Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Analýza procesů ve firmě se zaměřením na používání systému pro správu zákaznických požadavků

Plzeň 2019 Lenka Rychtářová

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 15. května 2019

Lenka Rychtářová

Poděkování

Na tomto místě bych ráda poděkovala vedoucímu práce Ing. Petru Přibylovi za vstřícnost a ochotu. Dále děkuji panu docentu Přemku Bradovi za cenné připomínky a odborné rady.

Veliké dík také patří mé rodině a dalším blízkým osobám, kteří mě ve studiu podporovali.

Lenka Rychtářová

Abstract

Content of the Diploma Thesis is based on different attitudes and methods of software engineering and tasks engineering. Main task of publication was to get a knowledge of methodology of specific company, to identify problems connected with workflows and propose changes and improvements of processes or appropriate tools.

Solution of discovered problems is mainly represented by creation of a methodic – knowledge group and implementation of Atlassian tool – JIRA. Purpose of this group is to standardize methodology management and software process improvement of chosen company. After consultation with key users, it was decided, that implementation of JIRA tool would not be appropriate for service projects. For development projects, the company sees benefit of usage of proposed tool.

Abstrakt

Diplomová práce se zabývá přístupy a metodikami softwarového inženýrství a nástroji určenými ke správě úloh. Hlavním cílem práce bylo seznámit se s metodikou vybrané společnosti, identifikovat problémy spojené s procesy a navrhnout změny procesů či nástrojů.

Řešením identifikovaných problémů je zejména vytvoření metodické znalostní skupiny a použití nástroje JIRA Software. Účelem této skupiny je standardizovaná správa metodik a řízené zlepšování procesů dané společnosti. V rámci ověření s klíčovými uživateli bylo stanoveno, že nástroj JIRA Software není v současné chvíli vhodné využít vzhledem ke specifikům na servisní projekty. U vývojových projektů bude nástroj přínosem.

Obsah

1 Úvod

Za účelem zvýšení úspěšnosti softwarových projektů jsou definovány přístupy a metodiky, jak k vývoji software přistoupit. Přístupů k vývoji v současnosti existuje nespočet. Nelze tak jednoznačně definovat ideální přístup. Základem všech by měly být aktivity jako je specifikace softwaru, návrh a implementace, validace a evoluce. Mnoho společností si vytvořilo procesy vlastní. Je důležité využít schopnost pracovníků společnosti a konkrétních specifik projektů.

Klíčovým pojmem v aktivitě specifikace softwaru jsou požadavky. Požadavky popisují, jaké služby software poskytuje a jaká jsou omezení pro jeho činnosti. Proces zjišťování, analýzy, dokumentace a správy spadá pod disciplínu inženýrství požadavků *(Requirements Engineering)*. Úloha *(Task)* je jednotka práce, která má být vykonána. Úloha může být typu požadavek, problém, úkol a další. Spravovat úlohy lze velmi jednoduše. V případě rozsáhlých projektů je možnost využít sofistikované, automatizované nástroje, které práci usnadní a zefektivní.

Vybraná společnost, jejíž název není z důvodů ochrany obchodních tajemství zveřejněn, pro správu hlášení (úloh) využívá vlastní nástroj ISZA. Nástroj vznikl na míru potřebám společnosti.

Cílem je seznámit se s metodikou vybrané společnosti a identifikovat slabá místa procesů. Na základě kritické analýzy nabytých poznatků jsou poté navržena řešení identifikovaných problémů. Nedílnou součástí je ověření řešení s klíčovými uživateli.

Úvodní kapitola popisuje přístupy softwarového inženýrství. Přístupy jsou spíše abstrakcí, na které je demonstrováno, jak se k vývoji softwaru přistupuje. Jsou shrnuty základní principy a výhody a nevýhody vodopádového, iterativního, inkrementálního, prototypového modelu a agilních přístupů. Metodika je chápána jako "model životního cyklu" či "vývojový proces". V druhé kapitole jsou popsány metodiky RUP a SCRUM. Pátá kapitola se zabývá inženýrstvím úloh, inženýrstvím požadavků a nástroji pro správu úloh. V praktické části práce je popsána metodika společnosti a nástroj ISZA. Šestá a osmá kapitola popisují identifikované problémy a návrhy řešení.

2 Přístupy softwarového inženýrství

Softwarové inženýrství je technická disciplína zabývající se aspekty produkce softwaru od počáteční fáze specifikace systému až po jeho údržbu [19]. Pojem byl zaveden již v roce 1967 vědeckým výborem NATO složeným z vědců zastupujících různé členské státy v oblasti počítačové vědy [13]. Výroba software doposud nebyla založena na teoretických základech a praktických disciplínách. Uskutečnila se konference, která měla za cíl osvětlit problémy softwarového inženýrství a diskutovat techniky a metody vývoje softwaru.

Některými členy konference byla situace ve vývoji software označována jako "softwarová krize"[13]. Neúměrně se zvětšila diference mezi ambicemi a úspěchy, která se objevila v několika dimenzích: mezi sliby uživatelům a výkonem software, mezi tím, co je možné a čeho lze v daném okamžiku dosáhnout, a taktéž mezi odhady nákladů a skutečnými výdaji. Alarmující byla skutečnost, že na moderních systémech závisí život nejen jednotlivce, ale i stovek lidí, jedná-li se například o systémy veřejné infrastruktury.

Přístupů k vývoji software v současnosti existuje nespočet. Základem všech by dle [19] měly být následující aktivity:

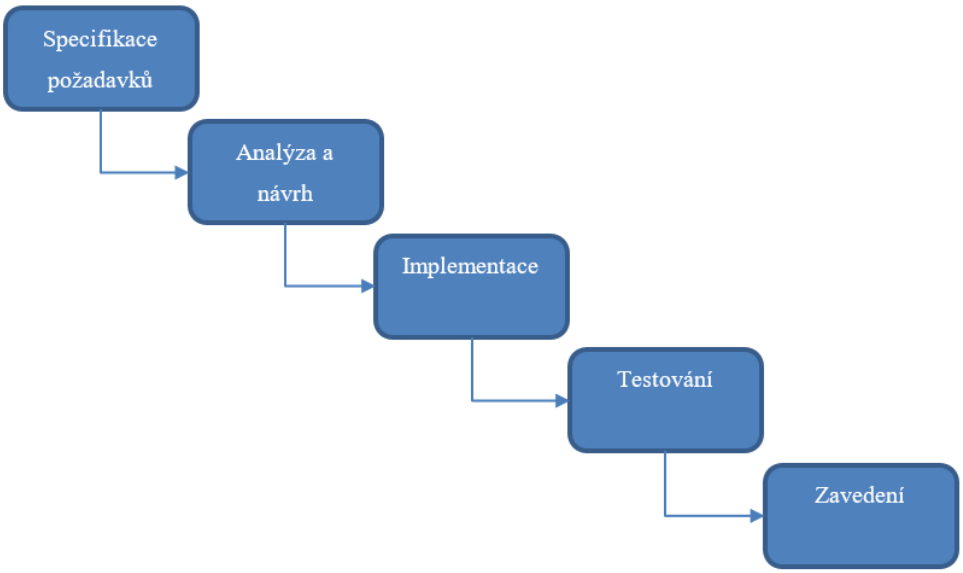
- specifikace softwaru,
- návrh a implementace softwaru,
- validace softwaru,
- evoluce softwaru.

Přístupy usnadňují vývoj softwaru a pomáhají celý proces vývoje pochopit [19]. Přístupy popsané v následujících podkapitolách nejsou popisem celého procesu, ale přístup pouze demonstrují. Každý model má své výhody i nevýhody. Nelze tak jednoznačně definovat ideální model. Organizace musí zohlednit schopnost pracovníků a konkrétní specifika projektů.

2.1 Vodopádový model

Vodopádový model je model postupného vývoje, který byl popularizovaný v letech 1970 [19]. Jedná se o nejstarší model softwarového inženýrství, široce používaný ve státních zakázkách. Na obr. č. 2.1 jsou znázorněny jednotlivé

fáze modelu, kdy výstup jedné fáze je vstupem další. Autor Ian Sommerville ve své publikaci Softwarové inženýrství [19] vodopádový model rozčlenil na fáze specifikace požadavků, analýza a návrh systému, implementace, testování a zavedení.



Obrázek 2.1: Vodopádový model

Cílem činnosti specifikace požadavků je popis funkcí, které systém musí poskytovat, a zároveň omezení[21]. Specifikace je základem pro plánování, návrh i implementaci systému a jeho následné testování.

Analýzou a návrhem jsou požadavky zpracovány tak, že se definuje celková systémová architektura. Je zahrnuta identifikace a popis základních abstrakcí a jejich vztahů.

Během fáze implementace se realizuje návrh systému na základě dokumentace z předchozích činností [19]. Kompletní systém se v další fázi důkladně otestuje. Musí být zajištěno, že byly všechny požadavky implementovány a pokryty testy. Po otestování se systém předá zákazníkovi.

Výstupem každé fáze je jeden či více schválených dokumentů [19]. Následující fáze by měla začít dříve, než předchozí končí. V praxi se tyto fáze často překrývají, protože proces nelze popsat jednoduchým lineárním modelem.

2.1.1 Výhody vodopádového modelu

Hlavní výhodou vodopádového modelu je jeho jednoduchost.

