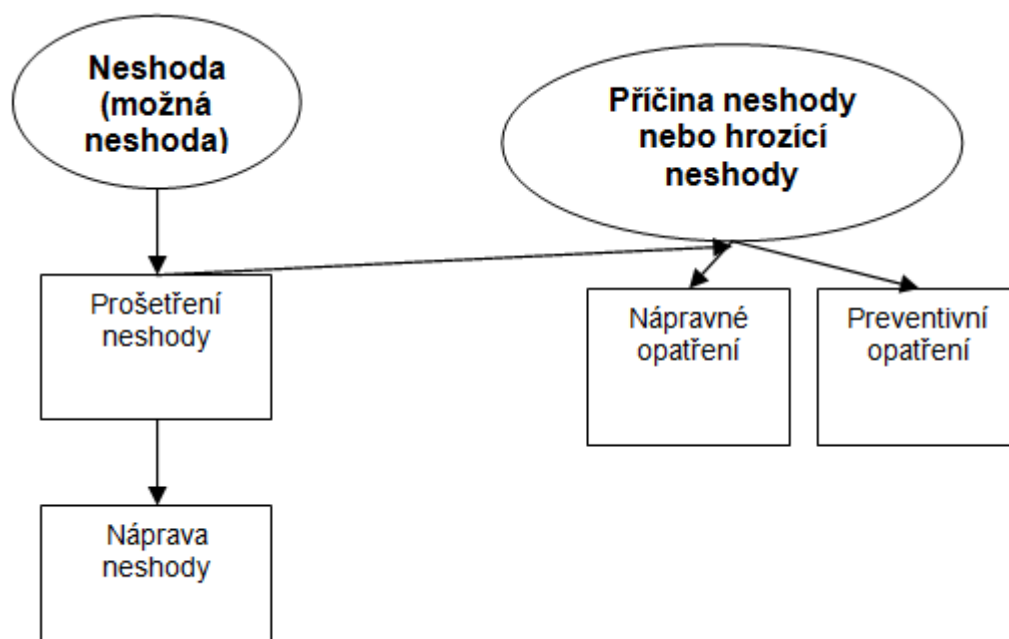
Pro zacházení s neshodným produktem, pokud je takový zjištěn v průběhu realizace zakázky, jsou stanoveny postupy, které popisují způsob označení a umístění neshodného produktu, aby se zabránilo jeho nechtěnému použití. Je stanoveno, jakým způsobem a kdo smí takový produkt vypořádat, tj. odstranit zjištěnou neshodu opravou, přepracováním, stanovením ceny s ohledem na neshodu nebo přerušením poskytování služby.

Opravený či přepracovaný produkt musí být podroben opětovné kontrole včetně pořízení potřebných záznamů.

Neshoda je stav produktu, kdy nesplňuje stanovené požadavky. Neshoda může být identifikována ve všech fázích výroby a poskytování služeb. Neshodný může být i proces (jeho průběh), měřicí nebo zkušební zařízení, které nesplňuje požadavky na něj stanovené.

Jako zdroj zjišťování neshod slouží především monitorování a měření produktů a procesů (volba monitorovacích a měřících míst, četnost uskutečňovaných měření a monitorování a kontrol, provádění interních auditů ...).

Základní princip řešení neshod popisuje následující obrázek:



Obrázek 14: Řešení neshod

Náprava neshody – opatření k odstranění vlastní neshody (zavádí se vždy při zjištění neshody).

Prošetření neshody – zjištění příčin neshody a posouzení, zda je neshoda takové povahy, že je účelné přijmout k odstranění jejích příčin nápravné opatření.

Preventivní opatření - opatření k odstranění příčin možné neshody (zavádí se na základě výsledků analýz a přezkoumání, z nichž plyne možnost vzniku neshody).

Nápravné opatření – opatření k odstranění příčin neshody (zavádí se po přezkoumání neshody a posouzení jejích příčin případně zjištění trendů nebo pravidelností).

9.4.3. Analýza dat

Aby organizace mohla neustále zlepšovat kvalitu produktů a služeb, sleduje a vyhodnocuje stanovené ukazatele a údaje, zejména v oblasti spokojenosti zákazníků, shody s požadavky na produkt, znaků a trendů procesů a produktů, kvality dodavatelů apod.

9.4.4. Neustálé zlepšování

Cílem zavedeného managementu kvality je neustálé zlepšování procesů a produktů a tím samozřejmě i spokojenosti zákazníka s výrobky a službami. To je umožněno využíváním politiky kvality a příslušných cílů, výsledků auditů, analýz údajů, opatření k nápravě, preventivních opatření a přezkoumávání managementu kvality vedením.

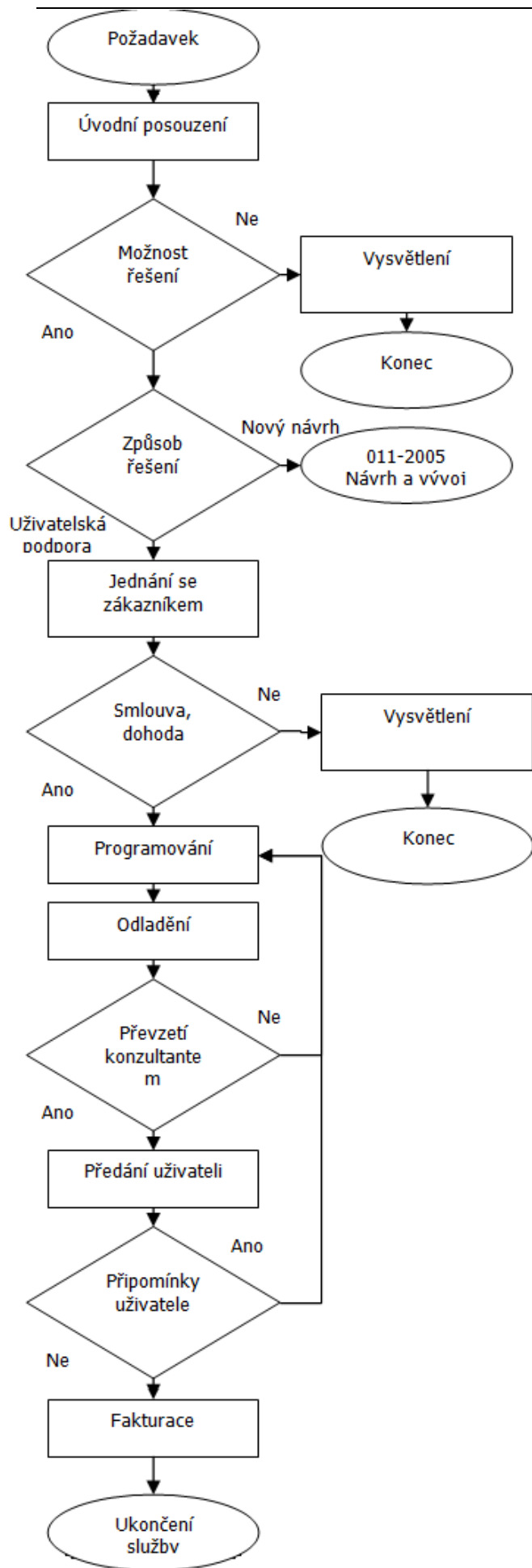
10. Vývoj a údržba v minulosti

Nyní se blíže podíváme na to, jak vypadaly procesy ve společnosti dříve.

10.1. Postup při řešení uživatelského požadavku

V této části si ukážeme, jak v praxi vypadá proces přijetí a řízení přijatého požadavku na úpravu informačního systému.

Na následujícím vývojovém diagramu je prakticky shrnuto, jak probíhají jednotlivé procesy řízení uživatelské úpravy systému. Nyní si je trochu přiblížíme.



Požadavek uživatelskou úpravu může vyplynout z Hotline podpory, osobního jednání, rady od konzultanta nebo z E-Hotline. Konzultant nejdříve provede úvodní posouzení, ve kterém provede analýzu problému. Z provedeného posouzení konzultant připraví jednotlivé možnosti řešení. V případě, že není možné řešit problém zákazníka, poskytne mu vysvětlení důvodu a proces je ukončen. V případě, že existuje řešení problému zákazníka, rozhodne konzultant, zda se bude problém řešit jako uživatelská úprava nebo jako nový návrh. Jestliže se bude problém řešit, jako uživatelská úprava, konzultant se setká se zákazníkem, kde mu předloží možnosti řešení a finanční podmínky řešení. V případě, že se konzultant dohodne na podmínkách je podepsána smlouva nebo dohoda se zákazníkem. Jestliže zákazník neakceptuje požadavky organizace - je poskytnuto vysvětlení a proces je ukončen. Po podepsání smlouvy nebo dohody je vyřešen problém naprogramováním požadavku zákazníka a je provedeno odladění produktu. Po naprogramování a odladění produktu převezme konzultant od programátora hotový produkt. V případě připomínek vrátí produkt k dopracování programátorovi. Po převzetí konzultantem je hotový produkt předán uživateli. V případě připomínek uživatele, jsou tyto připomínky předány programátorovi k dopracování. Po ukončení procesu jsou podklady předány referentovi obchodu, který vystaví zákazníkovi fakturu.

10.2.Návrh a vývoj

Nyní se dostáváme do části práce, kde bych rád popsal, jakým způsobem probíhá řízení návrhu nového požadavku na nové funkcionality a vylepšení informačního systému. Dříve (při používání vodopádového modelu) existovala pouze jedna skupina programátorů, která zabezpečovala jak vývoj nového systému, tak i opravy a údržbu stávajícího systému. Následující popis procesu vývoje tedy popisuje jak vývoj, tak údržbu.

10.2.1.Plánování návrhu a vývoje

Organizace plánuje a řídí návrh a vývoj nových nebo vylepšených produktů v HRP „Vývoj a implementace podnikového informačního systému a jeho následná uživatelská podpora“. Námět k návrhu na nový nebo vylepšený produkt (modul) může podat kdokoli (např. zaměstnanec, zákazník, externí legislativní pracovník apod.) a to dle požadavku trhu, konkrétního zákazníka, odborných poznatků, novel obecně závazných právních předpisů dotýkajících se produktu, požadavků nebo poznatků z před-implementační přípravy nebo vlastního plánu návrhu a vývoje (ze zkušeností, vylepšování v rámci jednotlivých modulů, návrhů nových modulů). Navrhovatel zasílá návrh formou E-Hotline na oddělení IKO (interní konzultační oddělení), kde je zapsán do systému IS Hotline. Ředitel servisu a podpory tyto návrhy přednese na technické poradě k posouzení a případnému schválení. Konečné stanovisko ke schválení má technický ředitel. V případě schválení se stanoví na technické poradě (termín, harmonogram). Následně se konečné stanovisko zapíše do systému IS Hotline a navrhovateli je sdělen výsledek posouzení návrhu.

10.2.2.Vstupy a výstupy návrhu a vývoje

Analytik zpracovává požadavky na produkt ve formě podkladů, které obsahují poznatky a informace odvozené z předchozích řešení problémů, zákonné požadavky a požadavky předpisů vztahujících se k produktu, které poskytne legislativní pracovník, a další požadavky včetně funkčnosti a provedení. Další řešení návrhu a vývoje je analytikem postoupeno ke zpracování programátorovi, který má za úkol je programově zpracovat. Výstupy ze zpracování musí být poskytnuty v takové formě a takovým způsobem, aby je bylo možno ověřit proti definovaným vstupům (testování a kontrola).

10.2.3.Přezkoumání a ověřování

Programové zpracování kontroluje tester spolu s konzultantem z hlediska funkčnosti, pokud námět na návrh a vývoj pochází od konzultanta nebo zákazníka. Tester vždy ověřuje testováním výstupy z programového zpracování a porovnává je se vstupními požadavky a konzultuje s legislativním pracovníkem tak, aby byly identifikovány a následně řešeny všechny problémy a byla vyhodnocena schopnost produktu plnit stanovené požadavky. Jestliže neshledá vážnější odchylku od původního návrhu nebo jiný závažný problém, který by bránil v dalším vývoji, zapíše změny vycházející z analytického zadání do manuálu a do podkladů pro návazná školení. Dále je zpracování postoupeno k dalšímu řešení.

