TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Vývoj a implementace softwaru pro zákazníky společnosti Actis, s. r. o.

Bakalářská práce

Studijní program: B6209 – Systémové inženýrství a informatika
Studijní obor: 6209R021 – Manažerská informatika

Autor práce: **Barbora Votrubcová**
Vedoucí práce: Mgr. Tomáš Žižka

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Barbora Votrubcová**
Osobní číslo:	**E13000052**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Manažerská informatika**
Název tématu:	**Vývoj a implementace softwaru pro zákazníky společnosti Actis, s. r. o.**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

1. Metody vývoje a implementace softwaru, licenční modely
2. Charakteristika společnosti Actis, s. r. o. z pohledu nabízených softwarových služeb a řešení
3. Analýza příležitostí a potřeb zákazníků Actis, s. r. o.
4. Proces implementace, adaptace a podpory programového vybavení
5. Zhodnocení použité metody vývoje

Rozsah grafických prací:

Rozsah pracovní zprávy: **30 normostran**

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

Seznam odborné literatury:

WIEGERS, Karl Eugene. Požadavky na software. Brno: Computer Press, 2008. ISBN 978-80-251-1877-1.
MARTIN, Robert Cecil. Agile software development: principles, patterns, and practices. International ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2012. ISBN 01-327-6058-4.
BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. Praha: Grada Publishing, 2012. ISBN 978-80-247-4153-6.
BASL, Josef a Roman BLAŽÍČEK. Podnikové informační systémy: podnik v informační společnosti. 3., aktualiz. a dopl. vyd. Praha: Grada Publishing, 2012. ISBN 978-80-247-4307-3.
Elektronická databáze článků ProQuest (knihovna.tul.cz).

Vedoucí bakalářské práce: **Mgr. Tomáš Žižka**
Katedra informatiky

Konzultant bakalářské práce: **Ing. Václav Ševčík**
Actis, s. r. o.

Datum zadání bakalářské práce: **31. října 2015**

Termín odevzdání bakalářské práce: **31. května 2017**

Technická univerzita v Liberci
Ekonomická fakulta

L.S.

doc. Ing. Miroslav Žižka, Ph.D.
děkan

doc. Ing. Jan Skrbek, Dr.
vedoucí katedry

V Liberci dne 31. října 2015

Prohlášení

Byla jsem seznámena s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědoma povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracovala samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Ráda bych tímto způsobem poděkovala vedoucímu práce Mgr. Tomáši Žižkovi a Ing. Václavovi Ševčíkovi, jednateli společnosti Actis, s.r.o. za podporu a vedení v průběhu zpracování této bakalářské práce. Dále pak celému kolektivu ve společnosti Actis, s.r.o. za přijetí a pomoc při získávání nových zkušeností.

Anotace

Bakalářská práce se zaměřuje na vývoj a následnou implementaci softwaru pro zákazníky společnosti Actis, s.r.o. Jejím cílem bylo zjistit a popsat, jakým způsobem by šel zlepšit celý proces příštího vývoje a implementace u dalších zákazníků, na základě pozorování u stávajících klientů. Teoretická část se proto zaměřuje na možné metodiky vývoje, cyklus, jakým by každý software měl projít. V praktické části je poté upřesněno, na čem nemá smysl lpět a co je naopak důležité pro urychlení, či celkově hladký průběh napříč celým životem softwaru.

Klíčová slova

Adaptace, implementace, metodika, software, zákazník

Annotation

This bachelor thesis focuses on the development and subsequent implementation of software for customers of the company Actis, s.r.o. Its aim was to identify and describe how they would go to improve the whole process of future development and implementation of other customers, based on observations of existing clients. The theoretical part is focusing on possible methodology of development. In the practical part is clarified, what is almost purposeless to solve and what is important to accelerate of the development, and overall smooth running throughout the process life of the software.

Key Words

Adaptation, customer, implementation, methodics, software

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

IT	Informační technologie
PDF	Portable Document Format
SW	Software
TUL	Technická univerzita v Liberci
VPN	Virtual Private Network
WF	Work Flow
XLS	Přípona sešitu Microsoft Excel

Úvod

Vývoj softwaru na míru je nesmírně zajímavá činnost, ale také se za touto aktivitou může skrývat mnoho problémů a překvapení. Každý zákazník je úplně jiný, ať už dokáže specifikovat své požadavky či nikoliv, má různé důvody, proč se rozhodl pro software na míru. Vyvíjení softwaru na zakázku ukazuje krásy i neduhy různých typů firem a vystavuje obě strany různým výzvám, které všechny zúčastněné posouvají dál.

Cílem této práce je nalézt a popsat, jakým způsobem lze zlepšit celý proces vývoje a implementace softwaru pro zákazníky společnosti Actis, s.r.o na základě pozorování u stávajících zákazníků. Nalezení vhodného způsobu či tipů pomůže uspořit jak časové, tak finanční náklady společnosti, ale i zákazníka.

Práce je členěna do dvou hlavních částí, do teoretické a praktické pasáže. Teoretická část popisuje náležitosti vývoje, celý životní cyklus softwaru, metodiky vývoje softwaru a jeho licenční modely.

Praktická část se poté zabývá produkty a zákazníky společnosti Actis, s.r.o. Popisuje, jakým způsobem probíhá reálný vývoj a implementace k zákazníkům. Ukazuje, čemu je lépe se vyhnout při vývoji i implementaci a které prvky a postupy by bylo vhodné využít i u dalších potenciálních zákazníků.

Závěr práce nabízí některá doporučení, která mohou pomoci urychlit proces implementace, ušetřit finanční prostředky (ať již za samotný vývoj, či cestovné) a snad i zvýšit spokojenost zákazníka.

1 Životní cyklus softwaru

Životní cyklus produktu je časový interval od stanovení koncepce produktu po jeho vypořádání (ČSN EN 60300-3, 2009).

Životní cyklus softwaru lze obecně rozdělit do několika fází. Po vyřešení základních otázek plánování, návrhu a řízení následují tyto fáze projektu podle Šmída (Šmíd, dat. vyd. neuveden):

- Předběžná analýza neboli specifikace cílů
- Analýza systému neboli specifikace požadavků
- Projektová studie neboli návrh
- Implementace
- Testování
- Zavádění systému
- Zkušební provoz
- Provoz a údržba

1.1 Předběžná analýza – specifikace cílů

Cílem předběžné analýzy je zjistit základní požadavky a cíle zákazníka, které pomohou nastínit celkový vzhled, funkce, vstupy a výstupy softwaru. Jedná se o hrubé nastínění požadavků tak, aby bylo možné odhadnout dobu realizace a náklady spojené s vývojem a implementací softwaru. Podrobnějšími souvislostmi, funkcemi a cíli se věnuje analýza systému, která je popsána v další kapitole.

1.2 Analýza systému – specifikace požadavků

Neméně důležitou fází je analýza systému, která úzce navazuje na přechozí část. V této fázi se dopodrobna rozebírají všechny situace, které by mohly po nasazení nastat. Podcenění této fáze, či zapomenutí mnohých detailů, může zapříčinit chybovost systému, či dokonce jeho

nefunkčnost. Přehlédnuté chyby se později odstraňují jen velmi těžko s vynaložením velkého úsilí a financí.

1.3 Projektová studie – návrh

Projektová studie obsahuje informace jako předpokládaný časový harmonogram, kalkulaci ceny, podmínky spolupráce, záručního i pozáručního servisu. Dále by v ní neměly chybět informace o předávání celého softwaru a informace o jeho dodavateli. Vše by mělo být srozumitelné a pochopitelné pro osoby, které budou o schválení, případně neschválení projektu rozhodovat. Proto je důležité i tuto fázi nepodcenit, protože každá nejasnost a nesrozumitelnost staví bariéry. Tyto bariéry poté mohou dopomoci k zavržení celého systému.

1.4 Implementace

Tato část je především o programování softwaru. Programátoři využívají všechny dokumenty, které při vzájemné spolupráci klienta a dodavatele již vznikly. Těch se musí programátoři držet tak, aby byly zachovány všechny vstupy a výstupy popsané v dokumentech, aby byly splněny dohodnuté podmínky. V této fázi se také připraví testovací data, případně napíšou testy. Dílčí testování by mělo probíhat již v této části vývoje.

Implementaci lze taktéž chápat jako vývoj a včleňování do již existujícího softwaru. Sice dochází k včleňování nových věcí, avšak i ty musí projít celým procesem testování a následného uvedení do provozu.

Dobře provedená implementace ovlivňuje celý start využívání softwaru, který pak je mnohem méně problémový. Aby byla implementace dobře provedena, je dobré se držet některých základních bodů:

- Stanovení plánu implementace a jeho následné dodržování (plán se může v závislosti na požadavcích klienta mírně odlišovat od původního plánu)

Na začátku implementace je potřeba stanovit plán, co se má v jaké části zvládnout a kdo je za danou věc zodpovědný. Jen tak pak tým nebude na konci procesu překvapen, co se nestihlo naimplementovat, což poté vede k nutným přesčasům pracovníků a vysoké chybovosti softwaru, protože vše pak probíhalo ve spěchu bez pečlivého testování.

- Promyšlení strategie vývoje
- Konzistentní podpora projektu od vedení
  Vedoucí skupiny, která má za úkol implementaci, by měl být podporujícím článkem celé skupiny. Měl by se starat o blaho svého týmu a dávat mu to patřičně najevo. Neměl by také často měnit názory a postupy, aniž by je prokonzultoval s ostatními členy implementačního týmu.
- Dodavatelé i zákazníci jsou na jedné lodi
  Mezi oběma tábory musí probíhat včasná a jasná komunikace. Všichni přeci mají za cíl úspěšnou implementaci softwaru
- Odměnění každého člena z týmu, kdo se podílel na implementaci
  Motivace dokáže téměř zázraky. Nemusí se však jednat přímo o finanční odměnu (to se dokonce i nedoporučuje), avšak skvělou prémií může být účast na konferenci, či nějaké školení. Tým se tak postupně bude zlepšovat a bude připraven čelit větším a větším výzvám.

1.5 Testování

Testování softwaru jako celku. Je potřeba zjistit reakce systému na obvyklé i neobvyklé situace, které mohou v živém prostředí nastat. Cílem této etapy je odchycení a následná náprava chyb, které při testování nastaly. Software by měl být již zabezpečený proti nesprávným vstupům, přesto je potřeba nasimulovat vkládání chybných vstupů a a následně pozorovat, jak se s nimi software vypořádá.

Při testování klient obvykle nachází další požadavky, které nebyly při analýze specifikovány, protože k nim například nedochází tak často, či jiné případy užití, které se při analýze nepodařilo odhalit. Po nalezení těchto nových případů užití následuje komunikace se zákazníkem.