- Jednoduchý systematický lineární model.

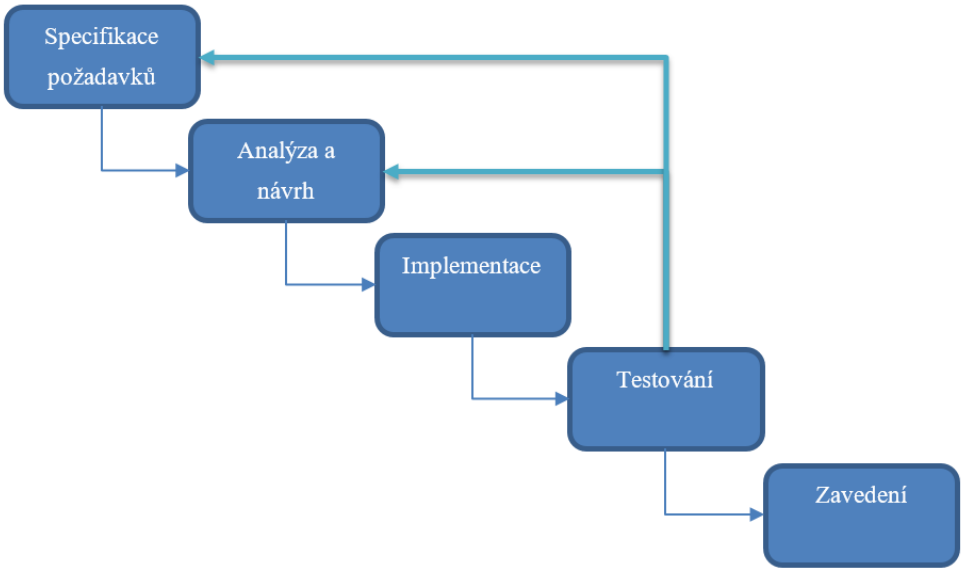
- Důkladná dokumentace.

2.1.2 Nevýhody vodopádového modelu

Mezi hlavní nevýhody patří nemožnost reagovat na změnu po skončení fáze [16].

- Je složité na začátku projektu definovat všechny požadavky.
- Model není přizpůsobený na změny v požadavcích.
- Systém je uživatelem viděn až v poslední fázi cyklu.
- Kvůli jednoduchosti není vhodný pro velké projekty.
- Chybí analýza rizik.
- Pokud v některé fázi nastane chyba, nelze dodat kvalitní software.

Implementace je velmi riskantní [15]. Problém je znázorněn na obr. č. 2.2. Testovací fáze je až na konci vývojového cyklu. Při testování nastávají události, které nelze přesně zanalyzovat. Jedná se například o časování, ukládání, přenos vstupů / výstupů apod. Pokud tyto jevy nesplňují omezení, je nutné provést zásadní změny. Musí být upraveny požadavky nebo změněn návrh. Na základě skutečnosti, že se proces vývoje vrátil na začátek, lze očekávat až desetiprocentní překročení plánů i nákladů.



Obrázek 2.2: Vodopádový model - změna

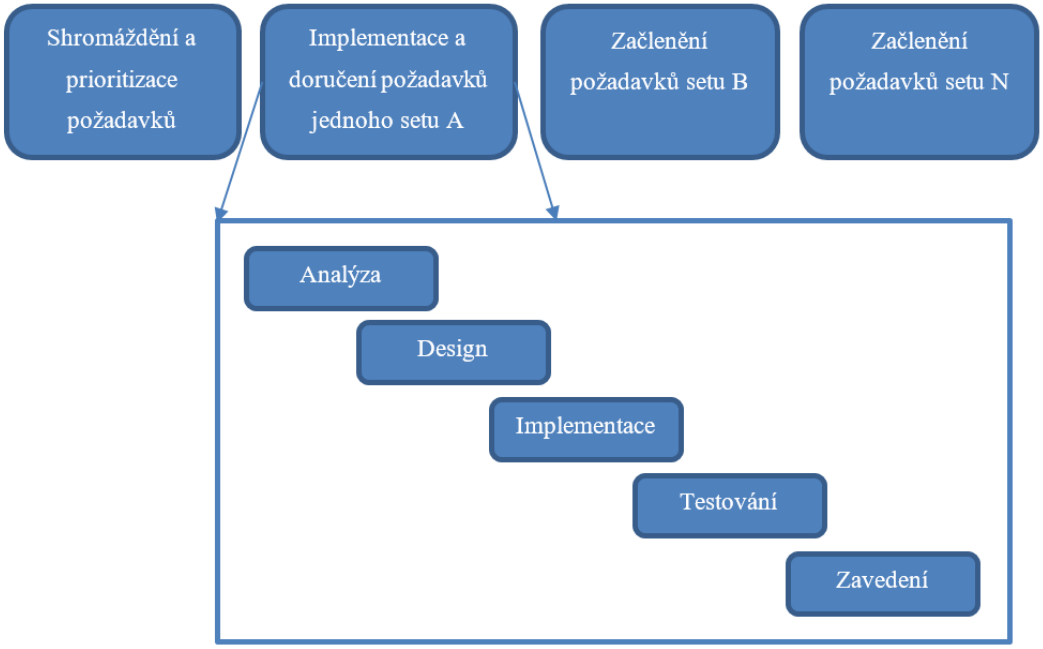
Vodopádový model se kvůli zmíněným nedostatkům v praxi příliš nepoužívá. Zejména nelze aplikovat na rozsáhlé projekty. Z původního modelu v současné době vychází mnoho modifikací, například V-Model, popsaný v článku [8]. Při použití V-Modelu se již počítá s případnými změnami po fázi implementace. Realizace pomocí V-Modelu je ale nákladná a časově náročná.

2.2 Iterativní model

Iterativní přístup využívá tzv. iterací. Vývoj softwaru se skládá z několika iterací v sekvenci. Každá iterace je malý projekt, který může být složen z činností, jako je analýza požadavků, návrh, programování a testování [11]. Cílem je po ukončení iterace vydat stabilní, integrovaný a otestovaný částečný software. Iterace může být teoreticky pouze ladění výkonu či optimalizace zdrojového kódu. Nejčastěji se ale iterace využívají za účelem přírůstků softwarů. Tento koncept se nazývá IID (Iterative-Incremental Development). Přírůstkový přístup jeho výhody a nevýhody jsou popsány v následující podkapitole.

2.3 Inkrementální model

Inkrementální model dokáže velmi rychle integrovat změněné či nové požadavky [17]. Systém je zákazníkovi dodáván postupně se zvyšující funkcionalitou. Jednotlivé požadavky se prioritizují a implementují po skupinách. Na obrázku č. 2.3 je model zobrazen.



Obrázek 2.3: Inkrementální model

První dodávka systému pokrývá omezené množství požadavků "A", které jsou zásadní pro celý projekt. V další dodávce jsou mimo požadavky setu "A" implementovány požadavky dalšího setu "B". V každé následující dodávce jsou poté začleněny požadavky dalšího setu.

2.3.1 Výhody inkrementální modelu

- Systém je dodáván po částech, celkové náklady jsou distribuovány [17].
- Není potřeba vytvářet velký tým, protože práce je dodávaná po částech.
- Chyby nalezené v dodávce lze opravit během další dodávky.
- Uživatel vidí systém v raných fázích projektu. Lze rychle reagovat na zpětnou vazbu uživatele. Riziko nepřijetí projektu je distribuováno.
- Funkčnost je inkrementována, tudíž testování není tak náročné.

2.3.2 Nevýhody inkrementální modelu

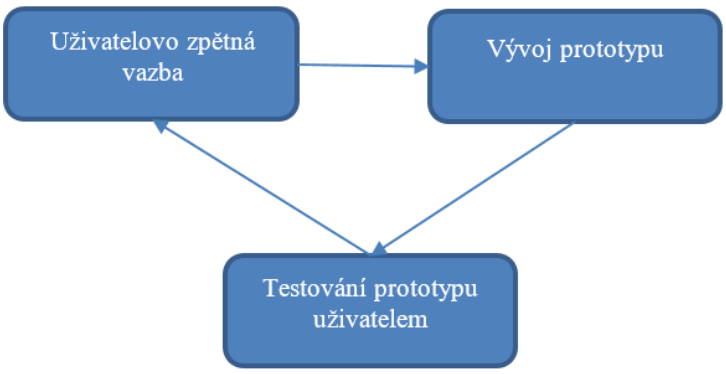
- Náklady na vývoj jsou vysoké kvůli dodávce systému po částech [17].

- Pro připojení modulů vyvinutých s každou fází je nezbytné důkladně popsat rozhraní.
- Model vyžaduje náročné plánování k distribuci práce.
- Testování jednotlivých modulů vede ke zvýšeným nákladům.

Inkrementální model je využíván, jsou-li požadavky definovány na začátku projektu. Současně se očekává, že se požadavky budou v čase měnit. Lze jej aplikovat také na projekty trvající déle než rok či je-li vyžadováno dodávat systém v intervalech.