10.2.4. Validace

Po dokončení programového řešení je hotový produkt předán pracovníkovi IKO a pracovníkovi vývojového oddělení ke konečné kontrole – Kompilační kontrola (provádí se duplicitně na odděleních IKO a Byznys týmu pro zabezpečení minimalizace výskytu chyb v produktu). Tato kontrola se provádí dle kompilačního protokolu. V případě, že kompilační kontrola nezjistí žádné nedostatky a řešení vyhovují všem požadovaným podmínkám, provede pracovník technického oddělení Interní oznámení o realizaci kompilace na obchodní oddělení a programový produkt je umístěn na distribuční místo. Tímto je určeno, že výsledný produkt splňuje zadání, vyhovuje daným požadavkům a je vhodný pro specifikované uplatnění a použití a bude zařazen do stálé nabídky organizace a může být poskytován zákazníkovi.

10.2.5. Prezentace nových funkcí

Po provedení validace dochází k internímu školení ostatních pracovníků společnosti. Na školení jsou představovány nové funkce zapracované do nově distribuovaného systému.

10.2.6. Řízení změn návrhu a vývoje

V případě závažných změn návrhu a vývoje se postupuje tak, jako by se jednalo o přijetí nového námětu návrhu a vývoje. Analytik navíc přezkoumává ohodnocení vlivu změn na již dodaný produkt.

10.3. Řízení procesu údržby

V této podkapitole se zaměřím na to, jakým způsobem se řídí a zpracovává přijatý požadavek na opravu/údržbu od zákazníka.

Prvním procesem je samotné přijetí požadavku. Pokud požadavek přišel od zákazníka, tak je informován, že požadavek byl přijat k analýze.

Následuje analýza požadavku. Přijatý požadavek je v této fázi nutno rozdělit na tři typy:

- 1) Může se jednat o požadavek, který pochází ze zákazníkovi neznalosti funkčnosti systému. Konzultant, který tento požadavek řeší, buď zná odpověď a sdělí ji zákazníkovi nebo ji nezná a poradí se s programátorem systému. Nakonec je ale zákazníkovi sděleno řešení jeho požadavku.
- 2) Jedná se o nahlášení nefunkčnosti systému, případně chyby, která uživateli znemožňuje práci se systémem. Chyby jsou klasifikovány dle závažnosti (přehled klasifikace chyb je popsán dále v této kapitole). Chyba je konzultantem popsána a podle závažnosti se přiřadí programátorovi k opravě. Tento okamžik se stává kritickým při minulém způsobu řešení problémů – vodopádovému cyklu. Konzultant musel jít za programátorem, který měl na starosti funkční celek systému (modul – např. Fakturace, Skladové hospodářství, atd..) a domluvit si s ním osobně termín opravy. Problém v této fázi byl ten, že programátoři, kteří měli na starosti nejvíce používané moduly (např. Fakturace) byli velice časově vytíženi a řešení problémů a požadavků se protahovalo. Bylo to způsobeno tím, že programátor řešil problémy postupně, jak přicházeli.
- 3) Jedná se o návrh nové funkčnosti. V tomto případě byl požadavek zaslán na technickou poradu ke schválení a ta rozhodla o jeho realizaci či nerealizaci – viz předchozí část „Návrh a vývoj“.

Již jsem v předchozím bodě 2) naznačil další část řešení a to je komunikace s programátorem. Konzultant programátorovi tedy specifikuje problém, dohaduje s ním termín a posílá mu úkol. Pokud je problém ohodnocen jako závažný, je určen do distribuční verze (více o distribuční a ostré verzi v následující části) a termín by měl být takový, aby se požadavek podařilo opravit a odladit do páteční kompilace (každý pátek byla kompilace poskytována zákazníkům). Toto znamenalo, že termín pro opravu programátorem musel být zadán vždy od pondělí do středy. Čtvrtek byl nastaven jako den pro otestování opravy a pátek byla verze distribuována zákazníkům.

Po domluvě termínu opravy s programátorem je konzultant povinen informovat zákazníka do jakého data a v jaké verzi systému bude mít opravu k dispozici.

Dle mého názoru, moment, kdy se dalo termíny měnit dle domluvy s programátorem, mělo největší vliv na spokojenost zákazníků. Častým jevem, který jsem za dobu své praxe v aplikační správě systémů vypožadoval je fakt, že pro každého uživatele je jeho problém vždy ten nejdůležitější a nejzásadnější – a už celkem nezáleží na tom, zda se jedná o zásadní chybu v chodu systému či například kosmetickou vadu na tiskové sestavě. Bohužel při výše popisovaném systému zpracovávání úkolů často docházelo k tomu, že zákazník, potažmo konzultant řešící zákazníkův problém, který si doslova „vyhádal“ (vzhledem k vysoké vytíženosti programátorů) nejbližší termín, tak ten byl řešen jako první. Bohužel docházelo i k situacím, kdy na úkor nového úkolu byl odložen ten stávající.

S tímto souvisí ještě jeden fakt, který bych chtěl nyní diskutovat. V takto nastaveném režimu docházelo k efektu, kdy programátor tzv. „dělá, co ho baví“. V praxi se tento efekt projevoval tak, že úkoly, které byly nějakým způsobem zajímavé či jednoduché, tak byly řešeny ihned či přednostně a nezajímavé a složité úkoly byly programátory tzv. „valeny“ před sebou.

Další kritickou částí tohoto procesu bylo hlídání termínu. Často se v praxi stávalo, že kvůli vytíženosti programátora mohl přijatý úkol projít. Pokud se stalo, že termín opravy byl prošlý, konzultant informoval vedoucího technického oddělení, který sjednal nápravu. Většinou byl poskytnut nový termín opravy a konzultant jej musel sdělit zákazníkovi. Tímto způsobem docházelo leckdy k negativním reakcím zákazníků – zvláště jednalo-li se o závažnou chybu, kvůli které zákazník nemohl pokračovat ve své běžné pracovní rutině. Posunutí termínu mohlo být také způsobeno faktem, že oprava neproběhla zcela korektně a při ladění tohoto zásahu se zjistilo, že problém není ještě na 100% odstraněn.

Tím se dostáváme do části testování opravy. Pokud po splnění úkolu konzultant zjistil, že problém není úplně vyřešen, zasílá úkol zpět programátorovi s vysokou prioritou tak, aby programátor v následující kompilaci problém odstranil.

Ve chvíli, kdy je problém úplně odstraněn, konzultant informuje zákazníka o odstranění problému a možnosti stažení si nové verze, kde je problém vyřešen.

10.4. Kategorizace chyb

V minulé kapitole jsem se odvolával na určité stupně ohodnocení závažnosti chyby, která se podle závažnosti předává programátorovi k řešení. Nutnost kategorizace má své opodstatnění i pro konzultanta, který je zodpovědný za to, že přijatý požadavek je správně označen. Podle tohoto označení se v procesu postupuje s různou náležitostí na rychlost řešení.

Nemá asi smysl zde popisovat veškeré podrobnosti, které určují závažnost problému. V následujícím výčtu je jen základní popis a kompletní výčet vlastností je přiložen v příloze.

- 1 – **nepodstatná chyba** - drobná chyba, která nemá vliv na zpracování.
- 2 – **lokální chyba** - chyba, která neovlivní zpracování ani prezentované výsledky, ale má pouze lokální dopad.
- 3 – **chyba zpracování** - chyba, která ovlivní zpracování nebo prezentované výsledky.

4 – **závažná chyba** - chyba, která způsobí nefunkčnost určité části systému nebo postupného zpracování.

5 – **zásadní chyba** - chyba, která znemožní zpracování.

6 – **jiné** – ostatní zaslané e-maily – nejedná se o chybu ani námět

7 – **námět** – požadovaná funkčnost, která má v BYZNYSu Win opodstatnění, ale nebude se realizovat ve známém termínu.

8 – **interní nezávažná chyba** – chyba, která byla odchycena v procesu ladění a která se nedostala k zákazníkům.

9 – **interní závažná chyba** – chyba, která byla odchycena v procesu ladění a která se nedostala k zákazníkům.

0 – **novinka** – zadání úkolů na dopracování funkčnosti, změnu stávající funkčnosti

10.5.Organizace zadání úkolu

Tato část by měla zmapovat povinnosti a standardy zasílání úkolů od konzultanta programátorovi. Dříve se k zasílání úkolů využíval poštovní klient Microsoft Outlook. Její hlavní částí by pak mělo být zmapování povinných údajů, které jsou vyžadovány ze strany technického oddělení. Tyto podklady potom zpřehlední vyhodnocování všech úkolů, zpřehlední výkaznost a umožní lepší orientaci.

10.5.1.Obecné povinnosti zadavatele úkolu

Všechny úkoly bylo třeba zasílat v níže předepsané formě jako klasická zpráva (nikoliv jako úkol) na uživatele **HotlineBW**, který je součástí intranetového adresáře mailových uzlů.

V kolonce Předmět (Subject) se museli specifikovat tyto **povinné údaje**:

Modul

Dvouznakové vyjádření modulu např. UC, FA, SK

Kód chyby

1 – 0 (viz předchozí kapitola Kategorizace chyb)

Kód verze, do které je žádána náprava

O-ostrá i vývojová verze; D- pouze vývojová (distribuční) verze (termíny překlápění distribuční verze do ostré budou upřesněny)

Stručný popis problému

Stručný popis, který ovšem **vyjadřuje jasně definici** problému (např. V opravě přijaté faktury nelze změnit kód pohybu a ne pouze oprava faktury)

Příklad kolonky Předmět:

FA2O – V pořízení zápočtu se chybně definuje seznam partnerů

Což v praxi znamená, že byl objeven problém ve fakturaci, byl označen jako větší nedostatek a je žádán opravit pro zákazníka do ostré verze. Popis jasně ukazuje na konkrétní problém, který by měl být důkladněji popsán v těle úkolu.

V těle zprávy by měly být popsány tyto skutečnosti:

Nahlášeno kým

Zákazník (případně dealer)

Verze, na které problém vznikl (datum + číslo)

Např. verze: 3.18.0138 ze dne 16.10.2009

Server, kde jsou umístěna data na kterých je problém navozen

Týká se pouze interní komunikace v rámci J.K.R. nebo dat, která jsou již v J.K.R. z dříve

Název databáze

Týká pouze interní komunikace v rámci J.K.R. nebo dat, která jsou již v J.K.R. z dříve

Požadovaný termín úkolu

Požadovaný termín, který nemusí být vyplněn. Skutečný termín se odvíjí od dohody mezi konzultantem IKO a programátorem (v závislosti na vytížení programátora a složitosti zásahu).

Popis + případně obrázek

Podrobný popis, jak bylo docíleno tohoto problému (chyby). Přesný postup a kroky, které vedly k selhání programu. Obrázek pro ještě názornější ilustraci. V případě chyby, bez obrázku bylo třeba opsat přesně chybové hlášení, které systém generoval.

10.6.Ostrá a distribuční verze

Tato část navazuje na teoretickou část, kdy bylo naznačeno, že většinou výrobce software vyvíjí systém ve dvou verzích. Jedna – ostrá- kde jsou zapracovávány hlavní změny v systému a udržuje se v ní uspořádanost jednotlivých úprav a čistota programového kódu a druhá – distribuční – kde jsou naopak prováděny rychlé zásahy programátorů za účelem odstranění naléhavých provozních problémů zákazníků. V určitém intervalu jsou pak tyto dvě verze sjednocovány do jedné, která se distribuuje méně často (konkrétně jednou za tři měsíce). V následujícím popisu se pokusím vysvětlit hlavní odlišnosti těchto dvou verzí systému.

10.6.1.Ostrá verze

Ostrá verze se realizuje v rámci jednoho měsíce – 14 dní programování a 14 dní ladění (+ oprava stejných zásahů jako do distribuční verze). Ladění se účastní i někteří programátoři. Jedná se o řízené ladění konkrétních oblastí, jmenovitě je určeno na počátku ladícího cyklu s konkrétním přiřazením laděné oblasti, kdo bude ladit jakou část systému. V polovině ladícího období jsou pozváni k ladění „svých“ chyb i externisté (dealeři). Externí ladění probíhá ve dvou skupinách, a to ve čtvrtek prvního ladícího týdne a v pondělí druhého ladícího týdne. Konkrétní sestava ladičů a organizace ladění externistů je upřesněna v organizačních postupech. Tento měsíční cyklus probíhá v rámci každého měsíce a na konci každého čtvrtletí je ostrá verze distribuována zákazníkům.