Víceméně jsou dva způsoby řešení vzniklé situace:

1. dohoda o vývoji do další verze projektu
2. kroky zpět při vývoji (pozdržení projektu), ale splnění požadavků do aktuální verze

Při testování a dodatečném vývoji dalších případů užití se může vyskytnout otázka, kdy poznat, že je software připravený pro zavedení ke klientovi. Wiegers (2008, s. 125), specifikuje tyto náznaky:

- *„Pokud už uživatelé nemohou přijít na žádný další případ užití, nejspíš máte hotovo. Uživatelé většinou případy užití hlásí dle důležitosti, ty nejdůležitější jako první.*
- *Pokud uživatel přijde s novým případem užití, ale všechny příslušné funkční požadavky už jste odvodili z jiných, nejspíš máte hotovo. Tyto „nové“ případy užití mohou být ve skutečnosti jen nové cesty skrz některý z už nahlášených případů užití.*
- *Pokud uživatelé jen opakují témata, která už jste probírali při předchozích rozhovorech, nejspíš máte hotovo.*
- *Pokud jsou všechny nově navrhované funkce a uživatelské nebo funkční požadavky mimo rozsah projektu, nejspíš máte hotovo.*
- *Pokud uživatelé navrhují funkce, které by mohly přijít na řadu „někdy v budoucích verzích systému“ (a nikoliv v „konkrétním systému, o němž mluvíme právě teď“), nejspíš máte hotovo, alespoň co se týká požadavků na plánovanou verzi.“*

Po odstranění všech zjištěných nedostatků je software připraven pro jeho zavedení.

1.6 Zavádění systému

Zavádění systému je další velmi důležitá etapa, která by se neměla podcenit. Jedná se o instalaci systému do reálného prostředí, jeho zavedení a zpřístupnění jeho uživatelům. Součástí této etapy je také školení pracovníků, kteří budou s tímto softwarem pracovat.

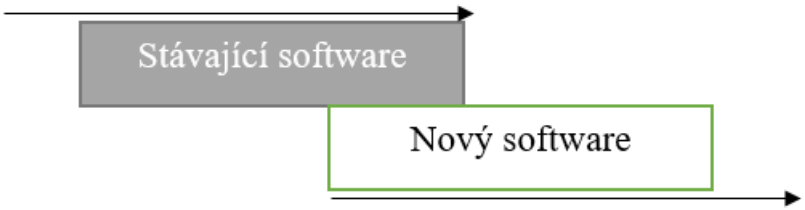
Co se týče školení pracovníků na nový systém, je vhodné nejprve proškolit vedoucí pracovníky, pracovníky, kteří budou mít v systému vyšší práva a až teprve poté proškolit

ostatní. Pro vedoucí pracovníky je také vhodné, aby školili ostatní osoby. Tímto přístupem si upevní čerstvě nabyté zkušenosti z vlastního poznání systému.

Zavádění může probíhat několika způsoby:

- Souběžný způsob

Ve firmách je používán stávající software s nástupem nového. Tento způsob neohrožuje příjmy klienta, protože v případě problémů s novým softwarem bude používán ten stávající. Jedná se tedy o velmi bezpečný způsob. Naopak pro zaměstnance a osoby pracující se softwarem, je tento způsob velmi náročný, protože musí dělat dvakrát totéž, což může vést k odmítání softwaru. Řešením může být najímání externích pracovníků pro vykonávání duplicitních operací.



Obrázek 1: Souběžný způsob zavádění softwaru
Zdroj: vlastní

- Pilotní zavádění

Systém se nasadí jen v určité části firmy. Poté, co bude vyladěn, může dojít k nasazení systému do celé společnosti. Při tomto způsobu je potřeba vybrat takovou část firmy, kde lze software co nejlépe otestovat na co nejvíce možných problémech a operacích.

- Postupné zavádění

Postupné zavádění systémů probíhá především u velmi složitých případů. Do reálného prostředí se nasadí pouze část systému, která je využívána jen k části operací. Postupem času se nasazují i další části.

Obrázek 2: Postupné zavádění softwaru
Zdroj: vlastní

- Nárazový způsob

Používání stávajícího softwaru se přeruší a začne se používat software nový. Tento způsob je velmi riskantní, nicméně šetří pracovní sílu. Tento způsob nasazení je vhodný pouze na výborně otestované systémy, kde bude riziko chyby minimální.

Obrázek 3: Nárazový způsob zavádění softwaru
Zdroj: vlastní

1.7 Zkušební provoz

Jedná se o dobu, kdy je poskytovatel softwaru povinen odstranit všechny chyby oproti zadání zjištěné během provozu, a to v co nejkratším možném čase. Tato doba bývá specifikována i ve smlouvě.

1.8 Provoz a údržba

Jedná se o vlastní důvod, proč byly všechny etapy vykonány. Jedná se o nejdelší etapu života softwaru. Do této etapy lze zařadit zabezpečení systému před neoprávněným přístupem, zajištění všech přístupovým práv k aplikacím, archivaci dat pro případ náhlého výpadku, či opětovné školení uživatelů.

Co se týče nových požadavků na systém, které je potřeba programovat, přesouvají se tyto požadavky na začátek životního cyklu celého vývoje.

2 Metodiky vývoje softwaru

Metodika obecně představuje šablonu, která se musí přizpůsobit danému týmu lidí, kultuře, projektu či účelu. Tato metodika pak vymezuje a popisuje souhrn všech etap, zásad, pravidel, dokumentů, které pokrývají celý životní cyklus SW.

Tyto metodiky se dále dělí na rigorózní a agilní.

Tradiční metodiky představují vývoj softwaru jako inženýrskou disciplínu s pevně stanovenými principy a metodikami. Naopak Alistair Cockburn (Cockburn, 1999) pohlíží na vývoj software jako na kooperativní hru s omezenými zdroji založenou na invenci a komunikaci. Jako příklad uvádí skupinu horolezců. Mají definovaný jasný cíl, kam se chtějí dostat a vědí přesný okamžik, kdy to skončí. Jakmile již budou stoupat k vrcholu, pravděpodobně se nebudou vracet do údolí pro další lana a cepíny, ale budou se snažit využít všechny zdroje, které nesou na svých bedrech.

Buchalcevová (2005, s. 62-64) níže popisuje vzájemné rozdíly mezi těmito metodikami:

Tabulka 1: Rigorózní vs. agilní metodiky

	Rigorózní metodiky	*Agilní metodiky*
1. Proces vývoje software	Přesně a podrobně definovaný proces, lze ho opakovat.	Empirický proces, není možné ho opakovat, ale vyžaduje adaptaci.
2. Změny a požadavky	Minimalizace změn (sběr požadavků předem a plánování předem), změny jsou součástí řízení změn.	Akceptace změn (přírůstkové shromažďování požadavků, plánování pro iteraci) přehodnocení požadavků, snaha o umožnění změn (v závislosti na nových znalostech).
3. Zapojení zákazníka na řízení projektu	Bez zásahu. Nedůvěra v zákazníka - Po podpisu smlouvy a vymezení dokumentu specifikace požadavků v počáteční fázi, zákazníka zajímá až konečné řešení.	Participace a spolupráce - Zákazník se aktivně zapojuje do celého projektu, může měnit priority funkcí při každé iteraci.
4. Kvalita	Více zaměřeny na kvalitu vlastních procesů, než na výsledek pro zákazníka.	Zaměření na priority zákazníka, na užitnou hodnotu pro zákazníka, tedy na výslednou kvalitu produktu.
5. Způsob a rozsah řešení	Podrobný popis procesů a činností, složité řešení. Při vývoji obsaženy všechny funkce, snaha o zabudování budoucích požadavků.	Eliminace nepotřebných činností, jednoduché řešení. Při vývoji zahrnuty jen potřebné funkce, snaha o minimalizaci, žádné začlenění budoucích požadavků.
6. Rozložení teamového know-how	Každý člen týmu má přiřazeny činnosti, v rámci role, kterou zastává. Požadavek na specializaci zaměstnanců. Převládá písemná forma komunikace (dokumentace).	Sdílení znalostí v týmu, spolupráce (kooperace) a aktivní komunikace v rámci týmu, společné řešení problémů, řízení a integrace znalostních toků.
7. Lidský faktor	r Pohled na lidi jako na sekundární faktor, procesy doplňovány rozsáhlou dokumentací. Jedinci jsou nahraditelní.	Lidé jako primární a klíčový faktor úspěchu projektu. Využití schopností a dovedností jedinců (individualit), důraz na znalosti, kvalifikaci.
8. Způsob vývoje	Vodopádový životní cyklus, iterativní s dlouhými iteracemi.	Přírůstkový (inkrementální) vývoj s velmi krátkými iteracemi.

Zdroj: Buchalcevová (2005, s. 63-64)

2.1 Rigorózní metodiky

Jak je zmíněno výše, u tohoto typu metodik vzniká velké množství dokumentace. Předpokládá se opakovatelnost procesů a možnost definovat všechny požadavky již na samém počátku. V průběhu také vzniká velké množství meziproduktů, kvůli kterým se později začne ztrácet celý cíl vývoje – tedy vytvořit software odpovídající potřebám klienta.

Neprobíhá komunikace se zákazníkem při vývoji. Zákazník se zapojuje až během předání aplikace, kdy dostane již hotový produkt.

2.1.1 Model „Napiš, oprav“

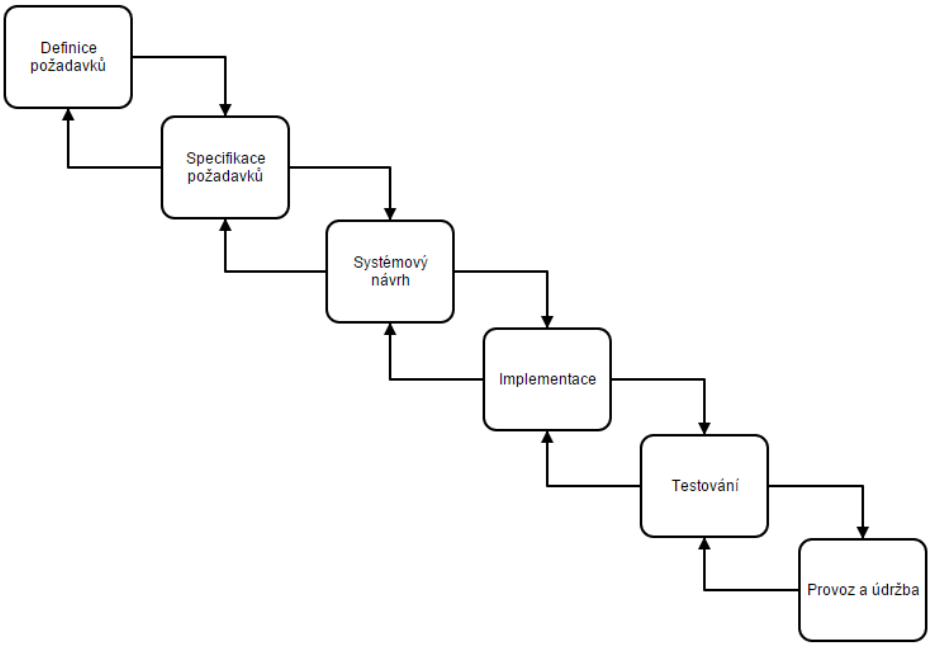
Jedná se o jeden z nejjednodušších modelů vývoje. Tento model se hodí především pro vývoj jednoduchého softwaru, který nevyžaduje tolik pozornosti jako složitější programy.