2.4 Prototypový model

V konceptu prototypování jsou vyvíjeny neúplné verze systému tzv. "prototypy", které jsou průběžně představovány uživateli [17]. Uživatel má tak možnost vidět systém na začátku cyklu. Nejprve je vytvořen papírový model, který je diskutován s uživatelem. Na základě zpětné vazby je poté vytvořen další prototyp. Proces pokračuje až do fáze, kdy je uživateli představen prototyp, který je uspokojující. Na obr. č. 2.4 je proces prototypování znázorněn.



Obrázek 2.4: Prototypový model

Existuje několik přístupů prototypování. Například *Rapid Throwaway Prototyping*či *Evolutionary Prototyping* [17]. U přístupu *Rapid Throwaway Prototyping* je systém vyvíjen bez hluboké znalosti systému. Rychle vytvořené, nedokonalé prototypy jsou ověřovány s uživateli. Pokud nesplňují zásadní omezení a požadavky, jsou prototypy zahozeny. Naopak přístup *Evolutionary Prototyping* je založen na znalosti požadavků. Proces je ukončen, pokud prototyp splňuje požadavky a omezení.

2.4.1 Výhody prototypového modelu

Hlavní výhodou modelu je, že je systém viděn uživatelem v raných fázích cyklu [17].

- Rychlá reakce na připomínky, požadavky.
- Nové požadavky lze snadno integrovat.
- Požadavky jsou zřejmé díky prototypům.
- Uživatelem je do projektu zahrnut již od začátku projektu. Cítí se tak bezpečněji, a pohodlněji.

2.4.2 Nevýhody prototypového modelu

Uživatel po brzkém představení systému nabude dojem, že systém bude doručen brzy [17].

- Uživatel nemusí znát rozdíl mezi prototypem a plně vyvinutým systémem.
- Rychlé tvoření prototypů inklinuje k vytvoření sub-optimálního řešení.
- Proces může být zdlouhavý.
- Uživatel může ztratit zájem na projektu, není-li na začátku dodán kvalitní prototyp.
- Nekvalitní dokumentace.

Prototypový model je vhodné použít, pokud nejsou plně specifikovány požadavky na systém. Pomocí prototypů tak lze pochopit zákazníkovy potřeby. Zároveň lze využít u projektů, kde se požadavky rychle mění.

2.5 Agilní přístupy

Software je žádoucí vyvíjet rychle, a to nejen kvůli proměnlivému IT prostředí, ale také kvůli konkurenceschopnosti [19]. Často nelze definovat úplnou sadu požadavků na software. Požadavky se v čase mění a mnohdy jsou klíčové funkčnosti vymezeny až po nasazení první verze softwaru. Modely procesů vývoje softwaru například vodopád, kde je kompletní specifikace v raných fázích vývoje, nejsou pro rychlý vývoj optimální.

Agilní přístupy prosazené koncem 90. let jsou navrženy tak, aby byl software vyvinut pohotově. Přístupy patří mezi metody inkrementálního vývoje popsaného v kapitole 2.3. Inkrementy jsou obvykle dodávány během dvou až tří týdnů. Zákazníci jsou často nedílnou součástí celého týmu, aby byla možná zpětná vazba na variabilní požadavky.

Základní principy a ideje agilních přístupů jsou shrnuty v agilním manifestu [7]:

„Objevujeme lepší způsoby vývoje software tím, že jej tvoříme a pomáháme při jeho tvorbě ostatním. Při této práci jsme dospěli k těmto hodnotám:

- ***Jednotlivci a interakce** před procesy a nástroji.*
- ***Fungující software** před vyčerpávající dokumentací.*
- ***Spolupráce se zákazníkem** před vyjednáváním o smlouvě.*
- ***Reagování na změny** před dodržováním plánu.*
- *Jakkoliv jsou body napravo hodnotné, zvýrazněných bodů nalevo si ceníme více.“*

2.5.1 Výhody agilních přístupů

Jednou z hlavních výhod agilních přístupů je rychlá reakce na změny, která dokáže udržovat konkurenceschopnost [9].

- Vysoká rychlost vývoje softwaru.
- Integrace zákazníka do vývoje, snížení chybovosti.
- Kvalitní zpětná vazba od zákazníka i členů týmu.
- Flexibilita při zpracovávání změn.

2.5.2 Nevýhody agilních přístupů

Principy agilních metodik se často v praxi realizují obtížně [9].

- Zákazník nechce být zapojen do procesu vývoje.
- Členové týmu nemusí mít vhodné vlastnosti na intenzivní zapojení v týmu.
- Obtížné stanovování priorit požadavků se všemi zainteresovanými osobami.

- Obtížná restrukturalizace společnosti například při přechodu z vodopádového přístupu na agilní.

Agilní přístupy je vhodné využít ve společnostech, které vyvíjí méně rozsáhlé systémy. Je zaručená vysoká konkurenceschopnost. Dále je vhodné aplikovat přístupy u zákazníků, kteří si plně uvědomují potřebu zapojit se do vývoje softwaru. Mezi nejznámější metodiky agilních přístupů se řadí například Scrum nebo extrémní programování. Extrémní programování je vhodnější pro týmy, které jsou vyspělé, jsou vhodně složené a mají velmi dobrou schopnost plánování. Metodika Scrum je vhodnější pro týmy, které potřebují pevnější vedení.

3 Metodiky

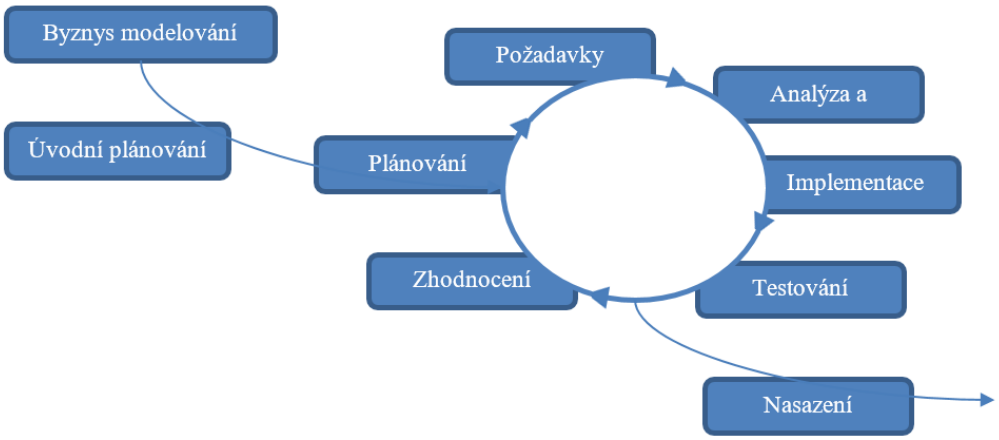
Pro účely této práce je pojem metodika chápána jako "model životního cyklu" nebo "vývojový proces" [9]. Pojem je vnímán jako souhrn postupů a návodů pro vývoj softwarové aplikace. V metodice jsou definované jednotlivé fáze životního cyklu. Metodika hledá odpovědi na otázky proč?, kdo?, kdy?, a co? Metodika do hloubky nepopisuje, jak proces provést. Pro konkrétní postup, který vede k vyřešení dílčího problému, je zaveden pojem metoda.

V následujících kapitolách jsou popsány metodiky RUP a SCRUM. Tyto metodiky byly vybrány jako relevantní k praktické části této práce.

3.1 RUP

RUP (Rational Unified Process) je iterativní metodika, která je složena z mnoha osvědčených postupů shromažďovaných experty po mnoho let. Metodika byla vyvinuta společností Rational Software, jejíž hlavní činností bylo poskytnout nástroje k rozšíření používání postupů softwarového inženýrství. RUP obsahuje komplexní popis procesů, příkladů a šablon [14].

V metodice je využit iterativní přístup, jež je složen z iterací čili opakování. Každá iterace obsahuje disciplíny, jako je sběr požadavků, analýza, návrh či implementace (viz obr. č. 3.1). Výstupem každé iterace je částečná realizace finálního systému. Každá úspěšná iterace staví na předchozí a systém se postupně vyvíjí a ladí, dokud není považován za hotový. Počáteční iterace kladou důraz na sběr požadavků, analýzu a design, kdežto pozdější na implementaci a testování.



Obrázek 3.1: Iterativní vývoj metodiky RUP

Výhodou je rychlá reakce na měnící se nebo nové požadavky, jelikož iterace netrvá déle než pár týdnů [14]. Změny jsou začleněny do následujících běhů. Zákazníkovi je systém průběžně dodáván, eliminuje se tím odhalení rizik v pozdějších fázích projektu. Mimo jiné se členové týmu každou iterací zdokonalují a dokážou se rychle poučit z vlastních chyb. Zatímco u vodopádového modelu je na každou fázi pouze jeden pokus.

3.1.1 Klíčové pojmy

Pracovníci *(Workers)*

Pracovník odpovídá na otázku **kdo** [9]. Pracovník definuje chování a odpovědnosti jedince nebo skupiny. Chování pracovníka je popsáno pomocí činností *(activities)*. Odpovědnosti jsou obvykle definovány ve vztahu k meziproduktům *(artifacts)*, které pracovník vytváří, modifikuje či kontroluje. Příkladem pracovníka je systémový analytik, designer, vývojář a další.