10.6.2.Distribuční verze

Jedná se o opravu závažných chyb (kategorie 3, 4 a 5 ve vazbě na kategorizaci chyb). Neznamená to, že automaticky každá 3, 4 a 5 je programována do distribuční verze, ale neměla by tam být programována chyba v kategorii nižší. O každém zákroku s konečnou platností rozhoduje vedoucí IKO spolu s vedoucím TO. Na zásah do distribuční verze je z hlavičky problému (Hotline) zasílán samostatný úkol. Kompilace probíhá v týdenní periodě – vždy v pátek (jen pokud byl nějaký zásah). V pátek proběhne zúžená kompilační kontrola (program kontroly je odvozen od klasické kompilační kontroly ostré verze, s rozsahem na jeden člověkodenní). Ladění zákroků do distribuční verze je prováděno okamžitě po opravě. Programátor též důkladně popíše, co za zákrok udělal, případně intuitivně, co se má „proladit“. Distribuční verze se umísťuje v pátek na web, kde si ji mohou zákazníci stáhnout. V týdně, kdy je kompilována ostrá verze, se distribuční verze nevytváří. Novou distribuční verzí se stává zkompileovaná ostrá verze.

10.7.Shrnutí

Výše popisovaný model údržby a vývoje byl ve společnosti zaveden přibližně od roku 2003 a z celkové doby mého působení ve společnosti, jsem v něm fungoval více než jeden rok. Model vychází z metodiky vodopád a v minulosti nebylo rozlišeno, zda se jedná o vývoj či údržbu. Existoval jeden tým programátorů, kde jednotliví programátoři byli specializováni na dané části (moduly) systému. Pro každý modul či oblast byly určeni programátoři senioři, kteří se zabývali spíše vývojem a programátoři junioři, kteří byli určeni k údržbě.

Při úvahách, jakým způsobem zhodnotit tento model s nástupnickou agilní metodikou, jsem došel k závěru, že bude nejlepší rozdělit toto shrnutí do několika oblastí a v nich definovat hlavní výhody a nevýhody. Tyto názory a zhodnocení bude sloužit k celkovému porovnání obou metodik a navržení vhodných postupů, které by společností mohli pomoci do budoucna.

10.7.1. Vytíženost programátorů

Jak jsem již naznačoval výše, tak vytíženost programátorů byla při této metodice velice vysoká. Nahrával tomu i fakt, že programátoři byli specializováni podle modulů, které spravovali. To mělo dva efekty. Prvním byl ten, že programátoři frekventovaných modulů (Fakturace, Skladové hospodářství, Finanční účetnictví) byli velice časově vytíženi a naopak programátoři „okrajových“ modulů mnohdy řešili nepodstatné problémy, jelikož v jejich oblasti se nevyskytovalo tolik oprav a vývojových požadavků. Druhým byl ten, že pokud si programátor junior nevěděl rady s realizací některého požadavku, tak musel jít pro radu k seniorovi – ten ale měl dost práce se svými problémy – vývojem. Tudíž se stávalo to, že místo toho, aby se senior věnoval vývoji, věnoval se opravám – tím se začínala kumulovat i doba strávená na jednotlivých požadavcích a docházelo k tomu, že termín opravy i vývoje se prodlužoval. Společnost toto řešila přijímáním nových programátorů, ale ti byli v pozici junior. Senior programátor byl většinou ten, který danou funkcionalitu vyvíjel od začátku systému či byl povýšen z pozice junior programátora. Přijmutí nových programátorů juniorů tedy paradoxně ještě zvýšilo vytíženost senior programátorů.

10.7.2. Vytíženost konzultantů

Dalším faktorem, který se týká vytíženosti, je taktéž vytíženost konzultačního týmu, ve kterém jsem působil. Kumulováním dosud nevyřešených požadavků na programátory mělo zásadní vliv na vytíženost konzultantů. Konzultanti přijímají požadavky od zákazníků a postupují je na programátory. Pokud se tedy přijal nový požadavek na modul, který je frekventovaný a programátor již je vytížen na dva týdny dopředu, tak se i konzultantovi začínali tzv. „hromadit“ nevyřešené požadavky uživatelů.

10.7.3. Spokojenost zákazníka

Spokojenost zákazníka s řešením jeho problému či požadavku je přímo závislá na předchozích dvou faktorech – vytížení programátorů a konzultantů. Na této vytíženosti záviselo to, kdy mohl zákazník očekávat opravu chyby či splnění jeho požadavku a také v jaké kvalitě. Kvůli časovému presu často docházelo k chybovosti jak na straně programátorů, tak na straně konzultantů, kteří kvůli náporu nových požadavků neměli dostatek času na dostatečné odladění funkčnosti, a často se stávalo, že oprava jedné chyby způsobila chybu jinou.

10.7.4. Týmovost

Konzultační tým, který primárně pracuje se zákazníky a přijímá jejich požadavky je složen z přibližně 10 konzultantů (počet se v průběhu času mírně mění), ale stálým základem je 5 zkušených konzultantů interního konzultačního oddělení a zbytek tvoří většinou noví členové týmu (jsou v týmu méně než půl roku) a konzultanti v režimu školení. Dle mého názoru, metodika, která se aktuálně používá, nemá zásadní vliv na soudržnost týmu konzultantů, který je ostatně na velmi vysoké úrovni.

11. Vývoj a údržba v současnosti

Přelomem v řízení životního cyklu a chápání procesů ve společnosti J.K.R. se stal rok 2011, kdy byla potřeba začít vyvíjet nový systém Byznys EVO. Se současným nastavením procesů a vývojových cyklů, kdy jeden tým programátorů prakticky jen udržoval stávající systémy Byznys Win a Byznys VR se tento posun dal předpokládat. Nebylo by totiž možné ve stávajících metodikách současně vyvíjet nový systém a udržovat současné. Proto od 1.1.2011 společnost přetransformovala svou vnitřní architekturu a začala vyvíjet nový systém a udržovat stávající pomocí agilní metodiky SCRUM. V této kapitole si tedy ukážeme, jak tento přechod probíhal, ukážeme si, jaké procesy vznikly a které zanikly a následně popíšeme proces údržby pomocí metodiky SCRUM v praxi.

V následujících podkapitolách popisují jak v praxi je použita metodika Scrumu pro údržbu v J.K.R. Klasická metodologie scrumu je převážně využívána pro vývojové aktivity, pro které byla také navržena. Dále popisovaný scrum je prakticky upravená metodika přesně pro vnitřní potřeby společnosti. Tento způsob řízení údržby řeší výše popisované problémy a hlavní rozdíly jsou diskutovány na konci této kapitoly.

11.1. Agilní vývoj v J.K.R.

Cílem agilních metodik je zaměření na hotový produkt a to v co nejkratších možných intervalech. Důvodem krátkých intervalů – iterací je schopnost organizace reagovat okamžitě na změny. V průběhu iterace všichni členové týmu sdílí společný cíl vývoje v iteraci, proto v tomto prostředí fungují principy sebmotivace a výkonu (všichni členové týmu jsou „na jedné lodi“ i vzhledem k tomu, že hodnocení výsledků probíhá za celý tým (nikoliv za jedince)). Metodika Scrum poskytuje mimo výše uvedené ještě další výhody:

- Pružná komunikace mezi jednotlivými profesemi.
- Priority vývoje řídí zákazník (Product Owner) nikoliv tým.
- Role v týmu přirozeně přizpůsobuje svou práci rolím navazujícím = zvýšení výkonu týmu oproti vodopádové metodice plánování vývoje.
- S postupem času dochází v týmu k vyšší zastupitelnosti – univerzálnosti jednotlivých jejích členů.

Ve společnosti JKR byl s platností od 1. 11. 2010 zahájen vývoj dle agilní metodiky „Scrum“. Vedle dodržování této metodiky se z důvodu vyšší efektivity paralelního vývoje produktu BYZNYS Evo a údržby stávajících produktů třídy BYZNYS, mění zároveň směry zaměření jednotlivých vývojářů (tým soustředěný pouze jedním směrem je 2x výkonnější nežli dvojnásobný soubor pracovníků, kteří se musí orientovat více směry, což je způsobeno zejména používáním různých nástrojů, postupů, technologií, složité organizace, režie apod.).

Aby bylo možné dodržovat pravidla agilní techniky plnohodnotně, stanovilo se, že Scrum budou dodržovat všechny vývojové týmy ve společnosti JKR. Jedná se o tyto týmy:

Aplikační týmy – tým, který má na starosti vývoj nabídek ERP produktu. V tomto týmu jsou zástupci rolí analytik, designer, aplikační programátor, GUI programátor a tester. Od 1. 11. 2010 zahájil svoji činnost první aplikační tým. Tento tým prozkoumá práci v týmu na příkladu – cíli definovaném od hlavní plánovací session. Zároveň definuje standardy (příručky) pro práci jednotlivých rolí v týmu. Na základě výsledků práce tohoto týmu došlo 1. 1. 2011 k tvorbě druhého aplikačního týmu, který byl doplněn dalšími analytiky a programátory. Vzhledem k průběžné stabilizaci stávajících produktů třídy BYZNYS existoval předpoklad, že v budoucnosti vznikne ještě jeden aplikační tým (z vývojářů, kteří v současnosti pracují v BYZNYS týmu – viz dále). Pro každou roli v aplikačním týmu funguje seniorská pozice, která má na starost údržbu standardu pro danou roli a kontrolu jeho dodržování v praxi.

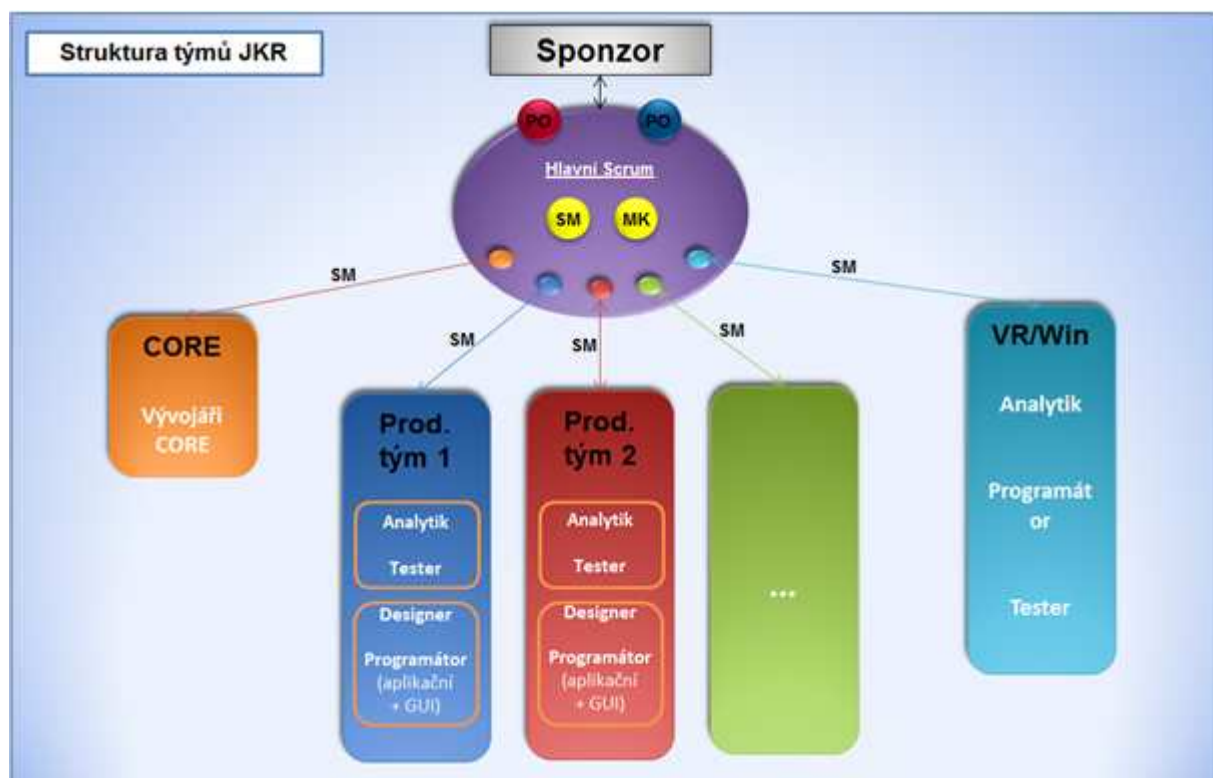
CORE tým – tým fungující zejména pro vývoj komponent do jádra, architektury a frameworku projektu Byznys EVO a dále pro podporu aplikačních týmů. Ve středu týmu je jmenován tzv. Architekt, který funguje jako technologický leader projektu. Jeho cílem je vývoj jádra a architektury v souladu s nefunkčními požadavky ERP produktu a dále metodická kontrola vývojářů aplikačních týmů, zda správně využívají poskytnuté nástroje, pravidla a vhodné části architektury.