Implementace -> Dodání -> Oprava chyb

2.1.2 Vodopádový model

Ve vodopádovém modelu vývoje má každá etapa jasně definovaný cíl. Na konci každé etapy dochází k vyhodnocení jejího cíle, zda byl splněn, či nikoliv. Po schválení, či přepracování a opravení zjištěných nedostatků je možné přejít do další etapy vývoje. V případě pozdějších zjištění je možné se vrátit do předchozí etapy.

Největšími výhodami tohoto modelu jsou bezesporu jednoduchost a jakýsi řád, podle kterého postupovat. Naopak tento model vývoje je velmi nepružný, takže není schopen reagovat na nově nalezené požadavky od zákazníka.

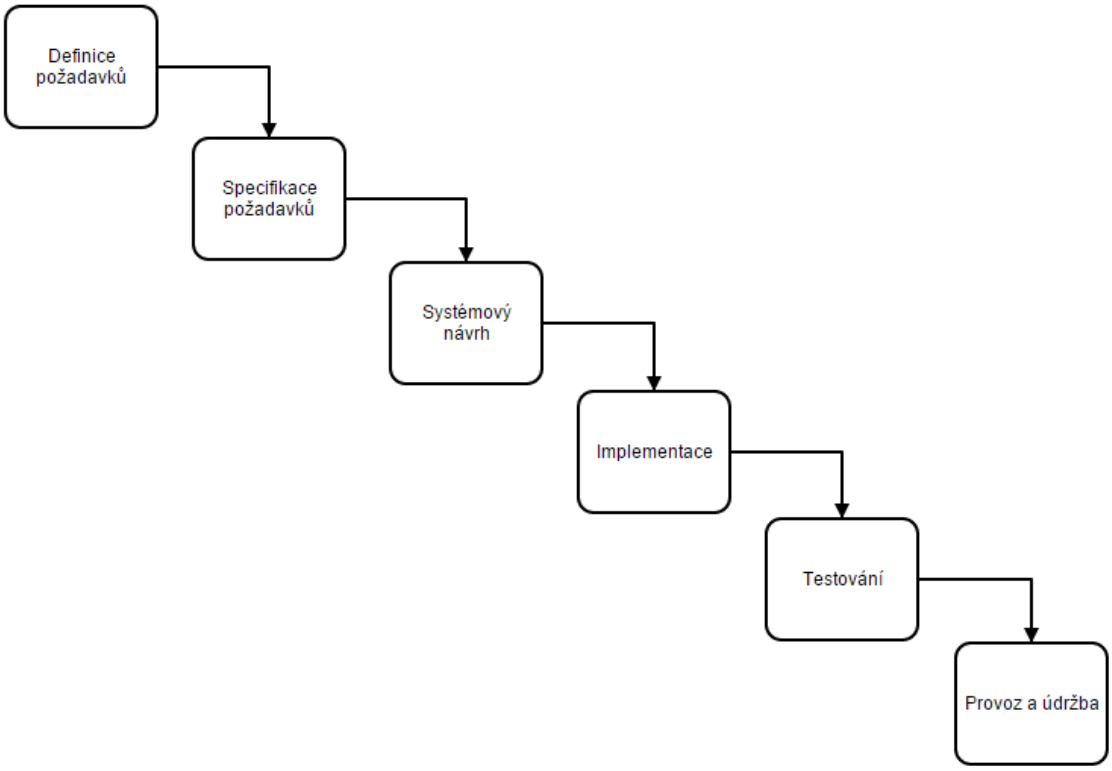


Obrázek 4: Vodopádový model
Zdroj: vlastní

2.1.3 Striktní posloupnost fází (Stagewise)

Striktní posloupnost fází, jak název napovídá, je model s přesně určeným postupem. Na rozdíl od vodopádového modelu se není možné vracet do předchozí etapy.

V tomto modelu chybí jakákoliv revidace požadavků, zpětné vazby či hledání rizik.

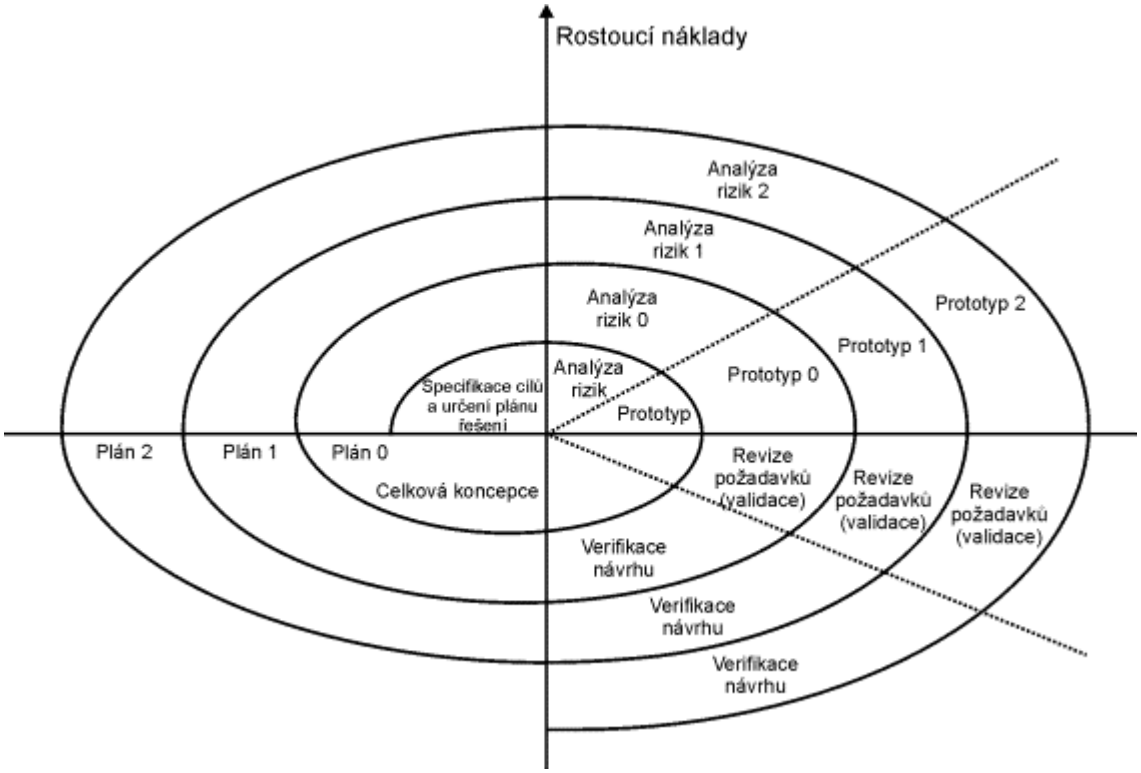


Obrázek 5: Striktní posloupnost fází
Zdroj: vlastní

2.1.4 Spirálový model

Základem tohoto modelu je neustálé opakování již proběhlých kroků při přidávání nové části na vyšší úrovni vývoje. Zjednodušeně to lze popsat jako opakující se model vodopád při vstupu do další etapy a dalšího rozšíření softwaru.

Tento model lze použít i pro složitější projekty. Opakovaně dochází k revizi požadavků zákazníka a k ověření, že se projekt vyvíjí správným směrem. Z toho však plyne i velká nevýhoda, konkrétně vysoká angažovanost zákazníka při jednotlivých etapách.



Obrázek 6: Spirálový model
Zdroj: Šmíd (Šmíd, dat. vyd. neuveden)

2.2 Agilní metodiky

Agilní metodiky na rozdíl od tradičních, nestanovují všechny požadavky na počátku projektu. Vymezí se cíl, maximální náklady a čas, do kdy má být projekt hotový. Tým je v průběhu projektu neustále v kontaktu se zákazníkem, kdy vyhodnocují a přehodnocují požadavky a priority.

Při těchto metodikách nemusí vznikat téměř žádná dokumentace. Nicméně doporučuje se zaznamenávat požadavky zákazníka pro případná nedorozumění. Hlavní dokumentací je ale především přehledný kód (je zde tedy kladen důraz na pečlivost a jasnost kódu pro ostatní programátory).

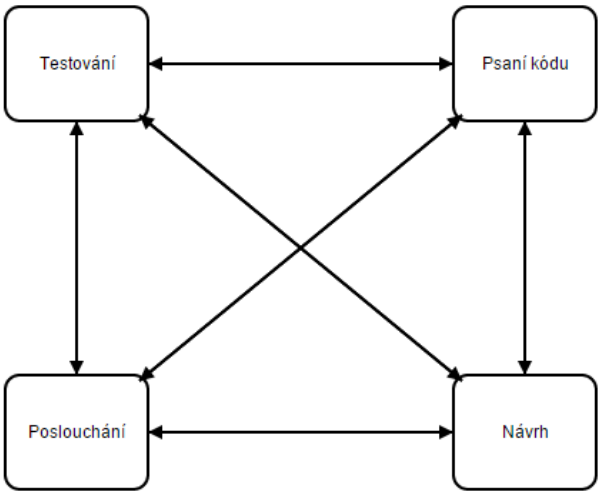
Obecně tyto metodiky slouží k rychlému programování, k téměř okamžité odezvě na požadavky zákazníka v celém průběhu vývoje.

2.2.1 Extrémní programování

Jedná se asi o nejznámější agilní metodiku. Jejími hlavními pilíři je komunikace, odvaha, jednoduchost a zpětná vazba. Jako pátý, mírně opomíjený, avšak velmi důležitý pilíř, je zmiňován respekt.

V extrémním programování lze nalézt čtyři základní činnosti, které tým provádí téměř nepřetržitě.

- Testování
- Psaní zdrojového kódu
- Poslouchání
- Navrhování

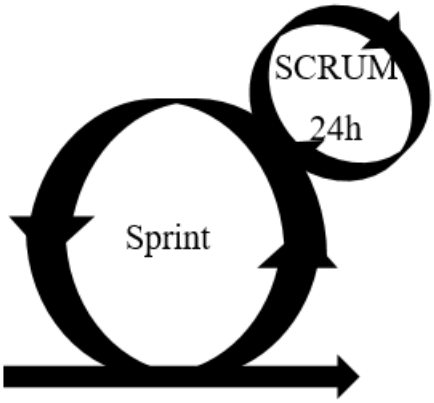


Obrázek 7: Extrémní programování
Zdroj: vlastní

2.2.2 SCRUM

Základním prvkem této agilní metodiky, je každodenní krátký meeting, nazvaný scrum. Každý člen týmu musí sdělit svým kolegům v týmu, čím se předešlý den zabýval a čím se bude zabývat dnešní den.

Vyvíjí se zpravidla v malých týmech. Jednotlivým etapám vývoje se říká sprinty.



Obrázek 8: SCRUM
Zdroj: vlastní

2.2.3 Lean development

Hlavní myšlenkou této metodiky je především odstranění zbytečného plýtvání časem v jednotlivých etapách vývoje. Jedná se o soubor pravidel, které umožňují zefektivnění a zrychlení vývojového procesu. Mezi některá z těchto pravidel patří eliminace zbytečného programování, úkonů, rychlé rozhodování, či pohled na systém jako celek.