Činnosti *(Activities)*

Činnost odpovídá na otázku **jak** [9]. Činnost je jednotkou práce, kterou provádí jednotlivec či skupina. Výstupem je smysluplný výsledek v kontextu projektu. Činnost má jasně definovaný účel, vstupní i výstupní meziprodukty. Činnosti se dále dělí do kroků. Obecné dělení kroků spočívá ve stanovení tří základních kategorií: úvaha *(Thinking steps)*, provádění *(Performing steps)* a přezkoumání *(Reviewing steps)*. Konkrétní činnost má vždy stanovenou posloupnost kroků, které vedou k její provedení.

Meziprodukty *(Artifacts)*

Meziprodukt odpovídá na otázku **co** [9]. Meziprodukty jsou výstupy projektu. Slouží pracovníkům jako vstupy pro činnosti a zároveň jsou výstupy. Za vytvoření a správnost meziproduktu je odpovědný přiřazený pracovník. Meziprodukty můžou být ve formě modelu (např. model případu užití), elementu (např. případ užití), dokumentu (např. specifikace), zdrojového kódu nebo spustitelné aplikace. V rámci RUP je definováno velké množství meziproduktů, ke kterým je poskytnuta šablona. Verze 2002 se dělí na skupiny meziproduktů:

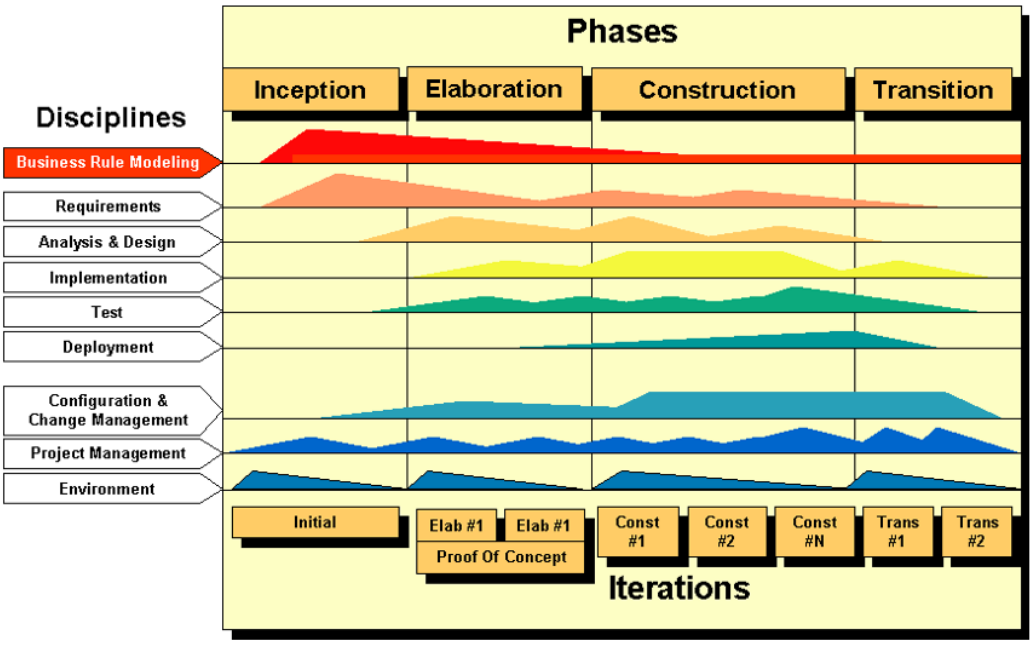
- meziprodukty týkající se požadavků *(Requirements Artifact Set)*,
- meziprodukty týkající se analýzy a designu *(Analysis and Design Artifact Set)*,
- meziprodukty týkající se implementace *(Implementation Artifact Set)*,
- meziprodukty týkající se testování *(Test Artifact Set)*,
- meziprodukty týkající se nasazení produktu *(Deployment Artifact Set)*,
- meziprodukty týkající se konfiguračního a změnové řízení *(Configuration and Change Management Artifact Set)*,
- meziprodukty týkající se projektového řízení *(Project Management Artifact Set)*,
- meziprodukty týkající se vývojového prostředí *(Environment Artifact Set)*,
- meziprodukty týkající se obchodního modelování *(Business Modelling Artifact Set)*.

Meziprodukty se typicky definují v závislosti na konkrétních specifikách projektu.

Pracovní procesy *(Workflows)*

Pracovní procesy odpovídají na otázky **kdy** [9]. Pracovními procesy jsou definovány posloupnosti činností a interakce mezi pracovníky. Pracovní procesy jsou modelovány například jako sekvenční digramy *(sequence diagram)*, diagramy spolupráce *(collaboration diagram)* nebo jako diagramy činností *(activity diagram)*. Důležité je zdůraznit, že diagramem nelze znázornit všechny závislosti, proto nelze diagramy interpretovat doslovně.

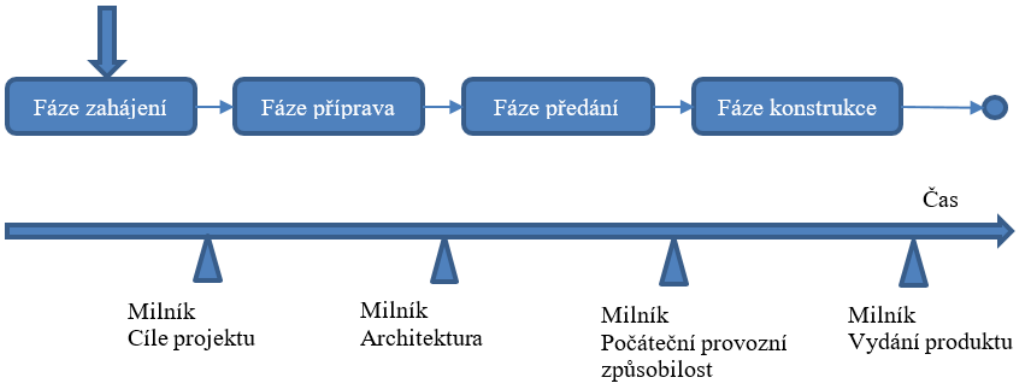
Obrázek číslo 3.2 znázorňuje celkovou architekturu RUP. Proces je rozdělen do dvou dimenzí: dynamické a statické struktury [14].



Obrázek 3.2: Dimenze metodiky RUP, [14]

Horizontální dimenze představuje dynamickou strukturu, časovou dimenzi procesu [14]. Zobrazuje, jak se proces vyvíjí během celého životního cyklu. Vertikální znázorňuje statickou strukturu procesu, která popisuje, jak jsou jednotlivé elementy procesů logicky uspořádány do hlavních disciplín.

Dynamická struktura člení projekt do čtyř fází: zahájení, příprava, konstrukce a předání [14]. Každá fáze zahrnuje jednu či více iterací, které vytvářejí technické výstupy nezbytné k dosažení cílů fáze. Mezi každou ze čtyř fází jsou definované milníky (viz obr. č. 3.3). Milníky slouží k revizi, zda byly cíle fáze naplněny.



Obrázek 3.3: Milníky metodiky RUP

3.1.2 Fáze zahájení *(Inception phase)*

Ve fázi zahájení je kladen důraz na pochopení jednotlivých požadavků a systému jako celku [14], [9]. Hlavním cílem je dosažení shody ohledně cílů a životního cyklu projektu se všemi zainteresovanými osobami. Na základě zřejmých požadavků je poté vytvořen rozsah projektu. Je eliminována řada obchodních rizik a je vytvořen obchodní případ. V této fázi je zřejmé, jedná-li se o lukrativní projekt či nikoliv.

3.1.3 Fáze příprava *(Elaboration phase)*

Ve fázi příprava je vykonána řada technických úkolů jako je design a výchozí spustitelná architektura systému. Provádí se plánování nezbytných činností a specifikace vlastností, které bezprostředně ovlivňují architekturu. Architektura zahrnuje subsystémy, jejich rozhraní, klíčové komponenty a mechanismy, jež se vypořádávají s mezi-procesní komunikací či perzistencí. Ve fázi jsou řešena technická a bezpečnostní rizika implementace. Dále je provedena částečná analýza rizik.

3.1.4 Fáze kontrukce *(Construction phase)*

Ve fázi konstrukce je systém vyvíjen ze spustitelné architektury. Ve fázi se kompletuje analýza a design. V rámci konstrukce je možné dodávat prototypy, probíhá průběžné testování a vyhodnocování kvality produktu stejně tak jako plnění požadavků. Nasazením několika interních verzí je zaručeno, že systém bude použitelný a vyhovující potřebám uživatelů. Nedílnou součástí fáze je rozhodnutí, zda-li je software v dostatečné kvalitě, otestovaný a

připravený k předání zákazníkovi.

3.1.5 Fáze předání *(Transition phase)*

Ve fázi předání je software předán zákazníkovi. Předání zahrnuje dodání softwaru, školení uživatelů a podporu a údržbu až do okamžiku, kdy je uživatel spokojen. Je zrevidováno, zda software reaguje na potřeby uživatelů. Produkt je otestován a na základě zpětné vazby uživatelů jsou provedeny drobné úpravy. Zpětná vazba uživatele je hlavně zaměřena na odstranění drobných problémů s používáním softwaru, s konfigurací či s instalací. Zásadní problémy jsou odchyceny a zpracovány již v dřívějších fázích životního cyklu.