BYZNYS tým – BYZNYS tým má na starost kompletní podporu stávajících produktů třídy BYZNYS. Jeho složení je obdobné jako složení aplikačního týmu, nicméně poplatné dříve stávající architektuře (role v BYZNYS týmu – analytik, programátor, tester). Z jeho středu jsou definovány role správců datového modelu a distribučních kompilací.

Byznys týmu je věnována celá následující kapitola, protože má zásadní vliv na výsledky a závěry této práce.

Hlavní plánovací session – řídicí tým, který vystupuje oproti výše uvedeným týmům jako zákazník (Product Owner) celého vývoje v JKR. Hlavním úkolem týmu je definice vývojových cílů každému týmu na čtyřtýdenní období, případně na základě priorit společnosti určení zaměření či mimořádné přeskupení vývojových týmů (je možné v sezónním období zaměřit např. aplikační tým na podporu BYZNYS týmu, v mimosezónním období naopak, poskytnout členy CORE týmu pro speciální úkoly v týmech aplikačních apod.). Členové hlavní plánovací session jsou dva Product Owneři (technický ředitel a ředitel servisu a podpory), vedení projektu BYZNYS Evo (manažer projektu a manažer technologií) a dále 1x za 4 týdny také Scrum masteři všech týmů.

Sponzor – Sponzorem je tým složený z vrcholového managementu společnosti (jednatelé a ředitel servisu a podpory) a vedení projektu BYZNYS Evo (manažer projektu a manažer technologií). Schůzky se sponzorem se konají pravidelně 1x za 3 měsíce. V průběhu schůzky dochází k představení dosaženého milníku, ke zhodnocení období a ke stanovení resp. upřesnění milníku následujícího.



Obrázek 15: Struktura týmů

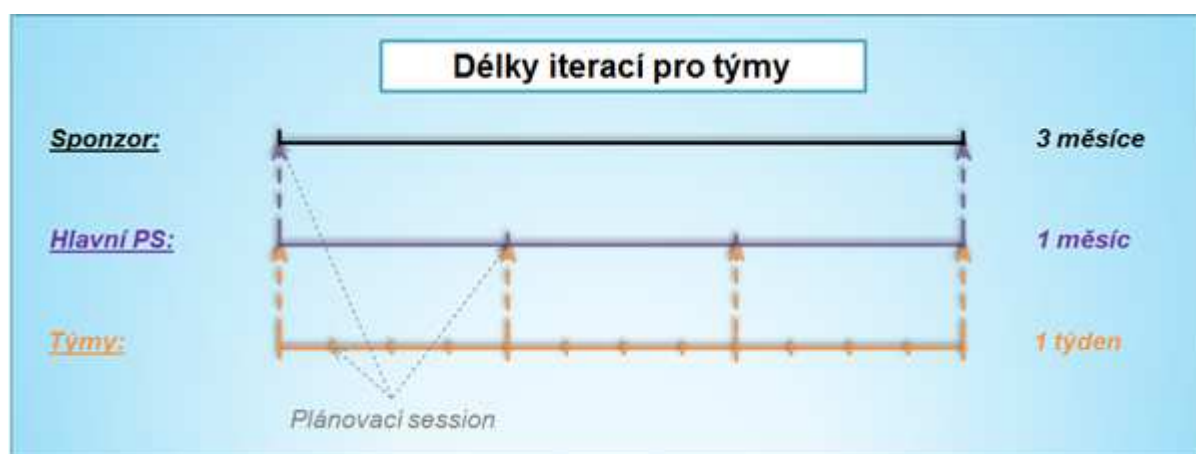
11.1.1. Délky iterací v jednotlivých týmech

Každý z výše uvedených týmů respektuje různé délky iterací, a proto také plánování cílů pro iteraci daného týmu je na jiné úrovni:

Sponzor = tříměsíční iterace – v rámci sponzorských setkání je období iterace dlouhé 3 měsíce (jedná se o tzv. 3 měsíční release). Na této schůzce se tedy definují cíle resp. milníky na příští tříměsíční období v úrovni definice částí modulů, licenčních oblastí či významných funkčních celků. Tyto cíle jsou dány schválenou strategií vývoje projektu BYZNYS Evo pro konkrétní rok, která je v souladu s významem a délkou iterace sponzorského setkání. Sponzorská schůzka nicméně posuzuje tyto cíle na základě aktuálních podmínek a v souladu s její kompetencí může náplň budoucích milníků změnit.

Plánovací porada = čtyřtýdenní iterace – délka iterace pro plánovací poradu je rovna čtyřem týdnům (tzn. všichni účastníci se sejdou mezi dvěma sponzorskými schůzkami 3x až 4x). Plánovací porada vychází z milníků sponzorského setkání a rozpadá je do menších cílů týkajících se již konkrétních nabídek ERP produktu.

Vývojové týmy = týdenní iterace – délka iterace tohoto týmu je pouhý jeden týden. Vývojovým týmům předávají cíle manažer projektu (aplikační týmy) a manažer technologií (CORE a BYZNYS tým). Povinností manažerů je rozpad cílů nabídek na konkrétní funkcionality pro určený tým a dále definice části nabídky, která ještě není hotová a zároveň nebude cílem vývojového týmu splněna (tzv. cíle k dopracování). Zároveň manažeri definují priority jednotlivých cílů mezi sebou. Na základě takto rozpadlých cílů přebírá cíl Scrum master vývojového týmu a komunikuje s manažerem projektu či technologií o rozčlenění cílů v jednotlivých iteracích vývojového týmu.



Obrázek 16: Délky iterací

11.2. Byznys tým

V této podkapitole se budu věnovat metodice práce výše zmiňovaného týmu pro maintenance stávajících systémů – tzv. Byznys tým. Výše zmiňované týmy mají své metodiky, ale ty nejsou pro účely této práce tolik podstatné, proto se jim dále již věnovat nebudu.

Přechodem na novou agilní metodiku vývoje pro systém Byznys EVO a její otestování v praxi dospěla společnost k závěru, že je nutné zavést obdobnou metodiku i u stávajícího vývoje a údržby systémů BYZNYS Win a BYZNYS VR. Zvláště pak, když se předpokládal přechod aktivních vývojářů stávajícího systému na vývoj nového produktu a tudíž bude kladen na tým vývojářů velký důraz na efektivitu, průhlednost a plánování stávajícího procesu. A na toto je agilní metodika SCRUM správnou odpovědí.

Zavedením agilní metodiky SCRUM pro BYZNYS tým došlo k zprůhlednění zadávání úkolů na vývojový a údržbový tým programátorů - BYZNYS tým (dále jen BT). V současné době vstupují do BT následující typy úkolů

- Vývoj legislativních novinek do systému BYZNYS Win/VR a jejich údržba
- Vývoj novinek od uživatelů do systému BYZNYS Win/VR a jejich údržba
- Vývoj uživatelských úprav a jejich údržba
- Vývoj interních systémů a jejich údržba (Hotline)

Z původního konceptu složení programátorského týmu se tedy přešlo na rozdělení jednotlivých funkčních skupin (vývoj, maintenance, analýza,...), což je dle mého názoru velice dobře. Zaměření jednotlivých týmů na určitou specializaci výrazně zvýšilo efektivitu zpracovávání jednotlivých úkolů. Týmy vznikly tak, že většina původních senior programátorů byla zařazena do vývojových týmů nového systému a tým pro údržbu je složen přibližně 50:50 senior a junior programátor.

Pro změnu metodiky bylo nutné změnit i stávající způsob komunikace mezi zadavateli úkolů (pracovníci oddělení IKO) a pracovníky TO (BYZNYS tým). Dříve probíhalo zadávání úkolů přímo z konzultanta na programátora přes aplikaci Hotline. Tento systém bylo navrženo zachovat, avšak změnit způsob zadávání úkolů. Konkrétní aplikaci zadávání úkolů a scrum v praxi si ukážeme v následující kapitole.

Pro novou metodiku SCRUM bylo nutné vytvořit nové role na oddělení IKO a TO.

Jedná se o tyto:

Dispečer IKO - zkušený senior pracovník, který má na starost filtrování úkolů od uživatelů systému. Řeší konfliktní situace a určuje prioritu úkolů z pohledu zákazníků. Intenzivně komunikuje se Scrum Masterem BT.

V praxi se ukázalo, že tuto roli by měl zastávat konzultant, který je nejvíce zasvěcen do procesů a má dlouhodobou praxi s fungováním společnosti. Dispečerem IKO se tedy stala má bývalá kolegyně, která dobrovolně převzala funkci nejvytíženějšího pracovníka interního konzultačního oddělení.

Scrum Master BT - zkušený senior vývojář/programátor, který má na starost plánování práce pro BT, řídí práce v týmu v rámci iterací, plánuje dlouhodobé projekty a komunikuje s dispečerem IKO.

Následně bylo nutné stanovit nové termíny pro zadávání úkolů při agilním vývoji/údržbě. Již nebylo možné si termín určovat tzv. ad-hoc, ale byl k tomu navržen následující princip:

Iterace - jedná se o jasně ohraničený časový úsek (jeden týden), do kterého se plánují vývojové práce pro BT, označují se inkrementálním číslováním po týdnech (Iterace 1, Iterace 2, ...)

Plánovací session - Každý pátek od 8.30 hodin probíhá v rámci BT plánování práce na příští Iteraci (týden). Plánování vede a řídí Scrum Master BT a účastní se jej všichni vývojáři BT, Dispečer IKO a Vedoucí TO. Plánování probíhá dle pravidel Scrumu (viz. kapitola č. 4).

Product Backlog - zásobník všech úkolů, které jsou připraveny pro následující Iterace. Z tohoto zásobníku vybírá v rámci Plánovacího sessionu Scrum Master BT společně s vývojáři úkoly, které se budou řešit v nadcházející Iteraci. Scrum Master zohledňuje v iteracích požadavky plánovací porady, která definuje úkoly vždy pro nadcházející měsíc.

Dále bylo nutné změnit i postup zadávání úkolů:

Pracovník IKO (konzultant) zavede do systému Hotline zápis zaevidováním hlavičky problému (obdobně jako v minulosti). Dále konzultant provede simulaci problému na aktuální distribuční kompilaci, případně na distribuční kompilaci, kterou má u sebe uživatel, který problém nahlásil. Pokud se problém nepodaří nasimulovat, reviduje celou záležitost s Dispečerem IKO, který rozhodne o dalším postupu. Dispečer IKO může požádat SCRUM Mastera BT o součinnost pro zjištění příčiny problému.

Pokud se jedná o prokazatelný a jasně specifikovaný problém či úkol k řešení, po dohodě s Dispečerem IKO zašle pracovník IKO úkol do Product Backlogu s konkrétním termínem navrhované řešení (v této variantě musí dojít k odsouhlasení mezi Dispečerem IKO a Scrum Masterem BT). Termín má význam spíše prioritní a ukazuje na termín, kdy se pracovník IKO bude moci věnovat otestování celé problematiky.