2.2.4 Vývoj řízený vlastnostmi

U této metodiky, jak z názvů vyplívá, je vývoj řízen vlastnostmi projektu. Vlastnost lze pak vnímat jako užitnou hodnotu z pohledu zákazníka.

U této metodiky je rozdíl od extrémního programování, nebo SCRUM v tom, že si jednotliví programátoři nevybírají část projektu, na které budou pracovat.

2.2.5 Crystal metodiky

Do této skupiny patří metodiky, které byly navrženy pro jednotlivé projekty. Vychází z toho, že pro každý unikátní projekt je zapotřebí také unikátní metodika.

Tyto metodiky vymyslel a specifikoval Alistair Cockburn. Každá z těchto metodik obsahuje soubor rad, postupů, a doporučení a také je vhodná k jinému řešení, velikosti vývojového týmu, či složitosti projektu. Dle obsáhlosti ke každé metodice přiřadil barvu – například čirá, žlutá, oranžová, hnědá, fialová. Čím je barva tmavší, tím více se hodí metodika pro rozsáhlejší, robustnější projekty.

Tyto metodiky však neposkytují přesný postup, ale mohou být jakýmsi receptem, jak dosáhnout cíle.

3 Softwarové licence

V této kapitole jsou zjednodušeně popsány některé základní, avšak velmi často využívané licence.

Softwarová licence je obecně smlouva, mezi dvěma či více stranami. Nejčastěji to však bývá mezi autorem programu (osobou či společností, která program vytvořila) a jeho uživatelem. Tato smlouva poté vymezuje a stanovuje podmínky, za kterých může uživatel program využívat.

Druhá definice softwarovou licenci přirovnává k nástroji, díky kterému je možné software, chráněný Autorským zákonem využívat, či redistribuovat.

3.1.1 Proprietární licence

Tato licence vymezuje, jak a kdy smí program daná osoba použít. Jak uvádí Malý (Malý, 2011) stačí popsat *„Tento program jsem udělal já, X.Y., a pan Jan Novák, r.č. toato, jej může používat obvyklým způsobem na svém počítači, a to každé první úterý v měsíci mezi 12:30 a 14:00.“*.

Ačkoliv se tato definice může zdát zvláštní, je plně právně závazná. Jejím úskalím může být, pokud osoba udělí takovouto licenci, která však bude příliš dlouhá, nikdo ji nebude číst, tedy ani dodržovat.

3.1.2 No whining

Tato licence ve zkratce znamená dílo užívat, šířit a měnit pod jednou podmínkou. Uživatel si nebude na nic autorovi ztěžovat.

3.1.3 BSD

Jedná se o licenci pro svobodný software. V licenci je uvedeno, kdo a kdy software udělal. Dílo je možné využívat, dále šířit. Autor se zříká veškeré odpovědnosti, co se se softwarem stane.

3.1.4 MPL

MPL, nebo-li Mozilla Public License, je taktéž licence pro svobodný software. Jak uvádí Malý (Malý, 2011) *„Hlavní rozdíl proti BSD je ten, že pokud do svého SW zahrnete software licencí MPL, musíte tuto část (i s případnými změnami) dál šířit jako MPL.“*.

3.1.5 Open Source software

Jedná se o software, jehož licence nám dovoluje číst a hlavně i upravovat jeho zdrojový kód. Licence nám však nedovoluje software za úplatu šířit – musí zůstat stále Open Source.

3.1.6 Public domain

Obecně se jedná o dílo, u kterého již uplynula doba vlastnického práva. Tato licence dovoluje software užívat a to i ke komerčním účelům či u něho provádět další změny. Není však samozřejmostí, že je vždy možné nahlédnout do zdrojových kódů.

3.1.7 Shareware

Program, který je šířen jako Shareware, bývá na nějaký čas poskytován zdarma, po čase však vyzve k uhrazení odměny pro autora softwaru. Program je možné dále šířit, využívat, nicméně nesmí být modifikován. Program je šířen ve formě spustitelného souboru, tudíž není možné prohlížet jeho zdrojový kód.

3.1.8 Freeware

Ve většině případů se jedná o software, který je možné bezúplatně používat (většinou to platí jen pro nekomerční využití). Ačkoliv je program šířen bezúplatně, vztahují se k němu licenční podmínky, které vymezují způsob použití programu.

Tento typ programu bývá šířen ve formě spustitelného souboru (není tudíž možné nahlédnout do kódů).

3.1.9 Demo

Program, který je distribuován jako demo, je vlastně jakousi ochuzenou verzí své plné verze. Tato ochuzená verze je šířená zdarma, nicméně neposkytuje takový komfort. Zpravidla slouží jen pouze k vyzkoušení programu, ale nepokryje všechny požadavky, které zákazník na software může mít. Plná verze programu je již za úplatu – jedná se tedy na rozdíl od výše zmíněných o komerční software.

4 Představení společnosti Actis, s.r.o.

Tato kapitola tvoří jakýsi úvod do praktické části. Cílem této kapitoly je představit, jakým způsobem se společnost měnila v čase, jak se změnilo myšlení a produkty firmy. Dále pak blíže představuje platformu, na které jsou produkty společnosti vyvíjeny.

Obecně se společnost zabývá softwarovými řešeními na míru v oblasti procesů a informačních technologií. Podporuje tak své zákazníky ve spolupráci a zvyšování výkonnosti firmy.

4.1 Historie

Společnost vznikla v roce 2005 jako dceřiná společnost větší společnosti. Actis, s.r.o. měla být společnost čistě pro vývoj, zatímco mateřská společnost pro prodej produktů a jejich konzultace. Za posledních deset let se firma formovala a měnila svoji podobu. Došlo k odtržení od mateřské společnosti a rozšíření spektra vykonávaných činností ve společnosti. Již od založení firmy stojí v čele Ing. Václav Ševčík.

4.2 Produkty firmy

Společnost Actis se zabývá celkem širokým spektrem produktů. Implementuje k zákazníkům partnerské produkty, ale především vyvíjí produkty vlastní.

Vlastní produkty společnosti se dají rozdělit do dvou základních skupin:

- Produkty na platformě MyDSy
- Produkty programované v IBM Notes

4.2.1 Produkty na platformě MyDSy

Platforma MyDSy je postavena na opensource frameworku Grails. Jedná se o obchodní značku Actisu. Jak uvádí některé výhody pan Ševčík (2015, interní zdroj):

- Nezávislost na zvolené databázové technologii (Oracle, MSSQL, MySQL, ...)
- Opensource – řešení lze v základní verzi používat bez nákupu licencí třetích stran
- Bez „překvapení“ – řešení je široce testováno v rámci správnosti funkcí (backendu) i uživatelského rozhraní.
- Workflow – všechna obchodní logika je popsána v přehledných diagramech, které tvoří základ technické dokumentace
- Bez instalace – pro přístup uživatelů se využívá webový prohlížeč, většinou s implementovaným jedním přihlášením (single-sing-on)
- Multijazyčnost – řešení je nastaveno pro více jazyků, výchozí je čeština

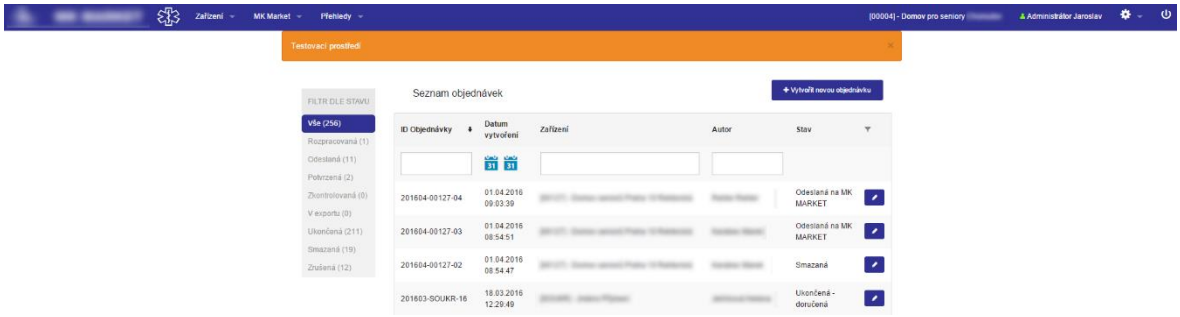
Další nespornou výhodou je možnost napsat automatické testy na software na této platformě. Během pár minut se celý software díky tomu může celý protestovat. Umožňuje tak rychlejší odchytání chyb. Nebylo by v lidských silách software otestovat tak rychle a kompletně, jako u těchto testů.

Již z tohoto soupisu je patrná výhodnost použití platformy MyDSy pro níže představené produkty společnosti (společnosti mají pro účel práce změněný název)

1. Portál

   Produkt pro společnost XY Pomůcky. Jedná se o velkoobchod se zdravotnickým materiálem a výdejnu zdravotnického materiálu. Pomůcky a materiál, které si zařízení objednají, následně smění za lékařem předepsané poukazy. Tyto poukazy poté XY Pomůcky uplatňuje na pojišťovnu.

   Společnost Actis vytváří systém, který zautomatizuje celý proces prodeje výrobků. Tento systém musí umět spolupracovat s dalšími systémy, např. Byznys, který slouží pro evidenci skladových zásob. XY Pomůcky však vydává zdravotnický materiál nejen na poukazy, ale i fakturu či na bonus z minulých objednávek. Systém proto musí umět rozsáhlou práci s databázemi, být uživatelsky přívětivý (do budoucna s ním budou pracovat samotná zařízení). Dále musí umět počítat přesné částky pro fakturaci, generovat účtenky, štítky, reporty, poukazy do PDF souboru, či seznamy do XLS.



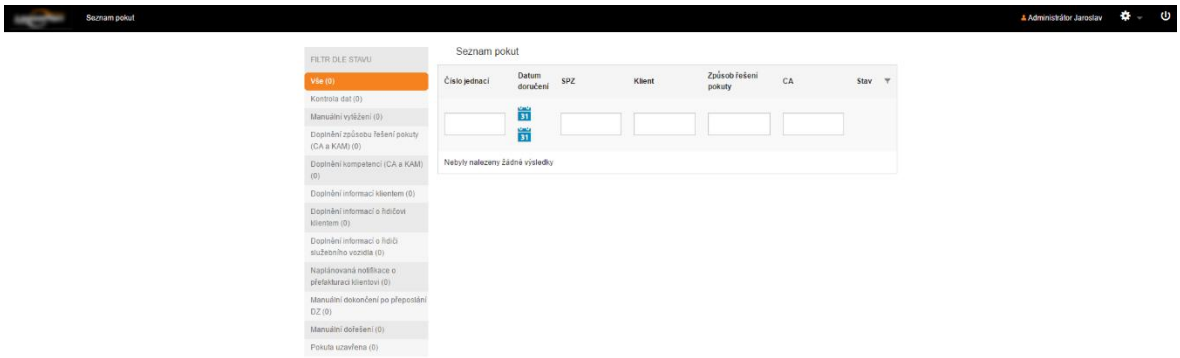
Obrázek 9: Pohled na Portál
Zdroj: Vlastní

2. Správce pokut

   Správce pokut je produktem pro společnost Leasing, což je mezinárodní firma, která v současné době působí ve 32 zemích. Jejími hlavními službami je zajišťování operativního leasingu svým klientům a komplexní správa vozových parků.