3.1.6 Iterace uvnitř fází

Dekompozice projektu neprobíhá pouze rozdělením na fáze, ale také na iterace uvnitř fází [9]. Počet iterací v rámci fází se liší dle specifik projektů. Každá iterace v sobě zahrnuje sekvenci aktivit jako ve vodopádovém modelu. Přesná doba trvání iterace není stanovena, ideálně by se měla pohybovat v rozmezí dvou až šesti týdnů. Podstatným rysem metodiky je, že plán a průběh fází není stanoven na začátku projektu. Zpravidla se po skončení fáze vypracuje plán iterací fáze následující. Značný důraz je kladen na práci s riziky. Součástí každého plánu iterace je také nalezení a analýza rizik. RUP definuje činnosti jako například nalezení a identifikace rizik *(Activity: Identify and Assess Risks)* nebo meziproudukty jako například seznam rizik *(Artifact: Risk List)*. Výsledky analýz ovlivňují plánování dalších iterací.

- **Fáze zahájení** má zpravidla jednu iteraci. U větších projektů lze zařadit i druhou iteraci, ve které vzniká prototyp. Případně se podrobněji analyzuje prostředí zákazníka či se vývojáři seznamují s novými technologiemi.

- **Fáze příprava** má většinou dvě iterace. V případech, kdy je nutné začlenit do projektu nové pracovníky či se očekává více rizik, lze naplánovat tři až čtyři iterace.

- **Fáze konstrukce** by měla obsahovat alespoň dvě iterace. Větší počet iterací je vhodný, je-li nutné systém důkladně integrovat a testovat. Rozsáhlé projekty můžou mít až čtyři iterace.

- **Fáze předání** mívá obvykle dvě iterace. Typicky je nasazena betaverze a následně plná verze systému.

3.1.7 Výhody metodiky RUP

Nejsilnější stránkou metodiky RUP je obecnost, šíře a mohutnost, díky které je vhodná pro širokou škálu projektů [9]. Další výhody plynou z iterativního a inkrementálního přístupu (viz kapitola 2.3). Metodika RUP je detailně propracována, jsou zdokumentovány všechny náležitosti, které vedou k vytvoření kvalitního softwaru. Pro aktivity jsou dostupné šablony, návody a příklady.

Další výhodou je objektově orientovaný přístup k vývoji softwaru, který umožňuje lepší přiblížení k problému. Dalším pozitivem je, že výrobce průběžně pracuje na vývoji metodiky a nabízí celou řadu doplňkových nástrojů.

Je důležité neopomenout výhodu obecného povědomí o metodice RUP v softwarovém inženýrství. Metodika je velmi rozšířeně používána mezi společnostmi, jež se zabývají vývojem softwaru.

3.1.8 Nevýhody metodiky RUP

Některé výhody metodiky mohou být vnímány zároveň jako nevýhody v závislosti na konkrétních projektech. Robustnost metodiky je nevhodná pro menší projekty [9]. Dále je nutné zohlednit čas strávený nad zkoumáním a studiem metodiky před implementací. Společnosti Rational proces nasazení usnadňuje nabízením vlastního procesu implementace, i přes to může být osvojení velmi obtížné.

Metodika RUP je vhodná především pro velké společnosti se zaměřením na vývoj rozsáhlých a náročných produktů. Společnost musí být ochotna vyčlenit dostatek prostředků na studium a implementaci metodiky. Dále je vhodné implementovat RUP na více projektech, které vyžadují zdokumentované a propracované fáze životního cyklu vývoje softwaru. Vhodná je pro systémy, které je nutné dále udržovat, opravovat a aktualizovat. Díky značným možnostem lze ale metodiku přizpůsobit konkrétním požadavkům projektů či společnosti.

3.2 SCRUM Process Development

SCRUM Development Process je agilní metodika, jejímž hlavním cílem je zvýšení efektivity při vývoji softwaru. Stejně tak jako metodika RUP využívá výhod iterativního a inkrementálního přístupu. Historie metodiky sahá až do roku 1986, kdy byl představen pojem SCRUM. V roce 1995 byl pojem formalizován autory K. Schwaber a J. Sutherland. Hlavní motivací bylo vyvinout flexibilní metodiku, která se dokáže vyrovnat s měnícími se požadavky,

a která dokáže zlepšit produktivitu vývoje softwaru [9].

SCRUM vychází z objektově orientovaného přístupu. Vývojář je tedy zodpovědný za množinu objektů s přesně definovaným chováním a rozhraním. Hlavní charakteristikou jsou pevně definované časové intervaly tzv. sprinty (Sprints). Sprint představuje základní členění vývojového procesu. Metodika předpokládá, že aktivity analýza, design a vývoj jsou v rámci sprintu nepředvídatelné. Z toho důvodu nejsou plánované přesné obsahy jednotlivých sprintů. Jsou zavedeny každodenní schůzky, na kterých jsou operativně řešeny činnosti a kroky. Každodenní konfrontace s postupem vývoje poté vede k pružnějšímu a flexibilnějšímu vývoji.

3.2.1 Klíčové pojmy

Scrum tým *(Scrum Team)*

Scrum tým se skládá z produktového vlastníka, vývojářského týmu a Scrum mistra [7]. Tým je samoorganizující a multifunkční.

- **Produktový vlastník *(Product Owner)*** je zodpovědný za maximalizaci hodnoty produktu vyplývající z práce vývojového týmu. Je jedinou odpovědnou osobou za produktový backlog.
- **Vývojový tým *(Development Team)*** je složen z profesionálu, kteří v průběhu sprintu pracují na přírůstku produktu.
- **Scrum mistr *(Scrum Master)*** je zodpovědný za propagaci a podporu metodiky Scrum, tak jak je definovaná v průvodci Scrumem [7]. Pomáhá členům týmu porozumět teorii, praktikám a pravidlům metodiky. Je lídrem pro celý tým.

Scrum události *(Scrum Events)*

Události jsou ve Scrum jasně definované a pravidelné [7].

- **Sprint** je jeden z fundamentálních pojmů metodiky. Sprint je základní iterace, která má obvyklé trvání třicet dní. Sprint se skládá z plánování *(Sprint Planning)*, denních scrumů *(Daily Scrums)*, vývoje *(Development work)*, revize *(Sprint Review)* a retrospektivy *(Sprint Retrospective)*.
- **Denní scrum *(Daily Scrum)*** je patnáctiminutová schůzka realizovaná každý den. Schůzka je určená pro vývojový tým. Na schůzce jsou řečeny plány na následující den.

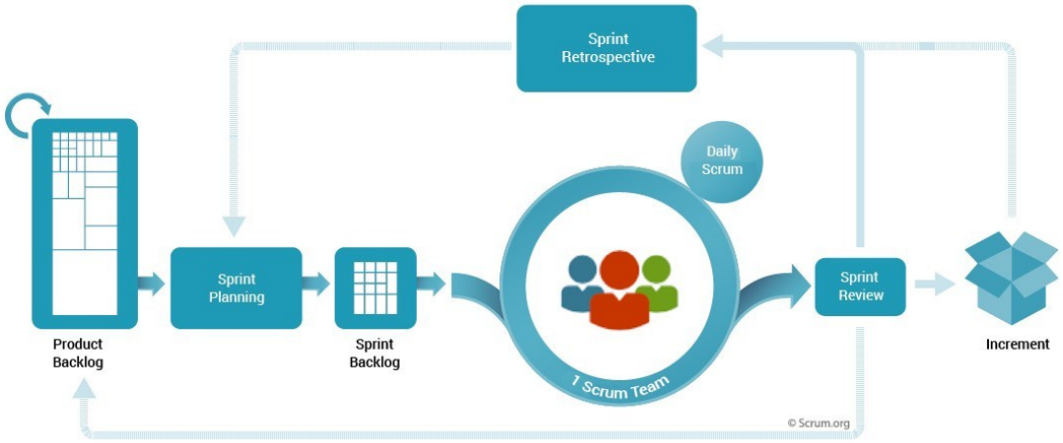
- **Revize *(Sprint Review)*** je schůzka, která se koná na konci sprintu. Účelem je zhodnotit inkrement produktu. Prezentace přírůstku má za cíl vyvolat zpětnou vazbu, případně optimalizovat další spolupráci.
- **Retrospektiva *(Sprint Retrospective)*** je příležitostí pro celý tým, aby vytvořil plán na zlepšení či optimalizaci následujícího sprintu.

Artefakty *(Scrum Artifacts)*

Artefakty představují práci [7].

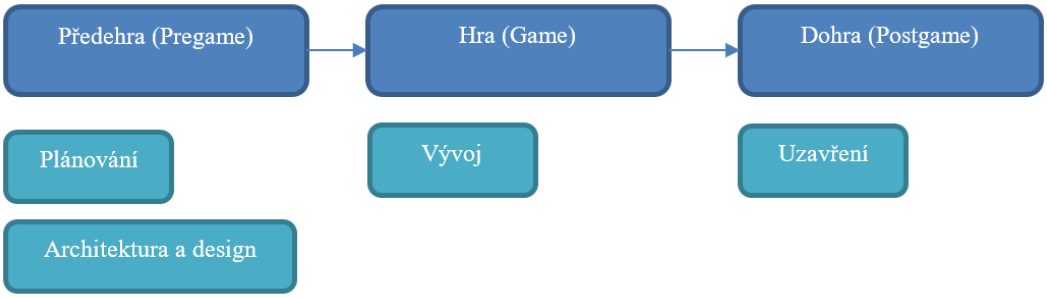
- **Produktový backlog *(Product Backlog)*** je nosič informace o funkčnosti či vlastnostech systému, které je nutné implementovat. Jedná se o uspořádaný seznam funkcí, požadavků, vylepšení či oprav, který je v čase dynamický. Modifikace a řízení backlogu je ve správě produktového vlastníka.
- **Sprint backlog *(Spint Backlog)*** je sada položek vybraných z produktového backlogu pro daný sprint.
- **Inkrement *(Increment)*** jsou realizované položky z backlogu, které byly dokončeny během sprintu.