Většina úkolů by měla implicitně směřovat do Product Backlogu k plánování na další Iterace, aby nedocházelo ke zbytečnému zatěžování aktuální Iterace. Ta je plánována z 50 procent kapacity vývojářů (tzn. cca 7 vývojářů * 8 hodin * 4,5 dne (odečtena plánovací session) = 252 hodin - z toho 50 procent = 126 hodin na týden). Procento plánování kapacity se může měnit na základě sledování vytížení a vždy po dohodě mezi Scrum Masterem BT a Dispečerem IKO.

Do plánované náplně (50%) jsou zahrnuty úkoly z Product Backlogu a to i včetně zakázek a požadavků z plánovací porady. Náplň Iterace schvalují všichni vývojáři a podílejí se na jejím plánování a vyřešení. Cílem celého týmu je splnit všechny plánované úkoly v rámci konkrétní Iterace. Pokud dojde k přeplánování Iterace, Scrum Master BT řeší tuto situaci s Dispečerem IKO a navrhuje nový termín řešení, případně jiný akceptovatelný postup (např. odložení řešení do Product Backlogu, apod.). Pokud dojde k vyčerpání všech úkolů z aktuální Iterace, Scrum Master BT vybírá (po dohodě s Dispečerem IKO) z Product Backlogu dle priority další úkoly dle volné zbývající kapacity v Iteraci.

11.3. Shrnutí

Model údržby pomocí metodiky scrum byl zaveden od 1.1.2011. Znamenalo to nové pojetí práce konzultantů i programátorů. Metodika scrum na straně programátorů znamenala kompletní změnu jejich práce. Hlavně se to týkalo způsobu realizace přijatých úkolů a ztrátu specializace. Z počátku nastával jeden zásadní problém. V metodice scrum je jedno, kdo z programátorů programuje jaký úkol, tudíž vyvstal problém s tím, že najednou programátor, který např. nikdy neviděl programové kódy fakturace, musel najednou řešit opravu v této oblasti.

Naopak výhodou tohoto přístupu byla ta, že na každý týden bylo přesně naplánováno, co je potřeba udělat a byl na to vyhraněný potřebný čas. Tudíž se již prakticky nestává to, že by programátor nestíhal nějaký požadavek splnit. Jednak proto, že na něj má naplánovaný čas, který si sám určí a také proto, že ho od práce nemá nárok vyrušit nikdo s jiným požadavkem.

11.3.1. Vytíženost programátorů

Vytíženost programátorů se postupem času ustálila na únosné a realizovatelné hranici proveditelnosti. Zavedením metodiky scrum přestalo docházet k přetíženosti programátorů. Velký podíl, dle mého názoru, na tom má kvalitní komunikace BT Mastra a IKO dispečera, kteří průběžně plánují a diskutují nárokové požadavky od uživatelů a řídí je dle urgentnosti.

11.3.2. Vytíženost konzultantů

Se zavedením metodiky scrum do procesu zpracovávání požadavků uživatelů prakticky opadla starost konzultantů o termín řešení daného požadavku uživatele. Vždy když mi přišel nový požadavek od uživatele, tak po vyhodnocení realizace jsem požadavek předal dispečerovi IKO, který dle urgentnosti požadavku určil termín programování a realizace. Na jednu stranu konzultantům odpadla nutnost diskutování termínu s programátory a navíc nebylo nutné termín u uživatele obhajovat – jelikož požadavky byly kupodivu řešeny dříve než v minulosti.

11.3.3. Spokojenost zákazníka

Spokojenost zákazníka je v tomto případě spojená s rychlostí a kvalitou řešení jeho požadavku. Jak jsem již naznačil výše, tak zavedením metodiky scrum do údržby se jednoznačně zrychlila odezva programátorů na požadavky a tím, že problémy měli určený i přesný termín řešení a pracnost, tak bylo možné i požadavek řádně odladit, tak aby byl zákazníkovi předán v řádné kvalitě.

11.3.4. Týmovost

Konzultační tým zavedením této metodiky zpočátku trochu trpěl. Bylo to způsobeno nedůvěrou v nový systém a také zbavení se zodpovědnosti za daný termín. Postupem času ale šlo vypořádat výrazné zlepšení plnění termínů zadaných programátorům a tím pádem také vyšší spokojenost zákazníků. Pro tým konzultantů to v podstatě znamenalo více času věnování se každému problému a zvýšení kvality odváděné práce.

11.4. Scrum v Byznys týmu - praxe

V předchozí kapitole jsem vysvětlil, jak v praxi funguje Byznys tým a oddělení IKO, které zadává úkoly pro tento tým. Nyní si ukážeme, jak pro čtenářovu představu vypadají jednotlivé dříve popisované pojmy v praxi.

Tuto část jsem rozdělil do dvou podkapitol – první ukazuje jak vypadá Přehled všech úkolů z pozice pracovníka IKO a druhá část ukazuje jak daný seznam úkolů vypadá z pozice programátora Byznys týmu v rámci jedné iterace.

11.4.1. IKO – přehled úkolů

Následující obrázek ukazuje, jak vypadá seznam úkolů, které je potřeba dokončit. Tento seznam je sestaven ze všech požadavků, které musí Byznys tým zpracovat, ovšem ne v rámci jedné iterace. Z tohoto seznamu Dispečer IKO vybírá úkoly, které je potřeba dokončit v rámci aktuální iterace.

Přehled úkolů									
Předmět	Řešitel	Stav	Iterace	Zakázka	Problém	Zadavatel	Klíč Info	Složitost	Termín dokončení
PR20 - PR - Do plánovaného vytížení zdroje	bigoni	S	1		35708	rous	330196	4	01.11.2010 00:00
MZ00 Elektronické odesílání Vyúčtování da	Kocura	S	1		34379	pos	330197	19	03.11.2010 00:00
MZ00 Změny v nastavení MaP od 1.1.2011	Kocura	S	3		36103		330200	12	30.11.2010 00:00
FA00 159 - DPS k faktuře u pořízení dps k	Nekolný	S	1		35317	rous	330201	6	02.11.2010 00:00
BW00 (U2100118) zakázka SVUM-CZ, s.r.o	novotny	S	2	U2100118	36157	havrlík	330203	4	15.11.2010 00:00
CM30 - nezapsání dokumentu k obchodní	novotny	V	1		36143	havrlík	330204	0	04.11.2010 00:00
CM60 Námět na TP: Logovat zvláštní opera	Nekolný	S	1		36017	havrlík	330205	2	01.11.2010 00:00
BW00 (U2100113) zakázka MEGA TRUCK	tychy	S	2	U2100113	36012	houskova	330206	10	12.11.2010 00:00
CM60 Tabulka slovenského DPH	novotny	S	7		34097	houskova	330208	16	17.12.2010 00:00
WO8D Dopracování časového spouštění	tychy	S	1		35603	rechková	330209	8	03.11.2010 00:00
CR10 - V číselníku fází po uložení konfigur	vlasaty	S	1		36149	kropackova	330215	2	03.11.2010 00:00
BW00 (U2100111) zakázka LEKKERLAND	Ladma	S	3	U2100111	36154	tnovotny	330216	9	18.11.2010 00:00
BW00 (U2100108) zakázka KALYPSO, spo	Ladma	S	6	U2100108	36130	tnovotny	330221	8	10.12.2010 00:00
BW00 (U2100120) zakázka POLDI Hütte s	vlasaty	S	1	U2100120	36141	maskova	330222	1	19.11.2010 00:00
MZ00 Změny v nastavení MaP od 1.1.2011	Kocura	S	3		36103	bendova	330226	2	30.11.2010 00:00
CM20 provádění doklad	Recman	S	2		35828	tnovotny	330227	1	09.11.2010 00:00
MZ00 Výpočet exekuce - další příjem	Kocura	S	3		36092	bendova	330231	8	26.11.2010 00:00
MZ00 Programování - Možnost propláčení	Kocura	S	2		35066	rysova	330233	24	15.11.2010 00:00
MZ60 občas jde na 1 den na rodičovskou	Kocura	S	23		37840	rysova	346584	2	05.04.2011 00:00
BW60 v případě, že ve wms není uveden ř	Ladma	S	16		37791	rous	346587	1	16.02.2011 00:00
SK70 inventura skladu	Havrlík	V	0		37382	matlova	346634	0	15.03.2011 00:00
SK00 sjednotit filtry v hromadné 1N a 1:1		O	0		37576	rous	346654	4	30.08.2011 00:00
CR00 - námět na úpravu definice přístupn	vlasaty	S	24		37776	rous	346657	2	11.04.2011 00:00
FA2D intrastat IT Bohemia	tychy	S	17		37867	havrlík	346661	3	02.03.2011 00:00
PO3D V editaci valutového pokladního dokl	Stěpánek	S	18		37933	pilous	346689	2	01.03.2011 00:00
FA0D středisko z výdejky do faktury vydané	Liska	S	18		37679	rechková	346699	2	04.03.2011 00:00
BW00 Lekkerland - chyba v log Win ze služ	Ladma	S	17	U2100111	37940	bejckova	346717	2	22.02.2011 00:00

Popis zadání
 Klíč problému: 36154
 Naplánovaná kapacita: 0
 Podpis: Tomáš Novotný
 Zákazník: LEKKERLAND Česká republika, s.r
 Klíč odběratele: 15251
 Dealer: J.K.R., s.r.o.
 Chybná verze:
 Server: xxx
 Databáze: xxx

Poznámka
 Děláno 9, zbylých 6 přesunuto do 4. iterace. podpory Celkem 23 hodin část 1. programování 15 hodin

Přidej

Detail

Refresh

Konec

Obrázek 17: Přehled úkolů

Po kliknutí na tlačítko Detail je na následujícím obrázku zobrazen detail označeného dosud nesplněného úkolu pro programátora Byznys týmu.

Detail úkolu

Předmět: BW00 (U2100111) zakázka LEKKERLAND Česká republika, s.r.o. 1/2

Řešitel: Ladma Iterace: 3 Zakázka: U2100111 Odesílatel: tnovotny

Zadání úkolu

Klíč problému: 36154
 Naplánovaná kapacita: 0
 Podpis: Tomáš Novotný
 Zákazník: LEKKERLAND Česká republika, s.r.
 Klíč odběratele: 15251
 Dealer: J.K.R., s.r.o.
 Chybná verze:

Termin: 18.11.2010 00:00:00
 Zahájení: 01.01.1900 00:00:00
 Zadáno: 29.10.2010 08:14:19
 Složitost: 9
 Důležitost: S
 Stav: S - Splněno
 Čas: 09:00
☐ Oprava dat

Popis řešení

Oblast ladění	Příčina	Provedené zásahy

Poznámka

Děláno 9, zbylých 6 přesunuto do 4. iterace. podpory Celkem 23 hodin část 1. programování 15 hodin

Ulož **Konec**

Obrázek 18: Detail úkolu

11.4.2.TO - BTScrum

Na následujícím obrázku je možno vidět, jak vypadá výběr úkolů pro aktuální iteraci z pohledu programátora Byznys týmu. Ten je složen z výběru úkolů z předchozí podkapitoly.