   Společnost Actis vytváří pro Leasing systém, který bude umět velmi urychlit práci s příchozími pokutami. Společnosti Leasing, jako vlastníkovi automobilu, přijde výzva k zaplacení pokuty. Společnost z tohoto dokumentu musí vytěžit údaje jako státní poznávací značka, čas, datum a místo spáchání přestupku, typ přestupku, bankovní číslo či variabilní symbol, kam částku uhradit. Dále pak dojde k prověření, kdo má vozidlo zapůjčené, zda se jedná o auto služební, či klienta. Nakonec pak výsledné řešení a generování dokumentu pro Policii ČR.

   Systém nyní umí vytěžit z dokumentu potřebné údaje, propojit se s interním systémem klientů, doplnit klienta, či požádat klienta o nahlášení řidiče. Vše automaticky. Dále pak generovat dokumenty napříč celým procesem (např. oznámení o zjištění řidiče). Celý proces se tímto výrazně zrychluje a omezuje riziko vzniku chyby při opisování údajů.



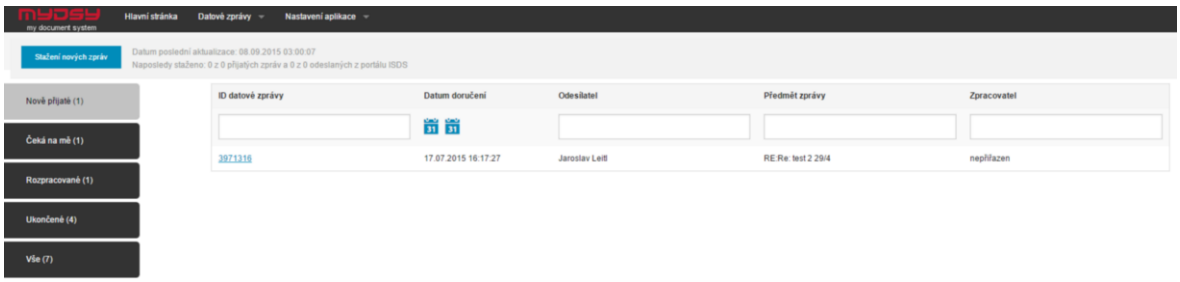
Obrázek 10: Pohled na Správce pokut
Zdroj: Vlastní

3. MyDSy Datové schránky

MyDSy Datové schránky umožňují lepší přehled a archivaci datových zpráv delší než 90 dní. MyDSy datové schránky jsou integrovatelné s interními systémy, což nabízí mnohá využití.

Například MyDSy datové schránky pro společnost Mediaprix. Tato společnost zajišťuje adresnou či neadresnou distribuci tiskovin.

Datové schránky již jsou integrované s interním systémem Mediaprix, což umožňuje přidělení jednotlivých zpráv a přehled, kdo je zodpovědný za příslušnou datovou zprávu.



Obrázek 11: Pohled na MyDSy Datové schránky
Zdroj: Vlastní

4.2.2 Produkty programované v IBM Notes

Jedná se o produkty, které byly napsány v IBM Notes a zatím nebyli přepsány do platformy MyDSy.

Obecně produkty v IBM Notes nenabízejí tolik možností, jsou velmi závislé na omezených prvcích v Notes. Další nevýhodou je nemožnost napsání testů na software. Testovat lze tedy jen uživateli, což může být někdy problematické, protože se v každé společnosti nemusí nacházet ochotná osoba, která bude tuto funkci vykonávat.

1. e-PostOffice

   Produkt e-PostOffice je jakási domácí poštovní kancelář. Umožňuje odesílání obyčejného i doporučeného psaní z pohodlí domova. Uživatelům nabízí přehledné statistiky či ukládání adres kontaktů. Je ovšem také možné systém napojit na interní software tak, že lze automaticky odeslat zprávy přímo ze systému do poštovní schránky příjemce.
2. Docházka

Docházka, jak už název vypovídá, má za úkol zaznamenávat příchody a odchody zaměstnanců. Mimo jiné umí generovat různé reporty, náležitosti, které souvisejí s pracovní dobou zaměstnanců.

Obecně se v Actisu již upouští od vývoje v IBM Notes, takže jen některé projekty pouze doznívají. Produktů na této platformě existuje ve firmě mnohem více, ale uvádět je do této práce by začínalo pozbývat smysl.

4.3 Cíle a plány

Cílem společnosti je především rozšířením MyDSy Datových schránek k co nejvíce zákazníkům. Dále pak dokončit vývoj a uskutečnit finální předání ostatních produktů, které jsou nyní ve vývoji.

Velký, zatím však nevyužitý potenciál má také produkt e-PostOffice, který společně s MyDSy Datovými schránkami může tvořit doplňující se produkty v oblasti tištěné korespondence, ale i korespondence nejen od státních institucí, prostřednictvím datových schránek.

Dlouhodobějším cílem společnosti je pak rozšíření produktů, které mají dopomoci ke zvýšení výkonnosti firmem zákazníků, zlepšení komunikace či zlepšení týmové spolupráce.

5 Potřeby a požadavky zákazníků

Každá firma má na počátku procesu nějaká očekávání, co by měl nově vzniklý systém umět zvládat. Každá firma si je vědoma, kvůli čemu chce přejít na nový software, a to samozřejmě za co nejkratší časové období a minimum prostředků. Málokterá však má v tomto pohledu reálná očekávání.

Potřebou zákazníka se může rozumět například snížení časové náročnosti úkonů, finančních prostředků. Zatímco požadavky zákazníka zase označují požadavky na samotný software, tedy co by měl umět, počítat, odesílat, vytěžovat. Obojí však nemusí být jak pro zákazníka, který si software objednal, tak pro firmu, která software dodává, zřejmé. Velice záleží na tom, zda společnost, která si vývoj objednala, má své IT oddělení (či alespoň admina, správce), či nikoliv.

5.1 Seznámení se s klientem

Na počátku celého procesu stojí seznámení se se zákazníkem. Ten se většinou o firmě Actis dověděl díky referencím svého obchodního partnera, známého či při účasti na konferenci. Málokdy se dokáže uzavřít nějaký kontrakt po nalezení kontaktu například na webu.

Při prvotním seznamování se nastíní základní požadavky na systém, aby bylo potvrzeno, že je plánovaný vývoj reálný a uskutečnitelný. Seznamování dále probíhá prostřednictvím informativní schůzky, kde se podrobněji popíší a zaznamenají všechny primární požadavky. Na této informativní schůzce by již měli být přítomni zodpovědní zástupci od klienta, který může usměrnit některé náležitosti, týkající se firemního IT.

Na této schůzce by se již měl dohodnout termín, kdy se provede prvotní analýza organizace.

5.2 Analýza organizace pro vznikající software

Analýza organizace a vznikajícího softwaru probíhá jinak, pokud klient má svoje IT oddělení a jinak, pokud takové oddělení ve společnosti chybí. V obou dvou případech je však důležité

tuto fázi nepodcenit a zaměřit se na každý detail, který by se mohl v dalších krocích rozvinout ve větší a těžce řešitelný problém.

Analýza má obecně dva stupně:

1. Prvotní analýza, tedy analýza toho, co zákazník všechno chce, jak by to mělo vypadat, vše na uživatelské úrovni. Mělo by být vše pečlivě zaznamenáno, protože se k tomuto dokumentu vývojáři a konzultanti často obracejí v pozdějších fázích (například při dohadech, zda to bylo původně dohodnuto, tedy v ceně vývoje či se jedná o úplně novou věc, tedy další položku, za kterou zákazník platí).
2. Druhá úroveň analýzy je spíše technologická. Vše se přepisuje do programovací terminologie. O tom, co by měl software umět, se již mluví jako o procesech, entitách, vztazích. Ukazuje to tak přesný dokument, podle kterého následně programátoři postupují. Díky tomu mohou vyvíjet software, aniž museli bezpodmínečně znát do hloubky fungování firmy zákazníka. Další výhodou vzniklého dokumentu je, že se pravděpodobněji nezapomene na některé prvky softwaru.
   Ukázku z výsledného souboru technologické analýzy lze nalézt pod přílohou A.

Některé společnosti, i kvůli finanční náročnosti druhé úrovně analýzy, tento krok odmítly. Nutno podotknout, že tyto rozhodnutí dělají spíše společnosti, které nemají svoje IT oddělení. Zástupci, kteří poté o tomto kroku rozhodují, nemusejí vůbec tušit, do čeho se vlastně bez prvotní analýzy pouštějí.

Zrovna takovým příkladem vystupuje společnost XY Pomůcky, která trvala na překročení druhého stupně analýzy z důvodu vysoké jednorázové finanční náročnosti. Společnost Actis se nakonec zákazníkovi rozhodla vyhovět, což později ukázalo několik nedostatků, které by měly odradit od přeskakování příští analýzy třeba u dalšího zákazníka:

- Nelze určit odpovídající cílovou částku za software
- Nelze stanovit konečné datum předání softwaru
- Pro vývojářskou společnost je zde menší povědomí o fungování firmy zákazníka
- Neustále se objevují nové a nové požadavky na software (které znamenají vysokou zátěž, protože se musí prvky vkládat do již existujícího softwaru)

- Dochází k změnám požadavků (opět finančně náročné, zákazník dá požadavek, který si zaplatí, poté však názor změní a opět platí)
- Během implementace se objevují nové a nové případy užití (opět vývoj a vysoké finanční náklady pro zákazníka, ale i časové pro Actis)
- Jako ve společnosti XY Pomůcky může vyjít najevo, že lidé, kteří objednávají software a stávají se hlavním komunikačním článkem s Actisem, nemají dostatečné informace o fungování firmy, principů, postupů.
- Dochází k vytahování „kostlivců ze skříně“ příliš pozdě (opět vysoká finanční zatížení)

Obecně se tedy ušetří náklady při analýze, které jsou však brzy převýšeny náklady za zbytečný vývoj. Stejně tak i pro Actis, který spoustu prvků kvůli nedostatečnému povědomí vyvíjel téměř zbytečně, a klient to tedy neproplatí.

5.3 Návrh softwaru

Samotný návrh softwaru je již vytvářen jen jako myšlenková forma mezi vývojáři. Prakticky se nikam nezapisuje, ani nezaznamenává. Přesto to však neznamená, že je tento krok přehlížen a přeskakován, to nikoliv.

Vývojáři se během několika schůzek rozhodnou pro nejlepší směr vývoje, tak, aby korespondoval s požadavky zákazníka. Poté se již může přejít k samotnému vývoji.

6 Proces implementace

Poté, co se dokončí návrh systému mezi vývojáři, může se přejít k vývoji. Ten se řídí hlavně prvotní analýzou, v pozdějších krocích také požadavky a úpravami, které se projednávaly se zákazníkem na poslední schůzce (ideálně těchto požadavků a nových případů užití, by mělo být co nejméně).

Jedná se o takřka nejdelší krok ve vývoji softwaru.