Vývoj podle metodiky SCRUM probíhá podle schématu na obr. č. 3.4.



Obrázek 3.4: Schéma metodiky SCRUM, [18]

Fáze se sestávají z předehry, hry a dohry (obr. č. 3.5) [9]. V rámci těchto fází jsou definovány čtyři vývojové kroky, a to plánování, architektura a design, vývoj a uzavření.



Obrázek 3.5: Fáze metodiky SCRUM

3.2.2 Plánování *(Planning)*

V rámci plánování je definován rozsah aktuální verze, dále harmonogram, a další. Důležitý pojem je backlog, který definuje funkčnosti a úkoly, které se budou ve sprintu realizovat. Vytvářením backlogu fakticky začíná vývojový proces. Při plánování se mapují položky z backlogu na objekty, provádí se analýza rizik, volí se podpůrné nástroje apod.

3.2.3 Architektura a design *(Architecture / Design)*

V rámci tohoto kroku se vytváří či modifikuje architektura systému v závislosti na nových požadavcích, rizicích a poznatcích.

3.2.4 Vývoj *(Development, Sprint)*

Vývoj je jeden iterativní cyklus vývojových prací. Délka sprintu je určena v závislosti na systému, množství rizik a dalších faktorů. Obvykle bývá v délce patnácti až třiceti dní. Sprint je ukončen schůzkou Revize, na které je představena nová verze systému a jsou detekovány nové položky backlogu.

3.2.5 Uzavření *(Closure)*

Jakmile je usouzeno vedením či zákazníkem, že parametry aktuální vrze jsou dostačující, nepokračuje se dalším sprintem, ale následuje krok uzavření. Náplní fáze uzavření je příprava produktu k finálnímu předání. V rámci kroku je provedena komplexní integrace a testování. Dále se například vytváří dokumentace.

3.2.6 Výhody metodiky SCRUM

Hlavní výhodou metodiky SCRUM je schopnost pružně reagovat na změny, které vznikají v průběhu práce na projektu [9]. Každodenně jsou reflektovány případné změny a rizika z nich vyplývající. Je umožněno změnit směřování projektu, a zvýšit tak efektivitu procesu.

Dále je vývojovému týmu poskytována svoboda pro návrh optimálního řešení. Přístup lze změnit v průběhu projektu dle toho, jaké jsou aktuální požadavky zákazníka. Další výhody plynou jak z inkrementálního, tak z objektově orientovaného přístupu.

3.2.7 Nevýhody metodiky SCRUM

Nevýhodou metodiky SCRUM je především skutečnost, že se jedná spíše o souhrn doporučení, než o specifikaci konkrétních kroků [9]. Je uváděno, že metodiku SCRUM je vhodné nasadit do prostředí, kde se již podle nějaké metodiky pracuje. Z metodiky SCRUM se pouze přejme základní vedení projektu.

Další nevýhodou může být obtížný přechod na nový způsob práce. Je důležité sestavit tým z lidí, kteří jsou flexibilní, odpovědní a spolehliví.

Obecně lze říct, že SCRUM je vhodný pro malé týmy a nepříliš rozsáhlé projekty. Metodika je vhodná pro objektově orientované projekty.

4 Inženýrství úloh

Úloha *(Task)* je v kontextu práce vnímána jako jednotka práce, která má být vykonána. Úloha může být typu požadavek, úkol, problém, defekt a další.

Definice termínu požadavek není v softwarovém oboru ani v prostředích společností konzistentní. V některých případech je požadavek abstraktní popis, v jiných formální definice systémové funkce. V kapitole je definován požadavek dle [21] a jsou shrnuty základní pojmy a postupy týkající se inženýrství požadavků *(Requirements Engineering)*.

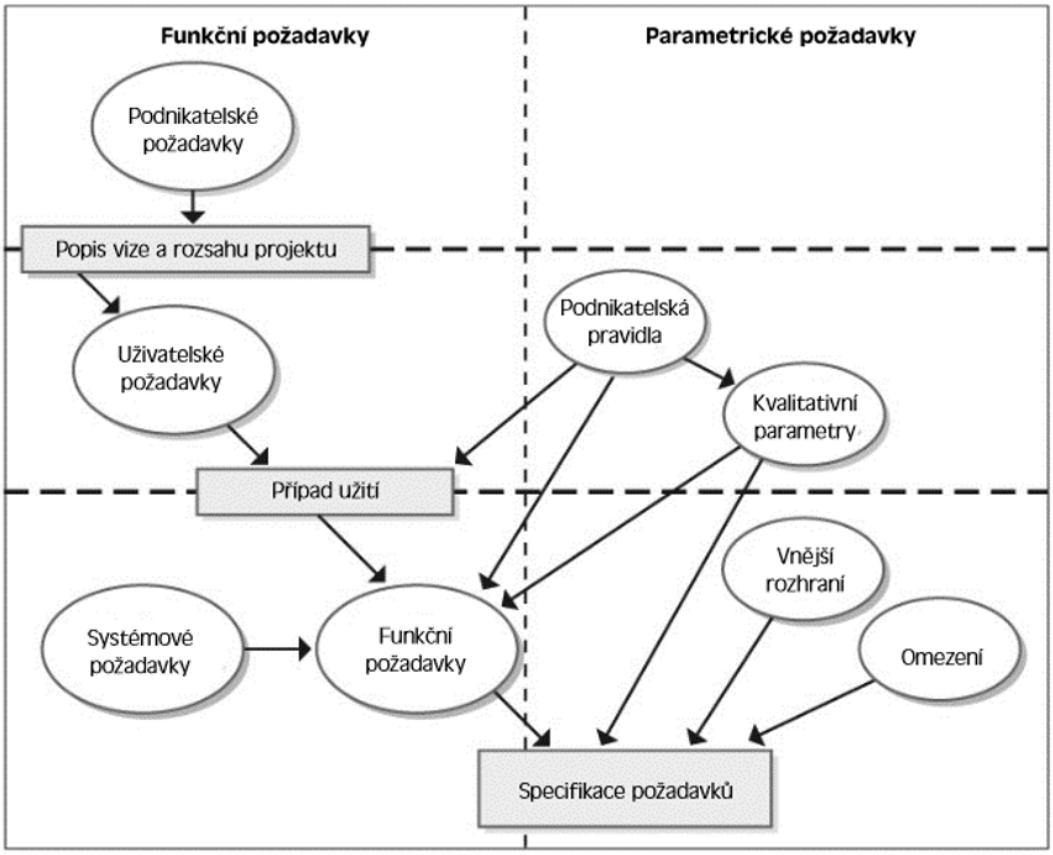
Úlohy lze spravovat velmi jednoduše například pomocí nástroje Microsoft Excel. Jednoduchá správa se ale stává komplikovanou, je-li jich velké množství a je-li potřeba efektivně sledovat jejich plnění. Existují sofistikované nástroje pro automatizovanou správu úloh, které jsou popsané v podkapitole 4.2. Jsou popsány nástroje JIRA Software, Redmine a Trello.

4.1 Inženýrství požadavků

Dle slovníčku softwarové terminologie (citováno z [21]) je požadavek:

- *"Podmínkou nebo funkcí, kterou uživatel potřebuje pro řešení problému nebo dosažení nějakého cíle.*
- *Podmínkou nebo funkcí, kterou musí systém nebo jeho část splňovat, aby vyhověl smlouvě, standardu, specifikaci nebo jinému dokumentu, jenž se formálně vztahuje.*
- *Dokumentovanou podobou některého z předchozích dvou bodů."*

Definice zahrnuje jak uživatelův pohled na vnější chování systému, tak vývojářský pohled například na vnitřní parametry systému. Z důvodů, že neexistuje univerzální definice, pro usnadnění komunikace a pochopení pojmu musí organizace zavést jednotné názvosloví. Karl Wiegers ve své knize Software requirements dělí požadavky na podnikatelské, uživatelské, funkční a systémové [21]. Na obr. č. 4.1 je znázorněno členění požadavků a vztahy mezi různými typy.



Obrázek 4.1: Typy požadavků a jejich vztahy, [21]

Podnikatelské požadavky *(business requirements)* označují cíle, kterých chce organizace prostřednictvím systému dosáhnout [21]. Uživatelské požadavky definují cíle a úkony jednotlivých uživatelů. Funkční požadavky popisují softwarovou funkcionalitu, kterou vývojáři implementují za účelem naplnění cílů uživatelů. Systémové požadavky jsou celkové požadavky na systém, který je složen z dílčích podsystémů. Požadavky jsou popisované ve specifikaci požadavků, která líčí chování systému. Specifikace požadavků může být dokument, databáze, tabulka či informace uložená v nástroji určenému ke správě požadavků.