Id	Obch.případ	Název	Řešitel	Stav	Stavce	Hlavní	Ka	ž	Termin	Stráven	Čas	Poznámka
2485	UC60	natažení úč. dokladu - dávky	Štěpánek	S	38	40230	3	19.07.2011	06:00	15.07.2011		
2487	FA30	Natažení faktury ISDOC	Štěpánek	S	38	40269	4	19.07.2011	04:00	15.07.2011		
2493	FA60	Vyskazuje neznámý problém	Štěpánek	S	38	40298	1	19.07.2011	01:00	18.07.2011		
2490	IN60	chyba při synchronizaci kalend	Štěpánek	S	38	40301	1	20.07.2011	01:00	18.07.2011		
2491	SK60	Z čtečky (BW scener) jim při v	Štěpánek	S	38	40305	4	20.07.2011	01:00	18.07.2011		
2474	WO3D	- Nelze použít první proces ty	tichy	S	38	40250	3	20.07.2011	04:00	13.07.2011		
2475	WO2D	- Špatné chování kontroly pr	tichy	S	38	40251	2	20.07.2011	01:00	13.07.2011		
2478	FA20	grid si konfiguraci nedrží, zjst	tichy	O	38	40256	2	20.07.2011	00:00	13.07.2011		
2483	FA60	Období DPH - po nové verzi o	tichy	D	38	39252	1	20.07.2011	00:00	15.07.2011		
2484	M200	dotaz k výkazu práce P 2 - 04	Štěpánek	S	38	40287	2	20.07.2011	02:00	15.07.2011		
2038	BIW00	(U2110061) zakázka PREFE	tichy	D	38	39620	8	20.07.2011	00:00	25.05.2011		
2248	+++	řešení datových schránek v J.K.	tichy	S	38	-1	3	20.07.2011	02:00	15.06.2011		
2290	MZ10	Oprava nepřesného názvu po	Štěpánek	S	38	39865	1	20.07.2011	01:00	20.06.2011		
2295	MZ10	Oprava názvu parametru Rež	Štěpánek	S	38	39865	1	21.07.2011	01:00	20.06.2011		
2503	SK60	chyba - slučování karet		O	0	40302	0	21.07.2011	00:00	20.07.2011		
2504	FA60	- chybný tisk faktury		O	0	40297	0	22.07.2011	00:00	20.07.2011		
2507	UC60	potíže při nastavení nového o		O	0	39914	0	22.07.2011	00:00	20.07.2011		
1987	@@@	Dovolená [VL]	Ladma	S	38	-1	36	22.07.2011	36:00	20.05.2011		
1988	@@@	Dovolená [JK]	Kocura	S	38	-1	36	22.07.2011	36:00	20.05.2011		
1989	@@@	Dovolená [TV]	vasaty	S	38	-1	36	22.07.2011	36:00	20.05.2011		
1990	@@@	Dovolená [SL]	Liska	S	38	-1	36	22.07.2011	36:00	20.05.2011		
1991	@@@	Dovolená [SN]	Nekolný	S	38	-1	36	22.07.2011	36:00	20.05.2011		
1992	@@@	Dovolená [JN]	novotny	S	38	-1	36	22.07.2011	36:00	20.05.2011		

Obrázek 19: BT Scrum

Opět po přechodu na detail daného úkolu je zobrazena informace s přesným zadáním úkolu pro programátora.

Byznys Team SCRUM

Filtr Seznam **Detail**

Předmět: FA3D Natažení faktury ISDOC Iterace: 38

ID: 2481 Zadáno: 15.07.2011 16:55:56

Hlavička: 40269

Info klíč: 372249

Zakázka:

Odeslatel: bejckova [Přehy](#)

Řešitel: Štěpánek [Ulo](#)

Termín: 19.07.2011

Složitost: 4

Stav: S - Splněno

Čas: 04:00

[Log úkolu](#)

Zadání úkolu

Klíč problému: 40269
 Naplánovaná kapacita: 0
 Podpis: Anna Bejcková
 Zákazník: SENCO Příbram spol. s r.o.
 Klíč odběratele: 16966
 Dealer:
 Chybná verze:
 Server:
 Databáze:

Úkol bude realizován v distribuční verzi.
 Schválil: Roman Vinš

Popis: Ahoj,

Prosim o opravu v natažení faktury ISDOC viz předchozí úkol. Špatně se doplní celková částka soupisu.

Problém soupisu je konkrétně v tom, že částky se mi přepočtou do Kč, prvotně mám označen přepočet Kč-Měna, ale ve skutečnosti je to bráno jak měna- Kč, tzn. fakturu takto uložím a když si j otevřu v KPF, vidím přepočet měna-kč.

Pozn.: Pokud hned při natažení změním přepočet na Měna -Kč a zpět na Kč- měna, částky se srovnají. Pokud to uložím bez této opravy, již to dodatečně neopravím.

Prosim o opravu, aby se rovnou po natažení zvolil správně přepočet Kč- měna.

Ofocené v příloze.

Díky AB.

Popis řešení

Při natažení FP přes ISDOC v měně se nesmí při přepočtu soupisky měnit KUR_REV na "R", když je Kč -> měna

Oblast ladění	Příčina	Provedené zásahy
Natažení ISDOC	mění se KUR_REV při fakturě v měně	revize [Vít Tichý] isdoc_fp v JKRPRO.VCX

Poznámka

Vazby (zakázka) Vazby (klíč probl.) Statist. za řešitele Statist. za úkol Rozsílání řádek Detail zakázky [Ulož](#) [Konec](#)

Obrázek 20: BT Scrum - detail

12. Případové studie

Dostali jsme se do části, kde si na konkrétních případových studiích ukážeme, jak probíhalo celkové zpracování jednotlivých požadavků v závislosti na použité metodice vývoje/údržby. Kapitola je rozdělena na dvě části, z nichž každá je zaměřena na odlišný pohled. První část se zabývá analýzou zdokonalovací údržby, tedy analýzou jak probíhala zdokonalovací údržba při vodopádové metodice a při metodice scrumu. Druhá část se zabývá analýzou chyb – čili opravářské údržby. Opět je k porovnání chybovost v období používání vodopádové metodiky a scrumu.

12.1. Zdokonalovací údržba

12.1.1. Vstupní data

K analýze metody vodopád a scrum jsem si vybral 5 analýz od každého typu (vodopád a scrum). Tyto analýzy byly vytvářeny pro uspokojení požadavků zákazníků společnosti a jedná se tedy o zdokonalovací údržbu. Dále v této kapitole bude představeno celkem deset případových studií – jejich krátký popis a bude následovat vyhodnocení z pohledu času zpracování a nákladů, které na zpracování daného požadavku byly spotřebovány.

K vyhodnocení potřebuji mít vstupní data – ty jsem čerpal ze zápisů technického oddělení, kde se zapisuje každá práce, která se na daném požadavku dělala. Z přehledových tabulek mě budou zajímat údaje o plánovaných a reálných termínech, počet chyb, které se museli opravovat a plánovaná a skutečná délka jednotlivých aktivit v rámci vývoje. Náhled této tabulky je níže:

E3 Vyřešit druhou variantu evidence spotřební daně															
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Plánované kapacity				Popis činnosti	Prob.	vývcs	Real	Termín	Skutečná délka	Čas zapsání	Oddělení	Délka	Pl. délka	Termín	Pl. termín
z	o	v	ýv												
b	3	37	24	Vyřešit druhou variantu evidence spotřební daně	28033	1	100	..		14.4.2009 14:51	AVO		24	..	..
v	4	33	1	Vyřešit druhou variantu evidence spotřební daně	28033	4	100	15.4.2009	0	7.4.2009 13:43	DOS + IKO		1	..	15.4.2009
c	4	33	1	Vyřešit druhou variantu evidence spotřební daně	28033	4	100	17.4.2009	0	16.4.2009 8:16	AVO		1	..	17.4.2009
c	1	35	16	Vyřešit druhou variantu evidence spotřební daně	28033	1	100	15.6.2009		14.4.2009 14:51	AVO		16	..	15.6.2009
L	4	35	1	SP_DAN při dotažení sortimentu z objednávky	28033	4	100	25.6.2009	3	18.6.2009 14:22	TO		1	..	25.6.2009
j	2	35	4	Vyřešit druhou variantu evidence spotřební daně	28033	1	100	23.7.2009	1	14.4.2009 14:51	TO	4	16	23.7.2009	30.6.2009
j	4	35	1	TA - Vyřešit druhou variantu evidence spotřební daně	28033	4	100	24.7.2009	1	21.4.2009 9:15	TO		1	24.7.2009	18.6.2009
0	ti	2	35	54 SP_DAN2 na DL	28033	3	100	7.8.2009	59	22.6.2009 8:08	TO	54	16	7.8.2009	30.6.2009
1	ti	4	37	2 SP_DAN2 - FV	28033	4	100	21.8.2009	2	19.8.2009 8:53	TO	2	1	..	21.8.2009
2	ti	4	37	4 SP_DAN2 - účtenka	28033	4	100	21.8.2009	1	19.8.2009 9:20	TO	4	1	..	21.8.2009
3	ti	4	37	2 Přenos sortimentu	28033	4	100	21.8.2009	5	19.8.2009 10:47	TO	2	1	..	21.8.2009
4	ti	4	37	5 SP_DAN2 - Intrastat	28033	4	100	24.8.2009	4	19.8.2009 10:26	TO	5	1	24.8.2009	21.8.2009
5	ti	4	37	1 SP_DAN2 - příjemka	28033	4	100	25.8.2009	1	19.8.2009 11:02	TO		1	..	25.8.2009
5	ti	4	37	1 SP_DAN2 v příjemce...	28033	4	100	31.8.2009	1	27.8.2009 8:24	TO		1	..	31.8.2009
7	ti	4	37	2 Součtová částka spotřební daně v devizové faktuře	28033	4	100	31.8.2009	4	27.8.2009 14:55	TO	2	1	..	31.8.2009
8	ti	4	37	3 Při zapném režimu cen s daní lze do intrastatu špa	28033	4	100	31.8.2009	3	27.8.2009 15:29	TO	3	1	..	31.8.2009
9	ti	4	38	2 Chyby v účtence - ceny s daní x ceny bez daně...	28033	4	100	1.9.2009	2	28.8.2009 9:14	TO	2	1	..	1.9.2009
0	ti	4	38	2 Ve skladové příjemce s přepočtem Měna --> Kč je n	28033	4	100	1.9.2009	2	28.8.2009 13:15	TO	2	1	..	1.9.2009
1	ti	4	38	1 Chyby v dodacím listu (ze skladové výdejky)...	28033	4	100	1.9.2009	1	28.8.2009 14:06	TO		1	..	1.9.2009
2	ti	4	38	1 Ještě jednou dodací list - přepočet Měna --> Kč...	28033	4	100	1.9.2009	1	28.8.2009 14:39	TO		1	..	1.9.2009
3	L	4	38	1 SP_DAN2 při dotažení sortimentu z objednávky do	28033	4	100	3.9.2009	1	1.9.2009 13:25	TO		1	..	3.9.2009
4	j	4	38	2 Ošetřit přenos vydaných faktur - položka ID1 v deta	28033	4	100	4.9.2009	1	2.9.2009 14:59	TO	2	1	..	4.9.2009

12.1.2. Metriky hodnocení

V následujících dvou kapitolách budou prezentovány případové studie pro metodu Vodopád a Scrum – vždy 5 od každé. Hodnocení případových studií jsem shrnul do jednotné tabulky pro zvýšení přehlednosti prezentovaných výsledků.

V každé tabulce je pod názvem případové studie uvedeno datum zahájení a datum ukončení. Datum zahájení vyjadřuje ukončení analýzy pro daný případ a započetí prací na vývoji daného požadavku. Datum ukončení vyjadřuje konec finálního testování zadané úpravy.

Pod krátkým popisem dané úpravy se nachází část detailní rozbor, která shrnuje kvantitativní parametry hodnocení – Skutečná délka (hod.) vyjadřuje reálný vykázaný čas strávený na daném úkolu od analýzy po dokončení testování, Plánovaná délka (hod.) vyjadřuje naplánovaný čas pro dané úkoly, Počet zadaných úkolů vyjadřuje množství úkolů, které bylo potřeba zadat, aby se dané zadání dokončilo – obecně platí, že čím více úkolů bylo zadáno, tím více programátoři chybovali při zpracování a úkoly se museli opravovat a opakovat. Posledním údajem je Odchylka od plánu, která vyjadřuje rozdíl mezi plánovanou a skutečnou dobou trvání – čím vyšší **červené** číslo, tím více se realizace protáhla nad rámec daný plánem a rozpočtem nad dané zadání. Čím vyšší **zelené** číslo, tím více se ušetřilo z plánované kapacity a programátoři se mohli věnovat jiným úkolům.

Na konci každého bloku případových studií (vodopád a scrum) je také shrnutí kvalitativních parametrů hodnocení. Vyjadřují přetížení programátorů a konzultantů, spokojenost zákazníka, důvěru zákazníka v řešení jeho problému a také tzv. týmovost, která panovala v řešitelském týmu.