6.1 Hladký průběh implementace

Hladký průběh implementace znamená čistou, bezproblémovou a poměrně rychlou implementaci softwaru do firmy zákazníka. Aby proběhla hladká implementace, je zde několik požadavků:

- Obsáhlá, objektivní a dobře provedená prvotní analýza společnosti
- Získání všech potřebných přístupů k existujícímu softwaru zákazníka
- Co nejlepší posbírání požadavků a případů užití a počátku vývoje
- Časté schůzky s klientem, který nahlíží do vyvíjeného softwaru, učí se a ním a umí rychle reagovat na změny

Z výše zmíněných bodů je patrné, že neexistuje zaručený návod pro hladký průběh, který bude fungovat ve všech společnostech.

Poměrně hladký průběh proběhl ve společnosti Leasing. V této firmě je IT oddělení, jehož členové se pravidelně účastní naplánovaných schůzek. Již při počáteční analýze se podařilo poměrně úspěšně obsáhnout budoucího fungování systému. Dále probíhala bezproblémová komunikace s klientem, kdy bylo patrné, co klient vyžaduje.

Problematičtější už bylo získávání přístupů do jejich interního systému, ve kterém spravují mimo jiné své automobily a zákazníky, kteří mají tyto automobily na operativní leasing (složitost systému je vyobrazena v příloze C). Nakonec se připojení k systému podařilo díky

VPN přístupům a vytvoření nových uživatelů, za účelem testování a ověření správnosti funkčnosti spolupráce těchto systému.

6.2 Problematičtější průběh implementace

Problematičtější průběh implementace označuje komplikovanější proces, než u hladkého průběhu. Objevuje se více problémů než u hladkého průběhu. Obecně složitější průběh implementace se objevuje častěji u společností bez vlastního IT oddělení. U takovéhoto průběhu se často objevují tyto symptomy:

- Neustále se objevující nové a nové případy užití
- Měnící se požadavky zákazníka (jednou tam tento prvek není potřeba, příště se bez něho nelze obejít)
- Nesrozumitelná komunikace se zákazníkem (zákazník chce v softwaru něco jinak, ale neumí popsat co a jak by to mělo fungovat, ví jen, že takhle je to špatně)
- Změna kompetentních lidí na straně zákazníka (může dojít ke měnám ve struktuře společnosti a do vznikajícího softwaru je za stranu zákazníka nasazena osoba, která o fungování softwaru, cíli příliš neví)

Za problematičtější implementaci lze považovat implementaci ve společnosti XY Pomůcky. Tato společnost odmítla kvůli finanční náročnosti druhou úroveň analýzy. Bylo tedy na místě snažit se co nejlépe pochopit fungování softwaru z uživatelského hlediska a zasvětit do tohoto hlediska i vývojáře, kteří začali řešit technologickou stránku softwaru.

Do procesu implementace v této společnosti také často vstupovaly nové případy užití a požadavky, které často byli poměrně invazivní a bylo nutné jim přizpůsobit vznikající software.

Kvůli měnícím se požadavkům se cena za dodatečné programování brzy přehoupla přes samotnou cenu technologické analýzy.

6.3 Testování

Testování by mělo probíhat pravidelně a ihned, jakmile se dokončí nějaká část softwaru. V ideálním případě jsou k softwaru napsané automatické testy, které probíhají ještě dřív, než se nově vzniklá část softwaru dostane k zákazníkovi. Další testování poté probíhá členem z Actisu, který prověřuje i provázanost a logiku některých částí softwaru vzhledem k celku (osoba musí chápat alespoň základní fungování klientské firmy). Jakmile se zdá, že je software v pořádku, přejde se k předání ke klientovi. Tam by mělo dojít taktéž k testování a případným úpravám odlišností.

Jakmile se zdá být software připraven pro nasazení, postupuje se k dalšímu kroku.

7 Adaptace a podpora softwaru

Jakmile je software připraven pro nasazení a hlavně pro používání v běžném provozu, je potřeba připravit uživatele na jeho používání. Je také důležité zvolit, jakou cestou se software nasadí (nárazově, současné používání starého a nového systému, částečné používání softwaru).

Je potřeba stanovit reálné datum, kdy dojde k přechodu na nový software a oznámit to všem osobám, které budou se softwarem pracovat. Do tohoto data by se měli naučit se softwarem zacházet. Pro tyto účely můžou výborně pomoci některé nástroje pro zaučení se softwarem:

- Tištěný manuál

  Jedná se o nejčastější formu pomoci pro zaučení se softwarem od Actisu. V každém takovém manuálu jsou popsány jednotlivé role, jejich práva v systému. Dále pak základy od přihlášení, základních operací v systému, až po obsáhlé operace na administrátorské úrovni. Kvůli jednoduchosti je manuál doprovázen četnými obrázky, aby bylo vše zřetelnější a pochopitelnější.

  Nevýhodou takového manuálu je bezesporu obsáhlost. Málokdo má čas a chuť si celý manuál pročíst (jednodušeji to bohužel asi nejde).

  Naopak nevyžaduje takové úsilí při jeho vytváření, tedy ani vysoké náklady.

  Ukázku tištěného manuálu lze nalézt v příloze B.
- Instruktážní video

  Jedná se o sérii videí, které mají uživateli popsat základní operace, či některé nejčastější neduhy. Velmi často musí i tak být doprovázen textovým manuálem pro další případy užití. V praxi se jedná o video vytvořené ze snímané obrazovky, nejčastěji doplněné textem.

  Nevýhodou je náročnost na čas a především nemožnost doplnit video o změny v softwaru (do textového manuálu se dá lehce doplňovat, zde je potřeba vytvořit zcela nové video).
- Interaktivní obrazovka

  Jedná se o spuštění aplikace na pozadí z pohledu aplikace softwaru. Na obrazovce se pak objevují instrukce co zvolit, jak postupovat dál. Většinou se jedná o placený prémiový

software, takže od tohoto typu návodu společnost Actis upustila. Nebyl dostatečný zájem o takto interaktivní manuál.

Poté, co se všechny osoby, které budou se softwarem pracovat, naučí aplikaci ovládat (alespoň základy), může dojít k ostrému nasazení softwaru do firmy.

Jakmile se software nasadí do společnosti, je potřeba počítat s tím, že se ještě objeví některé nedostatky, které se při testování neobjevily a bude je potřeba vyřešit co nejdříve. Jelikož firmy nebývají ze stejného města jako Actis, je potřeba si stanovit některé možnosti efektivní mezifiremní komunikace:

- Telefon

  Společně s e-mailem se jedná o nejčastější komunikační cestu mezi Actisem a zákazníkem. Nabízí se rychlé reakce, rychlá vysvětlení. Naopak po telefonu není možné vidět obraz toho, co se nechová korektně. Pokud je ve společnosti více osob, které se softwarem pracují, je zde také hrozící riziko, že bude každá osoba telefonovat několikrát, takže než se všichni prostřídají, nebude schopen člověk za Actis udělat nic jiného.
- E-mail

  Velmi výhodná cesta, nabízí rychlé reakce na změny, možnost přidání obrazového materiálu do příloh, které velmi pomohou nastínit vzniklý problém.
- Schůzka

  Po nasazení softwaru do společnosti se schůzky se zákazníkem musí stejně konat. Pokud se jim však přiřadí nějaká pravidelnost, je na nich možné probrat všechny věci na softwaru, který mají nějaký nedostatek (obvykle se nejedná o takzvané show-stopper, které se řeší ihned po jejich nalezení).
- Help desk

  Help desk je služba, přes kterou lze nahlašovat problémy s aplikací. Zákazník v ní poté může vidět, zda se vzniklá situace řeší, jak se řeší. Zda je to chyba, či novinka, o které se nevědělo a je ji potřeba teprve nacenit a po odsouhlasení i naprogramovat.

7.1 Hladký průběh adaptace

Hladký průběh adaptace označuje proces, kdy lidé přicházející do styku se systémem, tento systém neodmítají, jsou si vědomi jeho výhod, a chtějí se podílet na jeho zdokonalování a co nejrychlejším začlenění do fungování firmy.

Poměrně hladký průběh adaptace probíhal ve společnosti Leasing. Osoby měly zájem na co nejrychlejším uvedení aplikace do provozu. Byly si zároveň vědomi usnadnění, které by jim software měl přinést. Během celého adaptačního procesu se podílely na testování, aby se odchytaly chyby v co nejkratším časovém období.

Obecně vzato, nelze dát univerzální rady pro hladký průběh adaptace, velice totiž záleží na kultuře každé firmy, osob, které se softwarem přicházejí do styku, a vedení, které o softwaru rozhodovalo. Lze jen doporučit, aby nebylo slibováno to, co není možné realizovat. Vystupovat sebevědomě, nikoliv však povýšeně a především neotáčet se ke klientovi zády.

7.2 Problematičtější průběh adaptace

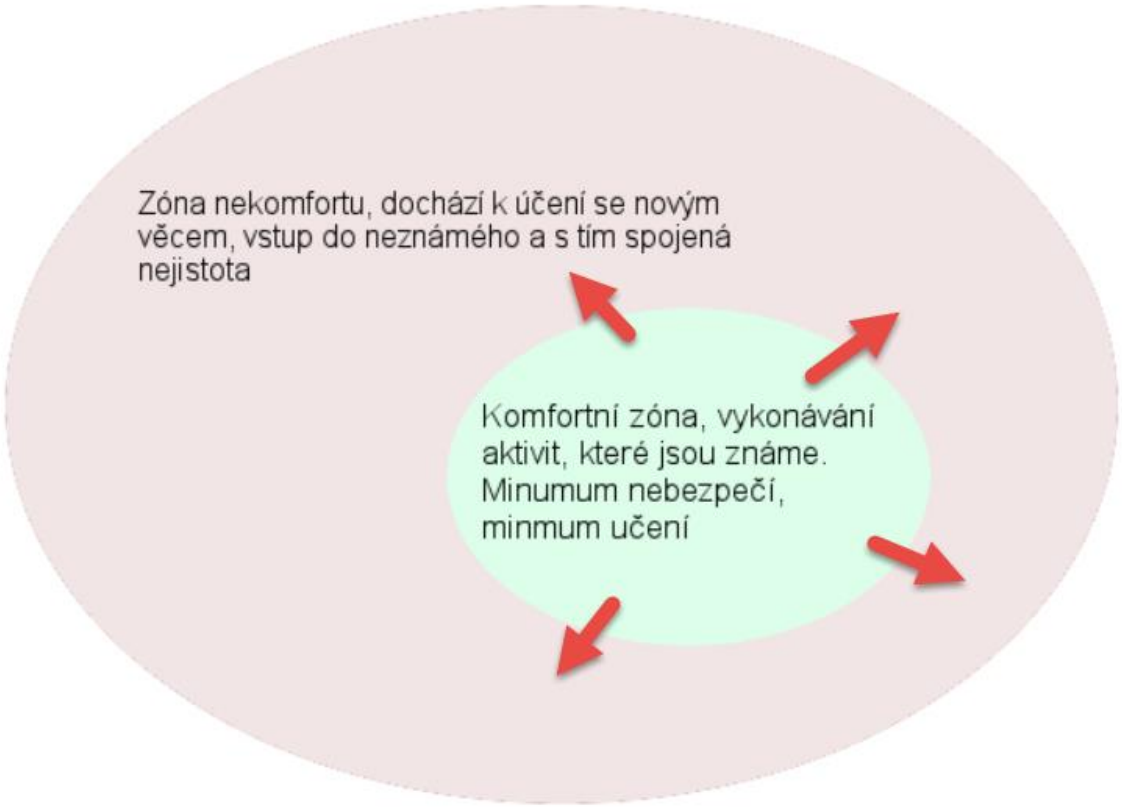
V případě, že dochází k odmítání softwaru osobami, které by s ním měly pracovat, se již jedná o problematičtější průběh. Takovéto osoby poté hledají sebemenší záminku, aby software shodily a ukázaly, že to není ta správná cesta. Pokud toto ve firmě zákazníka nastane, začíná takový boj kdo z koho. Na jedné straně je Actis společně s vedením firmy, do které je software dodáván, na straně druhé jsou pracovníci, jejíž hlavní pracovní náplní bude práce s tímto systémem.