Práci s požadavky lze rozdělit na vývoj požadavků a na jejich správu [21]. Do vývoje požadavků spadá sběr, analýza, specifikace a kontrola. Jedním z typických výstupů jsou dokumenty popisující rozsah a vizi projektu, případy užití, specifikaci požadavků, datový slovník nebo datové modely. Jsou-li dokumenty schváleny zákazníkem, vývoj požadavků končí a začíná jejich správa. V následující podkapitole jsou popsány principy a technicky pro správu požadavků.

4.1.1 Principy a techniky pro správu požadavků

Do správy požadavků spadají aktivity, které udržují integritu, konzistenci, aktuálnost a přesnost odsouhlasených požadavků. Dle [21] je správa požadavků rozdělena na:

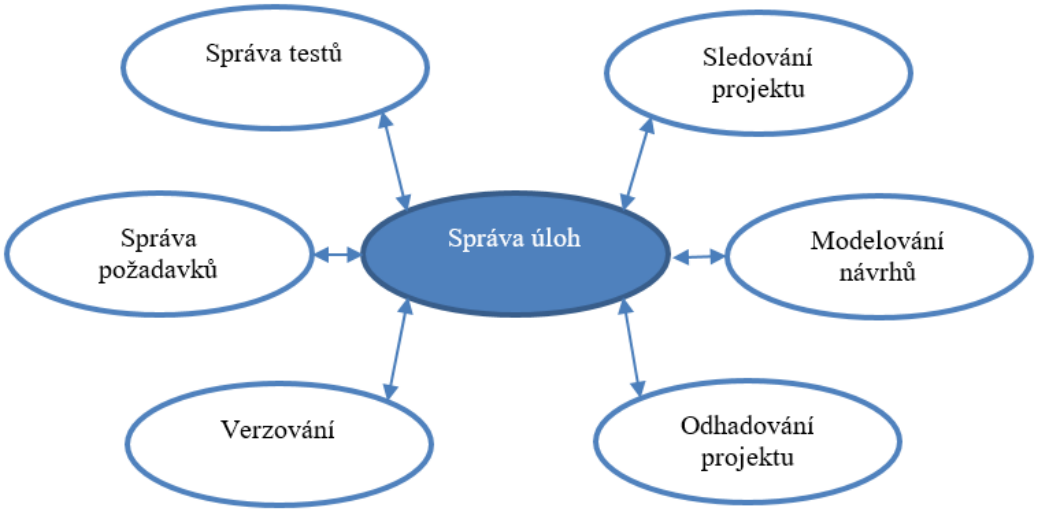
- řízení změn provedených ve směrné verzi,
- aktualizace projektového plánu podle požadavků,
- verzování jednotlivých požadavků i celkové dokumentace,
- sledování stavu požadavků ve směrné verzi,
- udržování odkazů mezi jednotlivými požadavky a dalšími pracovními postupy.

Každá společnost si definuje vlastní postupy, které členové projektového týmu pro správu požadavků používají [21]. Procesy by měly být popsané v metodice a zaměstnanci by s ní měli být seznámeni například formou školení. V metodice by měla být stanovena zodpovědnost za řízení požadavků. Obvykle se o nastavení podpůrných nástrojů, definici atributů požadavků, koordinaci změn stavu požadavků či nastavení odkazů nebo priorit stará projektový analytik.

4.2 Nástroje určené pro správu úloh

Robustní řešení nabízejí ukládání atributů k úloze, sledování stavu úlohy, řízení přístupů, komunikaci mezi členy týmu či sledování dílčích úkolů.

Nástroje jsou velmi často integrovány s dalšími nástroji (obr. č. 4.2). Při výběru nástroje je potřeba zohlednit možné integrace se správou testů, sledováním projektu, verzováním, modelováním návrhu, odhadováním projektu či řízení změn požadavků [21].



Obrázek 4.2: Možné integrace nástrojů

Úspěch používání nástroje je především odvíjen od používání uživateli [21]. Motivovaní uživatelé dokážou procesy vylepšit i s průměrnými nástroji, kdežto nemotivovaní či špatně proškolení uživatelé nedokážou využít funkcionalit robustních řešení a vynaložená investice tak nemá požadovanou návratovou hodnotu. Při výběru nástroje je nezbytné přijmout počáteční investici. Dále je nutné zohlednit čas strávený nad analýzou i učením.

V následujících kapitolách jsou popsány nástroje určené ke správě úloh. Nástroje byly vybrány pro účely praktické části této práce. V kapitole 4.2.4 jsou tyto nástroje srovnány.

4.2.1 JIRA Software

JIRA Software je nástroj sloužící k evidenci požadavků a chyb při vývoji softwaru [2]. Nástroj je vyvinut společností Atlassian Corporation v programovacím jazyce Java. Mezi hlavní funkce, které nástroj poskytuje, patří: Scrum boardy, Kanban Boardy, reporting, plány, vlastní filtry, integrace s nástroji pro vývojáře, nastavení životního cyklu úlohy a další.

Nástroj je nejčastěji používán pro sledování chyb softwaru, ale díky pokročilým funkcím je vhodný i pro zákaznickou podporu nebo projektový management. Dokumentace vedená v nástroji se ve většině případech využívá jako dokumentace pro zákazníka.

V následující tabulce 4.1 je uvedena cena nástroje. V rámci řešení Jira Software Cloud provozovatel zajišťuje hosting a instalaci. Tato možnost je převážně pro týmy, které chtějí začít pracovat rychle a snadno. V případě řešení Jira Software Server a Jira Software Data Center je Jira Software

hostován na hardwaru zákazníka. Tým si sám spravuje všechny podrobnosti nastavení a hostování.

Správa	**Počet uživatelů**	**Cena**	**Období**
Cloud	16 - 25	40 160 Kč (1 750 $)	roční
	26 - 50	80 319 Kč (3 500 $)	roční
	51 - 100	160 639 Kč (7 000 $)	roční
Server	10 - 25	57 371 Kč (2 500 $)	neomezené
	26 - 50	103 268 Kč (4 500 $)	neomezené
	51- 100	190 472 Kč (8 300 $)	neomezené
Data Center	500	275 381 Kč (12 000 $)	roční

Tabulka 4.1: Cena nástroje JIRA Sotware

Do nástroje je možné integrovat i další aplikace, které jsou dostupné na platformě Atlassian Marketplace. Například lze zakoupit aplikace:

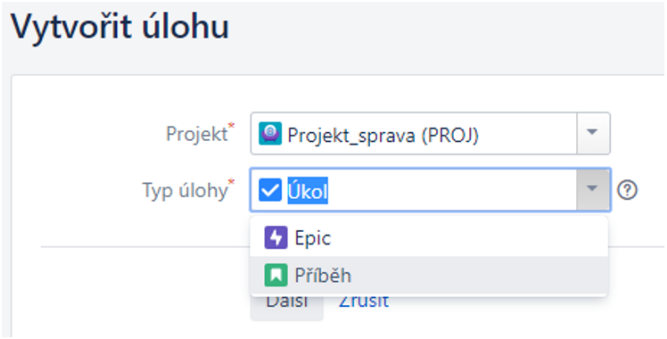
- **Zephyr for Jira - Test Management** – aplikace určená ke správě testů,
- **Git Integration for Jira** – aplikace zajišťující integraci JIRA Software s GIT,
- **Tempo Planner** – aplikace určená k plánování lidských zdrojů,
- **BigGantt - Gantt Chart for Jira** – aplikace určená k vytvoření Gantt digramu, WBS či k hledání kritických cest.

Zkušební verze nástroje JIRA Software i jednotlivých integrovaných aplikací jsou po dobu třiceti dní zdarma dostupné z oficiálních webových stránek nástroje [2].