12.2. Vodopád

12.2.1. Případová studie 1 – Spotřební daň 2 – kolky

Název: Spotřební daň 2 – kolky			
Datum zahájení		15.6.2009	
Datum ukončení		31.8.2009	
Popis:			
Do systému bude zapracován nový způsob práce se spotřební daní. Tento umožní pracovat se spotřební daní už na straně nákupu, kdy je potřeba zohlednit zaplacení spotřební daně v podobě kolků (např. u cigaret a alkoholu). Pořizovací cena zboží je v takovém případě navýšena o hodnotu kolku.			
Detailní rozbor:			
Skutečná délka(hod.):		141	Odchylka od plánu (hod.): 41
Plánovaná délka (hod.):		100	
Počet zadaných úkolů:		29	

V této případové studii docházelo k častému chybování programátora – 29 zadaných úkolů o tom jasně vypovídá. Těchto 29 úkolů znamenalo, že se odchylka od plánu ustálila na 41 hodinách, které mohl programátor věnovat jiným úkolům. Počet zadaných úkolů je na vysoké úrovni, jelikož s každým zadaným novým úkolem se bylo potřeba vrátit i do analytické části řešení.

12.2.2. Případová studie 2 – Komunikace s insolvenčními rejstříky

Název: Komunikace s insolvenčními rejstříky			
Datum zahájení		30.6.2009	
Datum ukončení		31.8.2009	
Popis:			
Novinka umožní uživatelům přímo ze systémů třídy BYZNYS zjistit, zda někteří jejich obchodní partneři nejsou zapsáni v Insolvenčním rejstříku. Toto zjištění bude probíhat dvěma způsoby. Uživatel se bude moci přímo připojit k Insolvenčnímu rejstříku a zjistit, zda v něm příslušný obchodní partner nefiguruje. Zjišťování bude probíhat také automaticky v předem nastaveném čase, ideálně v noci.			
Detailní rozbor:			
Skutečná délka(hod.):		166	Odchylka od plánu (hod.):
Plánovaná délka (hod.):		112	54
Počet zadaných úkolů:		20	

Tato případová studie jednoznačně ukazuje na chybovost programátora a o překročení plánu o 54 hodin. Počet zadaných úkolů je na vysoké úrovni, jelikož s každým zadaným novým úkolem se bylo potřeba vrátit i do analytické části řešení.

12.2.3. Případová studie 3 – Datové schránky – fáze I.

Název: Komunikace s Datovými schránkami - Fáze I.			
Datum zahájení		10.7.2009	
Datum ukončení		31.7.2009	
Popis:			
Do lišty systému třídy Byznys přibude nová ikona, která při stisku zavolá defaultní internetový prohlížeč s adresou přihlášení k datové schránce.			
Detailní rozbor:			
Skutečná délka(hod.):		66	Odchylka od plánu (hod.): 13
Plánovaná délka (hod.):		53	
Počet zadaných úkolů:		8	

V této případové studii byl plán překročen „pouze“ o 13 hodin, ale toto číslo se podařilo udržet na poměrně nízké úrovni kvůli menší chybovosti programátora.

12.2.4. Případová studie 4 – Oddělení Osvobození od daně a Bez daně

Název: Oddělení plnění Osvobozeno a Bez daně v DZP			
Datum zahájení		30.4.2010	
Datum ukončení		15.6.2010	
Popis:			
Účelem novinky je oddělení sazeb „Osvobozeno“ a „Bez daně“ v detailu přehledu DPH. S novinkou souvisí i oddělení těchto sazeb v tiskové sestavě DZP.			
Detailní rozbor:			
Skutečná délka(hod.):		34	Odchylka od plánu (hod.): 0
Plánovaná délka (hod.):		34	
Počet zadaných úkolů:		11	

Tato případová studie se zdá být nejlepší z mnou hodnocených. Plán byl dodržen, ale počet úkolů je i přesto na vyšší úrovni.

12.2.5. Případová studie 5 – CreditCheck vs. Insolvenční rejstřík

Název: CreditCheck vs. Insolvenční rejstřík			
Datum zahájení		15.4.2010	
Datum ukončení		31.5.2010	
Popis:			
V systémech třídy BYZNYS bude zapracováno nové vyhodnocování partnerů z hlediska projektu CreditCheck jako alternativa současnému Insolvenčnímu rejstříku. Oba typy vyhodnocení nebude možné provozovat současně, vždy bude aktivní pouze jeden z nich. V souvislosti s novinkou bude přejmenován výraz Insolvenční rejstřík v kartě partnera na Důvěryhodnost			
Detailní rozbor:			
Skutečná délka(hod.):		88	Odchylka od plánu (hod.):
Plánovaná délka (hod.):		88	0
Počet zadaných úkolů:		35	

Tato případová studie si udržela hodinovou dotaci vzhledem k nastavenému plánu, ale opět je počet úkolů na vysoké úrovni.

12.2.6. Zhodnocení

Na předchozích pěti případových studiích se dá jasně vypořádat přetížení a celková vytíženost programátorů i konzultantů. Vypovídají o tom zcela zřejmě hodnoty v předchozích pěti tabulkách. Prakticky se ukázalo, že tím jak byl programátorský tým zaneprázdněn zpracováváním více úkolů najednou a zároveň byl neustále vyrušován jinými úkoly, se podepisovalo na výrazných odchylkách při plnění úkolů. To vedlo hlavně k tomu, že úkoly se stávali nezvladatelnými a že nálada a motivace programátorů postupem času klesala.

Současně s tímto faktem klesala i spokojenost zákazníků. Bylo to hlavně dáno faktem, že jejich problémy byly většinou řešeny pod časovým tlakem, což vedlo k chybám a dodaná oprava nebyla zcela správná. V lepším případě se stávalo to, že byl neustále oddalován termín opravy, z důvodů vytíženosti programátorů. Dle mé zkušenosti, jsem jako konzultant zhruba 40% oprav musel posouvat do dalších verzí, jelikož nebyly dokončené. Tento stav se stával natolik závažným, že společnost najímala další programátory, ale to pomohlo jen dočasně a poté se jev opět opakoval.

S nespokojeností zákazníků s dodaným řešením jejich problémů je spojen i další faktor a to důvěra zákazníků v dané řešení. Zkusím popsat z mé zkušenosti jeden modelový příklad. Jako konzultant jsem měl na starosti desítky firem, které využívali naše služby. Dvě z nich byly natolik spřízněné, že se zaměstnanci vzájemně znali. Nebylo výjimkou, že jsem při řešení problému jedné z těchto firem často poslouchal uživatele, kteří si stěžovali na řešení podobného problému, který jsme řešili pro druhou společnost. Časté byly narážky typu: „No snad nám to opravíte lépe než oné druhé společnosti“. To si myslím dostatečně vypovídá o důvěře, která mezi zákazníky panovala.

12.3.Scrum

12.3.1.Případová studie 1 – Fakturace ISDOC a PDF

Název: Fakturace ISDOC a PDF jednotlivým partnerům			
Datum zahájení		21.1.2011	
Datum ukončení		28.2.2011	
Popis:			
V Číselníku obchodních partnerů bude umožněno nastavit automatické zasílání faktury ve formátu ISDOC nebo PDF. Pro partnera bude možno vždy nastavit pouze jednu z variant formátu i směru odeslání faktury.			
Detailní rozbor:			
Skutečná délka(hod.):		18	Odchylka od plánu (hod.): 3
Plánovaná délka (hod.):		15	
Počet zadaných úkolů:		5	

Vzhledem k tomu, že toto zadání bylo již řešeno pomocí scrumu, tak se nepodařilo dodržet stanovený plán o 3 hodiny. Počet zadaných úkolů je ale v naprostém pořádku.

12.3.2.Případová studie 2 – Nové formuláře pro Mzdy

Název: Nové formuláře Vyúčtování daně z příjmů fyzických osob			
Datum zahájení		10.1.2011	
Datum ukončení		28.1.2011	
Popis:			
Nové formuláře Vyúčtování daně z příjmů fyzických osob ze závislé činnosti a funkčních požitků Vyúčtování srážkové daně včetně příloh dle daňového řádu, včetně změn v elektronickém vyúčtování.			
Detailní rozbor:			
Skutečná délka(hod.):		5	Odchylka od plánu (hod.): 0
Plánovaná délka (hod.):		5	
Počet zadaných úkolů:		5	

Tato případová studie je nejlepší ze všech hodnocených. Bylo u ní splněno úplně vše podle stanoveného plánu.

12.3.3. Případová studie 3 – Období pro DPH

Název: Období pro DPH			
Datum zahájení		25.3.2011	
Datum ukončení		22.4.2011	
Popis:			
Od 1. 4. 2011 platí nový přístup k uplatnění nároku na odpočet DPH. Plátce si může uplatnit nárok na odpočet DPH až v okamžiku fyzického vlastnění daňového dokladu. Z tohoto vyplývá, že nákladově doklad bude v jiném období než DPH. Z tohoto důvodu bude nutné do modulu Fakturace tuto novou funkčnost zpracovat.			
Detailní rozbor:			
Skutečná délka(hod.):		170	Odchylka od plánu (hod.): 0
Plánovaná délka (hod.):		170	
Počet zadaných úkolů:		5	

Tato případová studie byla též řešena efektivně a v rámci daného plánu.

12.3.4. Případová studie 4 – Rozdělení období

Název: Rozdělení období na účetní a daňové pro faktury reverse charge			
Datum zahájení		3.6.2011	
Datum ukončení		17.6.2011	
Popis:			
Umožnit pořídit fakturu přijatou v režimu reverse charge s dvojím obdobím.			
Detailní rozbor:			
Skutečná délka(hod.):		12	Odchylka od plánu (hod.): -2
Plánovaná délka (hod.):		14	
Počet zadaných úkolů:		4	

V této případové studii došlo k opačnému efektu než u všech ostatních – skutečná délka byla o 2 hodiny kratší než naplánovaná.

12.3.5. Případová studie 5 – Dobropisy ke skontu

Název: Dobropisy ke skontu			
Datum zahájení		3.6.2011	
Datum ukončení		17.6.2011	
Popis:			
Novela zákona o dani z přidané hodnoty účinné od 1. 4. 2011 ukládá povinnost provést opravu základu daně a výše daně z tzv. skont, slev či bonusů. Plátce je povinen vystavit opravný daňový doklad – dobropis a řádně ho zaúčtovat. Nově tedy bude možné vystavit dobropis ke skontu.			
Detailní rozbor:			
Skutečná délka(hod.):		53	Odchylka od plánu (hod.): 6
Plánovaná délka (hod.):		47	
Počet zadaných úkolů:		8	

V této případové studii došlo k překročení plánu o 6 hodin. Jedná se o nejvýraznější odchylku v rámci případových studií řešených metodikou scrum.

12.3.6. Zhodnocení

Hodnoty v tabulkách předchozích případových studií, které byly řešeny metodikou scrumu, jsou přímo úměrná i kvalitativním faktorům hodnocení této metodiky.

Přetížení týmu prakticky vymizelo hlavně tím, že celý tým programátorů byl rozdělen na funkční týmy a každý řeší pouze svou svěřenou oblast (dále se budeme bavit pouze o Byznys týmu). Rozdělením se eliminoval stav, kdy každý programátor řešil jak vývojové úkoly tak opravárenské. Vybudoval se tým, který byl určen pouze pro opravy a v něm již nebyla použita specializace každého programátora na danou oblast, ale každý programátor řešil vše. Díky denním poradám, kdy si každý programátor řekl, kolik na daný úkol potřebuje času, bylo s tímto časem počítáno a vymizel stav, kdy programátoři nestíhali danou problematiku řešit. Zároveň nemohli být vyrušováni jinými úkoly (dané problémy pro iteraci byly pevně dány). To přispělo k faktu, že úkoly začali být plněny téměř bez chyb a v daném termínu. Výrazně se to odrazilo ve spokojenosti a důvěře zákazníků, jelikož se již nestávalo, že daný pevný termín splnění požadavku nebyl dodržen a požadavky byly plněny bez chyb.