Zde je potřeba neustále opakovat klady nového řešení a být odmítajícím osobám vstřícný a otevřený. Je důležité s těmito osobami hovořit, aby se přišlo na jádro problému s odmítáním.

S adaptací softwaru se nyní bojuje ve společnosti XY Pomůcky. Od počátku vývoje byly proti dámy, které se softwarem mají pracovat. Bylo zapotřebí najít jakousi komunikační frekvenci a kompromisy, aby změnily svůj ostře odmítavý postoj.

Zajisté tomu napomohly časté dotazy, co by tam mělo být jinak, aby s tím začaly spolupracovat a předělávání softwaru k jejich představě.

Na druhou stranu je zapotřebí zachovat nějakou hranici (dámy sice diktovaly, co se nelíbí, ale oni za software nevydávají finanční prostředky). Někdy i může být pochopitelný odmítavý postoj, například z důvodu strachu o pracovní pozici, vystoupení z komfortní zóny, změny náplně práce. Opět vše záleží na vedení společnosti, jak se poté k takovým zaměstnancům postaví.



Obrázek 12: Výstup ze zóny komfortu
Zdroj: Vlastní

8 Výběr a zhodnocení nejlepšího postupu pro hladký průběh

Asi každý zákazník, který se obrátí na společnost, která vyvíjí software na míru, očekává vyšší finanční náročnost, než kdyby software zakoupil (a nebyl s ním kvůli jeho univerzálnosti spokojen). Možná ale již ne každý, si umí uvědomit, že je to právě o něm, jeho požadavcích, jeho specifikacích, vzájemné komunikaci. Tito zákazníci poté mohou být nedočkaví, co se týče dodacích termínů. Se zákazníkem, který si uvědomuje, že se doba vývoje odvíjí od konkrétnosti a jednoznačnosti jeho požadavků, je dnes radost spolupracovat.

Tato kapitola se zaměřuje na některé tipy, doporučení, co při vývoji a implementaci nepodcenit a zároveň které věci raději před jejich využitím zvážit dvakrát.

1. Zaujmout potenciálního zákazníka

   Aby se potenciální zákazník, který uvažuje o koupi softwaru a míru, stal opravdu zákazníkem, později třeba i partnerem, je potřeba ho zaujmout. Zaujmout ho může styl vyjadřování zástupce vývojářské firmy, originalita navrhovaného řešení, komplexnost řešení.

   Určitě by zákazníka zaujala i cena, což je ale poměrně krátkodobé hledisko. Nelze něco slibovat na počátku, pokud nebyla provedena ani základní analýza. Mělo by být tedy pravidlem, že při získávání zákazníků zaznívají jen reálné a realistické možnosti řešení, ať už finanční, či technologické.
2. Snažit se odhadnout typ zákazníka

   Odhad zákazníka s sebou přináší možná úskalí, zejména pokud zástupce vývojové firmy nemá dostatečné zkušenosti. Je potřeba se již na počátku alespoň trochu seznámit s fungováním společnosti, zjištění kompetentních osob, které by později mohly mít stěžejní informace pro budoucí software. Tyto osoby je poté vhodné pozvat na některé schůzky, týkající se požadavků na software. Tím se může předejít budoucímu odmítání softwaru z jejich strany.
3. Nepodcenit počáteční analýzy

   Je možné, že jako společnost XY Pomůcky, se objeví další zákazníci, kteří se rozhodnou odmítnout technickou analýzu. Pokud jim vývojářská společnost rozhodne vyhovět, určitě je potřeba je upozornit na některá úskalí tohoto rozhodnutí. Takovým příkladem

může být to, že firma nakonec na vývoji neušetří ani čas, ani finance, může docházet k dvojímu chápání jedné záležitosti v procesu, či že je vhodné mít vše černé na bílém pro obě strany.

4. Komunikace především

   Obě strany spolu musejí komunikovat, ať už, pokud dojde k nějaké změně u zákazníka, či pokud se narazí na nějaký problém či nejasnost u vývojářů. Neřešení problémů s sebou nese jejich prohlubování, až se z toho může časem vyvinout poměrně nepříjemná záležitost pro obě strany. Druhá strana bohužel nevidí to stejné, co ta první, a proto, i

   pokud se jedná o na první pohled maličkost, je potřeba to řešit.

5. Nadále využívat agilní metodiky pro vývoj

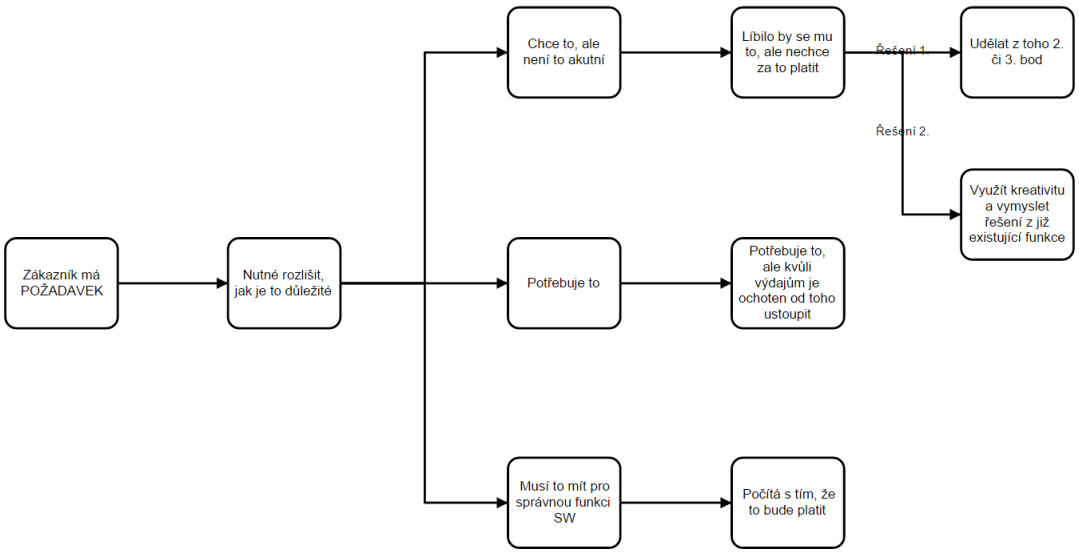
   Jen ty jsou totiž tak pružné, že dokáží obsáhnout rychle se měnící společnost a s tím spojené měnící se požadavky zákazníka. Určitě, pak již bude na zvážení, který druh agilní metodiky, či její kombinace, u kterého zákazníka realizovat.

6. Testovat a zase testovat

   Aby se předešlo snížení důvěryhodnosti u zákazníka, je potřeba jim předávat z programového hlediska bezchybný software. Nic nedokáže software shodit (především u těch osob, který ho odmítají) více, než když se při předávání objeví chybová hlášení, či bílá obrazovka.

7. Být kreativní

   Zákazník bude mít zajisté mnoho požadavků na software. Je důležité umět rozlišit, jak moc danou věc potřebuje. Pokud ji chce, ale nechce za ni platit, je vhodné využít kreativitu a zkusit vymyslet jiný případ užití z již existující funkce. Pokud danou věc chce a musí mít, pravděpodobně počítá s tím, že za to také zaplatí.



Obrázek 13: Požadavek zákazníka
Zdroj: vlastní

8. Zvážit způsob nasazení softwaru

   V závislosti na zjištěném typu zákazníka, jeho kultury a obecně na jeho připravenosti přijmout nový software, společně rozhodnout, jakým způsobem dojde k nasazení nového softwaru do společnosti. Nelze obecně rozhodnout, který způsob je nejlepší, protože každý má svoje výhody a nevýhody. Bude se to také odvíjet podle typu softwaru, velikosti firmy zákazníka, počtu vstupů a výstupů softwaru.

9. Odchytávat připomínky a nedostatky za běhu

   Je vhodné se dopředu domluvit na komunikačním kanálu, jakým způsobem bude probíhat komunikace mezi firmami a přizpůsobit se tomu. Zvolit jakým způsobem, se budou tyto připomínky zaznamenávat, kdo je za to odpovědný a na jaké místo, nejlépe přístupné všem dotyčným, budou vkládány.

10. Nesmířit se s tím, že se nic neděje

    Velmi často se může stávat, že ačkoliv nepřicházejí další připomínky, tak, že software má svoje nedostatky, jen o nich vývojáři nevědí. Poté mohou ve velké skupině dorazit jako blesk z čistého nebe. Zde je to opět o komunikaci mezi firmami.

11. Být pozitivní

    I velmi dlouhodobý proces implementace softwaru má svůj konec. Je dobré zůstat pozitivní, hlavně při komunikaci se zákazníkem a soustředit se na jeho posunutí a dokončení.

Závěr

Cílem této bakalářské práce bylo zjistit a zhodnotit, zda existují nějaké univerzální doporučení pro hladký průběh implementace a nasazení softwaru, a pokud existují, blíže je specifikovat.

Před tímto zjišťováním bylo nezbytné popsat základní fungování celého vývoje softwaru a některá úskalí a nebezpečí pro jednotlivá rozhodnutí. Dále více přiblížit problematiku implementace a adaptace softwaru do společnosti. Vše je doplněno konkrétními příklady z praxe.

I přesto, že nelze přesně využít stejné metody u všech typů zákazníka, lze se řídit některými obecnými doporučeními, které mohou pomoci hladkému průběhu celého procesu. Toto shrnutí napříč celým procesem se nachází v závěru práce.

Seznam použité literatury

BRUCKNER, Tomáš. *Tvorba informačních systémů: principy, metodiky, architektury*. 1. vyd. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-247-4153-6.

BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. 1. vyd. Praha: Grada, 2005. Management v informační společnosti. ISBN 80-247-1075-7.

COCKBURN, Alistair. *SW development as a cooperative game* [online]. In: . 1999 [cit. 2016-04-28]. Dostupné z: http://members.aol.com/humansandt/papers/asgame/asgame.html

ČSN EN 60300-3. *Management spolehlivosti.* 2009.