12.3.7.Celkové vyhodnocení případových studií

Po zhodnocení předchozích deseti případových studií mohu konstatovat následující vypořizovaná fakta:

- Při použití metodiky scrum se výrazně snížil počet zadávaných úkolů. Tento fakt je ovšem nutné podložit faktem, že při použití metody scrum není nutné tolik úkolů zadávat. Při chybě v metodě Vodopád se zadával úkol na analytika a další úkol byl zadán opět programátorovi. Při použití metody Scrum je tento proces výrazně zkrácen, jelikož na úkolu pracuje celý tým a není nutné úkol zadávat vícekrát.
- Výběr metodiky neměl vliv na dobu trvání jednotlivých úkolů v rámci každé případové studie – lze tedy potvrdit tvrzení z teoretické části, že vývojová metoda neurychlí vývoj.
- Výrazně se snížila chybovost při použití metody Scrum. Tento fakt je ovšem nutné podložit skutečností, že velmi záleží na programátorovi, který daný úkol plní. Z vlastní zkušenosti mohu potvrdit to, že pokud úkol v metodě vodopád plnil programátor senior daného modulu, tak chybovost byla nižší než u programátora juniora.
- Přechodem na metodiku scrum se začali plnit termíny požadavků zákazníků včas a tím vzrostla spokojenost zákazníků a celková důvěra zákazníka ve společnost, která jim dodává informační systém.

12.4.Oprava chyb

12.4.1.Vstupní data

Pro analýzu opravny chyb jsem shromáždil dva soubory hlaviček řešených problémů za srovnatelná období roků 2010 a 2011. Konkrétně se jedná o rozmezí mezi 1.3. – 30.6. 2010 a 2011. Hlavička problému je označení pro jakýsi rámec veškeré dokumentace, komunikace, úkolů a dalších záznamů vztahující se k jednomu konkrétnímu řešenému problému, který obsahuje informaci o vzniku problému a datem jeho vyřešení – toto datum je do systému zadáno až po potvrzení funkčnosti uživatelem. Tyto dva soubory dat jsou vybrána z období od zavedení nové ostré verze do distribuce (1.3.) a celého jejího „životu“ až do nástupu další verze (30.6.). Data v roce 2010 zahrnují všechny hlavičky, které byly za fungování metodiky vodopád a data v roce 2011 analogicky zahrnují hlavičky při používání metodice scrum.

12.4.2.Metriky hodnocení

Ze vstupních dat se dají agregovat různé informace, které mi pomohou určit, která metodika je účinnější při opravě chyb. Z dat, která mám k dispozici se dají agregovat následující informace:

- Počet chyb úrovně 1-5 (viz kategorizace chyb – záměrně jsem vyřadil ostatní označení hlaviček 0 a 6 – 9, jelikož tyto chyby se nedostaly k zákazníkům)
- Průměrná doba řešení problému (tu získám rozdílem doby dokončení úkolu a dobou jejího vzniku)
- Medián doby řešení problému (získám ho obdobně jako průměrnou dobu řešení problému, ale medián má větší vypovídající hodnotu)

12.4.3. Výsledky metody vodopád

Vodopád (03-06. 2010)	
Kategorie chyby	Počet chyb
1	52
2	139
3	113
4	18
5	4
Celkem	326
Průměrná doba vyřešení	24 dnů
Medián doby vyřešení	10 dní

12.4.4. Výsledky metody scrum

Scrum (03-06. 2011)	
Kategorie chyby	Počet chyb
1	51
2	111
3	91
4	15
5	1
Celkem	269
Průměrná doba vyřešení	15 dní
Medián doby vyřešení	10 dnů

12.4.5. Vyhodnocení opravy chyb

Jak je dobře vidět v tabulkách výše, tak lépe za srovnatelné období obstála údržba chyb za použití metody strum a to jak počtem chyb, které se za dané období vyskytly, ale i rychlostí jejich opravy. Konkrétně se zlepšení projevilo hlavně ve zkrácení průměrné doby o 9 dnů. Medián sice zůstal stejný, ale ze zdrojových dat je vidno, že v období Scrumu se již nevyskytují chyby, které by byly řešeny déle než 1 měsíc, naopak u Vodopádu jich bylo hodně, které tuto dobu i výrazně převyšovali. Chybovost použitím scrumu klesla o 18%.

Celková spokojenost zákazníků se zvýšila používáním metodiky scrum a reorganizací programátorského týmu.

13. Závěr

Procesní modely vývoje software a životní cyklus jako takový se používají již několik desítek let a dle mého názoru je to jediné dobře. Za dobu svého používání se jich vyvinulo opravdu velké množství, přitom jen několik málo se koncepčně ujalo a některé z prvních se dokonce používají dodnes.

Má práce byla zaměřena primárně na zástupce dvou velkých skupin těchto modelů a to metodu Vodopád a metodu SCRUM. V teoretické části jsem představil i ostatní zástupce těchto metod a v praktické části jsem ukázal, jak v reálném fungování tyto modely pracují.

Společnost J.K.R., s.r.o. používala metodu Vodopádu od svého založení do konce roku 2010. Využívala tento model pro vývoj svých klíčových aplikací, které nabízí svým zákazníkům dodnes. Tato metoda byla velice silným nástrojem pro vývoj v dobách, kdy se společnost propracovávala na vrchol v dané oblasti a kdy ještě nebyl tak silný tlak na vývoj dalších aplikací. Metodu Vodopád společnost, dá se říci, vyčerpala na kraj jejich limitů ve chvíli, kdy začala cítit potřebu opět začít vyvíjet nový software pro své zákazníky. Po několikaletém období, kdy pouze udržovala dva stěžení systémy – Byznys Win a Byznys VR, byla metoda dostatečně silným nástrojem pro správný chod procesů s tím spojených.

Technický pokrok ovšem postupuje dopředu mílovými kroky a společnost se rozhodla nezůstat pozadu a začala v roce 2010 vyvíjet nový produkt – Byznys EVO. Bylo potřeba provést zásadní změny v organizační struktuře společnosti (především v týmu programátorů) a také přejít na nový model vývoje, který by svou agilitou a pružností zvládl řídit nové požadavky společnosti na vývoj a údržbu stávajících systémů zároveň.

Model SCRUM je převážně určen pro vývojové potřeby softwarových společností. Při nasazení této metodiky ve společnosti se brzy zjistilo, že by kooperace dvou odlišných modelů nefungovala a proto se metodika SCRUM se značnými úpravami zhostila i stávající údržby systémů, které nyní společnost nabízí.

V mé práci jsem se silně zaměřil právě na tento fakt a chtěl jsem ukázat, jak se zlepšily dříve používané zastaralé procesy a metody přechodem na metodiku SCRUM. Samotný vývoj nového produktu jsem nechal trochu stranou a zaměřil jsem se na údržbu z jednoho hlavního důvodu. Tím důvodem je fakt, že jsem se aktivně procesu údržby zúčastnil jako zaměstnanec a mohl jsem sledovat změny, které zde probíhaly. V následujících odstavcích se pokusím shrnout názory, ke kterým jsem v práci došel.

Fungování společnosti jako celku začínalo s příchodem vývoje nového produktu mírně „pokulhávat“. Bylo to dáno hlavně faktem, že čím byly vyšší nároky na vývoj nového produktu, tím menší byl čas programátorů na zpracování požadavků uživatelů stávajících systémů. Bylo to hlavně dáno vysokou vytížeností programátorského týmu, důsledkem byla i velká vytíženost konzultantů a následkem toho klesala spokojenost a důvěra zákazníků v nabízené služby. Z toho lze usuzovat, že metoda Vodopád se stává nevhodnou ve chvíli, kdy je potřeba dělat více činností současně (vývoj a údržba).

S příchodem agilní metody SCRUM se díky dodržování termínů a vhodnému rozložení úkolů do iterací začala vracet důvěra a spokojenost zákazníků, která do té doby klesala. Dnes se dá říci, že společnost dokáže efektivněji a bezchybně plnit požadavky zákazníků a to se odráží na společnosti jak finančně, tak i prestiží, kterou svým zákazníkům nabízí.

Dle mého názoru metodika SCRUM přináší společnosti, která jej využívá lepší manévrovatelnost při řešení požadavků zákazníků a to je jediné dobře. Pokud by společnost na tento model nepřešla, tak by v této chvíli pravděpodobně nedokázala splnit termín dokončení nového systému a to by mohlo vyústit jak ve finanční ztráty, tak ve ztráty současných zákazníků, kteří by byly odlákáni konkurencí, která dokáže vyhovět jejich potřebám lépe a včas.

Zdroje

- [1] Articlealley [online]. 2009 [cit. 2011-12-05]. Incremental programming. Dostupné z WWW: <<http://johndirk.articlealley.com/incremental-programming-8557>>.
- [2] BASL, J.; BLAŽEK, R. Podnikové informační systémy. Podnik v informační společnosti.. Praha : Grada Publishing, 2008. 198 s.
- [3] ELLIOTT, Geoffrey. Business Information Technology. USA : Addison Wesley, 6 May 2004. 520 s.
- [4] Erpy.cz [online]. [18] <http://www.erpy.cz/2010/05/alfa-samec-erp/>, 17/05/2010 [cit. 2011-12-05]. Alfamec ERP. Dostupné z WWW: <<http://www.erpy.cz/2010/05/alfa-samec-erp/>>.
- [5] HAJDIN, Tomáš. Agilní metodiky vývoje software. Brno, květen 2005. 77 s. Diplomová práce. Masarykova univerzita v Brně.
- [6] Just Another Blog by JohnNy [online]. 2008 [cit. 2011-12-05]. Klasické a agilní metodiky vývoje software. Dostupné z WWW: <<http://johnny.bloguje.cz/740763-klasicke-a-agilni-metodiky-vyvoje-software.php>>.
- [7] KAJZAR, Dušan; POLÁŠEK, Ivan. Projektování informačních systémů I : strukturovaný a objektový přístup. Opava : Slezská univerzita v Opavě, prosinec 2003. 218 s.
- [8] KENDALL, K.E. System analysis and design. Philadelphia : Pentice Hall, 1991. 343 s.
- [9] Král, J.: Informační systémy, (Information systems, in Czech) Science, Veletiny, 1998, 356 pp
- [10] KUČEROVÁ, Helena. Projektování informačních systémů [online]. Praha 4, Pacovská 350/4 : Vyšší odborná škola informačních služeb, 2007. 115 s. Sylaby ke kurzu. Vyšší odborná škola informačních služeb. Dostupné z WWW: <http://web.sks.cz/users/ku/DOKUMENTY/pri_syl.pdf>.
- [11] MERUŇKA, Vojtěch. Metody tvorby informačních systémů [online]. Plzeň : [s.n.], 2006 [cit. 2011-12-05]. Dostupné z WWW: <http://etext.czu.cz/img/skripta/64/pef_226-1.pdf>.
- [12] Metodologie vývoje softwaru. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-12-05]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Metodologie_vyvoje_softwaru>.
- [13] Polák, J. Merunka, V. Carda, A. Umění systémového návrhu. Grada, Praha 2003, ISBN: 80-247-0424-02
- [14] Procesní standard pro vývoj a správu software. Česká republika : ISO, 2008. 138 s. Dostupné z WWW: <[15] <http://www.pdqm.cz/Standards/ISO-12207.html>>.
- [15] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 17-19
- [16] SODOMKA, Petr. Informační systémy v podnikové praxi. Praha : Computer Press, a.s., 2006. 352 s. ISBN :80-251-1200-4.

- [17] SOMMERVILLE, Ian. Software Engineering. 6th Edition. England : Pearson Education Limited, 2001. 693 s. ISBN 020139815X.
- [18] Stručná historie systémů ERP. Hospodářské noviny [online]. 26. 4. 2006, 1, [cit. 2011-12-05]. Dostupný z WWW: <<http://hn.ihned.cz/c1-18324610-strucna-historie-systemu-erp>>.
- [19] Wideman, R. Max: The Role of the Project Life Cycle (Life Span) in Project Management, <http://www.maxwideman.com/papers/plc-models/intro.htm>
- [20] Wikipedie [online]. 23. 11. 2011 [cit. 2011-12-05]. Životní cyklus informačního systému. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Životní_cyklus_informačního_systému>.
- [21] O společnosti [online]. 2011 [cit. 2011-12-05]. Web společnosti J.K.R. Dostupné z WWW: <<http://www.jkr.cz/o-spolecnosti/profil-spolecnosti>>.

Příloha č. 1

Kategorizace chyb

Příloha č. 2

Spokojenost zákazníka v rutině

Příloha č. 3

Spokojenost zákazníka v implementaci

Příloha č. 4

Spokojenost zákazníka - moduly