MALÝ, Martin. Softwarové licence: úvod pro obyčejné lidi. In: *Zdroják: o tvorbě webových stránek a aplikací* [online]. Pobočná 9, 140 00 Praha 4: Devel.cz Lab s.r.o., 2011 [cit. 2016-02-02]. Dostupné z: https://www.zdrojak.cz/clanky/softwarove-licence-uvod-pro-obycejne-lidi/

ŠMÍD, V. Životní cyklus informačního systému. In: *Http://www.fi.muni.cz/* [online]. Brno: Fakulta informatiky MU [cit. 2016-01-31]. Dostupné z: http://www.fi.muni.cz/~smid/mis-zivcyk.htm

WIEGERS, Karl Eugene. *Požadavky na software*. Vyd. 1. Brno: Computer Press, 2008, 448 s. ISBN 978-80-251-1877-1.

ŠEVČÍK, Václav. *Hlavní výhody platformy MyDSy*. Liberec, 2015.

VOTRUBCOVÁ, Barbora. *Semestrální zpráva z praxe a analýza organizace Actis, s.r.o*. Liberec, 2016. Semestrální zpráva. Technická univerzita v Liberci.

ZÁDOVÁ, Vladimíra. *Vývoj IS: Vývoj, cyklus, metodiky*. EF TUL

Seznam příloh

Příloha A Ukázka výsledku technologické analýzy

1.1.1. VÝŠE POKUTY

V případě, že je výše pokuty vyšší než povolený limit v nastavení aplikace, dojde k upozornění CA/KAM. Pokud není CA/KAM zadán, notifikace neodejde, stavem WF pouze projde.

1.1.2. OVĚŘENÍ ZPŮSOBU ŘEŠENÍ POKUTY

Ze systému NOLS jsou obdrženy následující hodnoty:

- P - Přefakturovat
- N - Nepřefakturovat
- Jiná – pak se vyvolá stav pro doplnění NOLS

1.1.3. DOPLNĚNÍ INFORMACÍ KLIENTEM

Formulář bude obsahovat nastavitelný text, logo společnosti leaseplan, název, pole pro vložení informací o řidičovi. Uživatel se nebude přihlašovat. Odkaz má platnost pouze po dobu, kdy je proces ve stavu Doplnění informací klientem. vzhled bude dle grafického manuálu LPCZ, Uživatel může kdykoli informace doplnit jak přes vygenerovaný odkaz (klient alespoň uvidí rekapitulaci).

Volba:

P - přefakturace (pouze když lze)

N – klient řeší sám (tj. LPCZ neplatí a platí např. řidič)

- Jméno*
- Příjmení*
- Datum narození
- Adresa
  - Ulice
  - č.popisné/č.orientační – jedno pole dohromady
  - PSČ
  - Město
- Zaškrtávátko – „Pokutu vyřešíme sami"
- Tlačítko potvrdit – otevře zrekapitulované údaje
  - Přidat rekapitulaci údajů a ještě jednou potvrzení (vyplní, systém ukáže, co vyplnil a klient dá odeslat)
  - Tlačítko pro odeslání

1.1.4. VOZIDLO JE V MAJETKU LPCZ, ALE NEBYL IDENTIFIKOVÁN KLIENT

Systém automaticky vygeneruje PDF s přednastaveným textem:

TEXT NA ÚRAD – texty nejsou součástí analýzy LPCZ dodá.

1.1.5. MANUÁLNÍ DOŘEŠENÍ

Uživatel má možnost v případě manuálního dořešení zvolit přednastavenou šablonu pro vygenerování PDF, případně PDF přiložit do WEB a nechat odeslat odpovědí zpět úřadu nebo si vygenerovat ze systému Plnou moc (ve formě PDF), která zůstane u pokuty přiložena – lze přikládat libovolné přílohy.

1.2. LIBOVOLNÉ PŘÍLOHY, KOMENTÁŘE

Umožnit přiložení libovolné přílohy a komentáře. Přílohy lze odstraňovat (nebudou vidět, v systému zůstanou), komentáře odstraňovat ani měnit nelze.

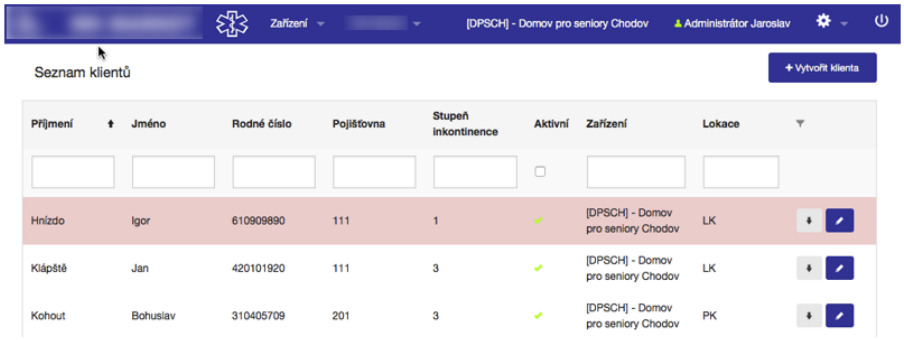
1.3. KONTROLA DAT

Každý výstup ze systému bude doplněn o kontrolu dat uživatelem, před odesláním výstupu na příslušný úřad nebo k fakturaci. Kontrola dat bude postupně odebírána s tím, jak se bude zlepšovat automatizace zpracování pokut.

1.4. NOTIFIKACE V MYDSY

Systém podporuje nastavitelné znění notifikací. Avšak, pokud má notifikace obsahovat

Příloha B Ukázka uživatelského manuálu



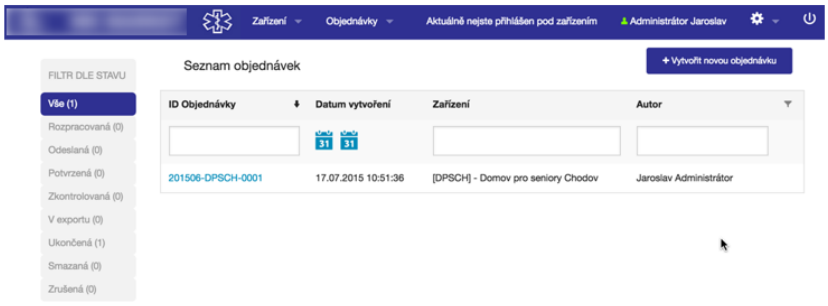
Výběr produktů **není** omezen dle stupně inkontinence (lze vybírat i produtky na přímou platbu, atd.). Dále je u klienta nutné zvolit lékaře, který je exportován do dávky pro pojišťovnu. Objednávka hlídá vyplnění lékaře u klienta až v dalších krocích procesu, je tedy lepší lékaře vždy vyplnit rovnou s klientem. Číslo lékaře je uvedené v jeho razítku na zadní straně poukazu:

Pokud má zařízení lokace, všichni klienti musí patřit k nějaké lokaci. Pokud došlo k zadání klientů již dříve, je nutné je všechny znovu editovat a vybrat jim příslušnou lokaci.

3.4 Přidání klientů k zařízení

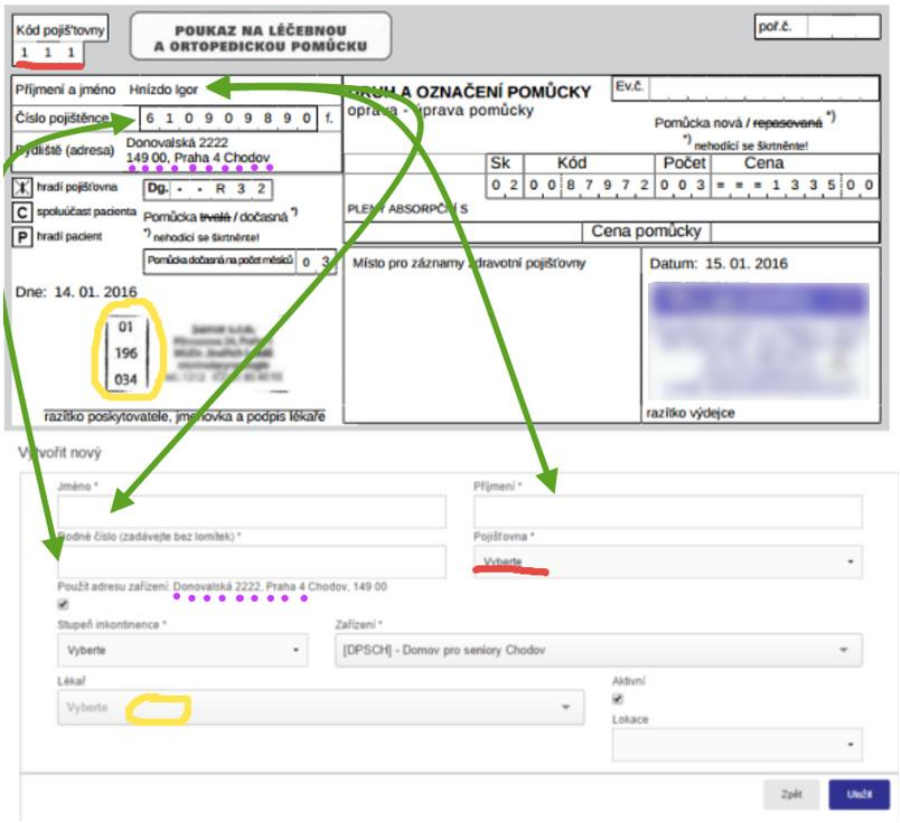
Bez klientů není možné vytvářet objednávku. Klient musí mít zadanou alespoň jednu šablonu poukazu, aby mohl objednávkou projít.

Začnou zadávat klienty (uživatel s vyššími právy může zadávat klient za zařízení, stačí, když klikne v hlavičce na "Aktuálně nejste přihlášen pod Zařízením) a zvolí jiné):



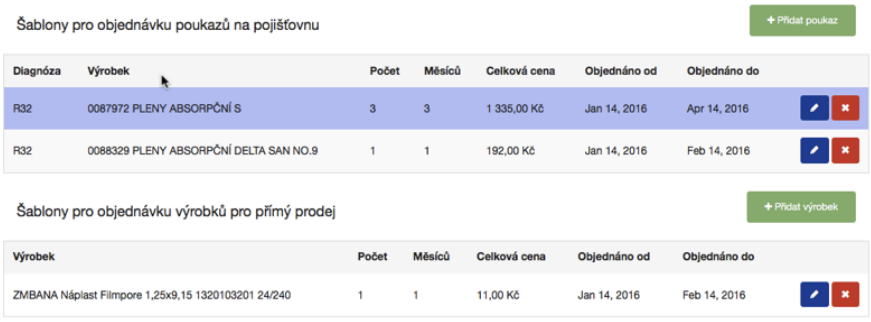
Klienty začne zadávat pod Zařízení/Klienti, pomocí Vytvořit klienta. Klient bez zadaných produktů je zobrazen červeně:

3.5 Přidání klientů k zařízení z poukazu



3.6 Přidání poukazů a šablon výrobků na klienta

Klient musí mít vytvořen alespoň jeden poukaz, aby jej bylo možné vložit do objednávky.



Historie objednávek pro klienta – zobrazit/skrýt

Příloha C Složitost procesů ve společnosti Leasing

Příloha C se vzhledem ke své velikosti nachází pouze na přiloženém CD.