Použití JIRA software pro správu úloh

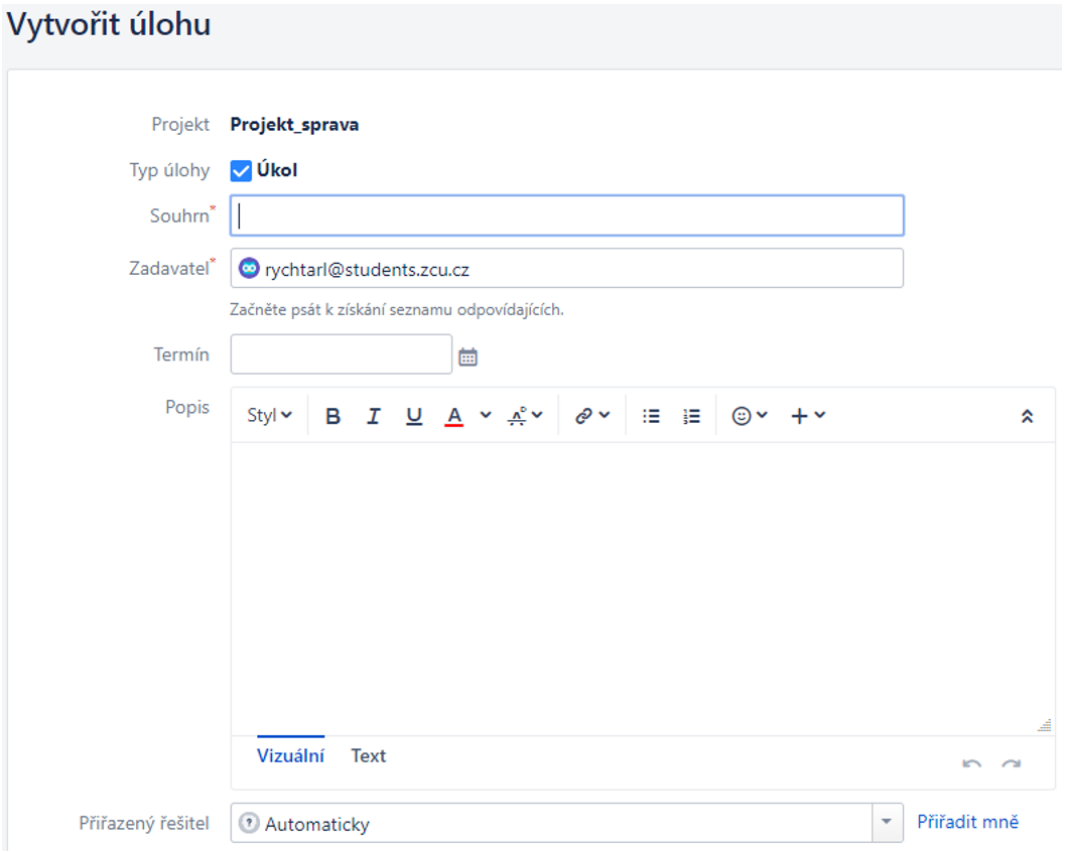
Projekt lze v nástroji založit třemi možnostmi, a to jako: správa projektů, scrumový vývoj softwaru a kanbanový vývoj softwaru.

Při vytvoření úlohy *(Task)* se nejprve definuje její typ (obr. č. 4.3). Nástroj nabízí základní typy úloh jako je úkol, problém, požadavek a další. Správce systému má možnost definovat vlastní typy úloh pro jednotlivé projekty.



Obrázek 4.3: JIRA: Výběr typu úlohy

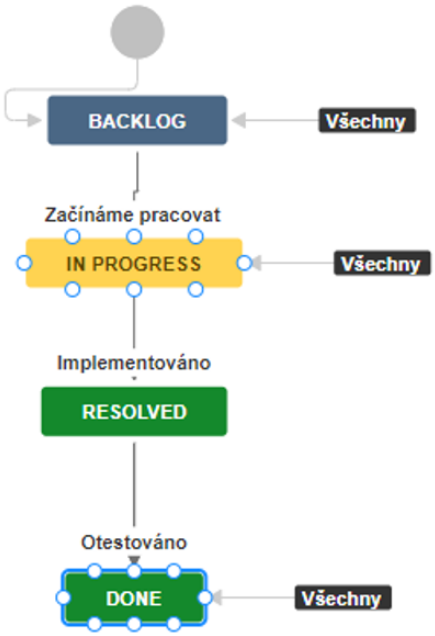
U úlohy se nakonfigurují atributy, které se při vytváření vyplňují (viz obr. č.4.4). Atributy úlohy lze označit jako povinné, či nepovinné. Mezi atributy patří například název projektu, typ úlohy, název, popis, priorita a další.



Obrázek 4.4: JIRA: Vytvořit úlohu

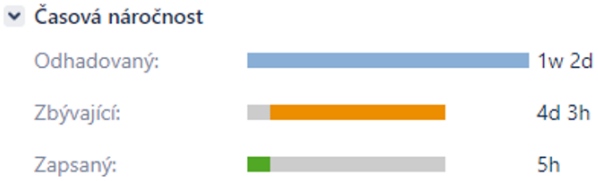
U jednotlivých typů úloh lze definovat životní cyklus (obr. č. 4.5). Cyklus zobrazuje stavy a jejich přechody. U každé úlohy je tak zřejmé v jaké fázi

životního cyklu se nachází. Zároveň lze u jednotlivých přechodů definovat podmínky jako například oprávnění.



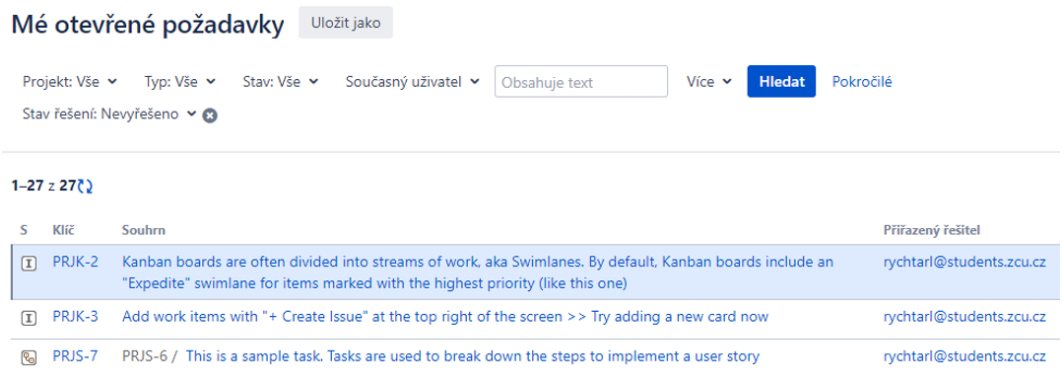
Obrázek 4.5: JIRA: Nastavení životního cyklu

Na úloze jsou zobrazeny základní informace, které se zapsaly při vytváření. Dále, kdo úlohu vytvořil a kdo na ní aktuálně pracuje. K úloze lze vkládat komentáře či přílohy. K úloze je také možné dodat časový odhad. Nástroj poté přehledně zobrazuje, kolik času bylo na úloze odpracováno a kolik času zbývá pro její dokončení(obr. č. 4.6). K úloze se dá také vložit štítek *(label)*.



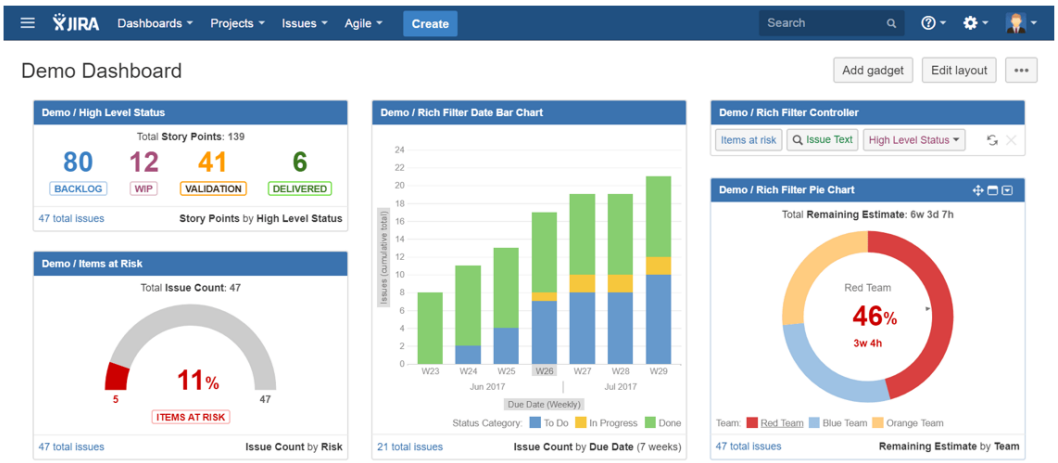
Obrázek 4.6: JIRA: Časová náročnost úlohy

Nástroj umožňuje vytváření filtrů či nástěnek. Ukázkový filtr je na obr. č. 4.7. Filtry lze sestavovat na základě atributů úloh. Výsledky filtru lze tisknout, i exportovat do formátu RSS, CSV, HTML, XML a Word.



Obrázek 4.7: JIRA: Filtr úloh

Ukázková nástěnka je na obrázku č. 4.8.



Obrázek 4.8: JIRA: Nástěnka

Nástroj JIRA Software je velmi uživatelsky přívětivý a dá se využít pro různé typy projektů. Nástroj je hodnocen jako velmi intuitivní. Nástroj také umožňuje projekty řídit. Je umožněno reportování přehledů za definované období. Lze sledovat například: "Přehled časové náročnosti", "Vytížení pracovníka", "Doba řešení požadavků"a jiné. Pozitivně je hodnocena možnost štítkování úloh. Tým má možnost jednoduše definovat vlastní štítky, pomocí kterých lze úlohy filtrovat. Dále je kladně hodnoceno vytváření filtrů a nástěnek.

4.2.2 Redmine

Redmine je flexibilní webová aplikace určená k řízení projektů a správě požadavků [10]. Nástroj je implementován v jazyce Ruby a je multiplatformní.

Mezi hlavní funkce nástroje se řadí: sledování více projektů, Ganttův diagram a kalendář, správa dokumentů a souborů, projektová fóra a další. Redmine je otevřený software *(open source)* a vydává se za podmínek GNU General Public License v2 (GPL).

Nástroj lze rozšířit pomocí produktů Easy Redmine. K dispozici jsou například pluginy: Resource Management Plugins, Agile Management Plugins či Finance Management [12]. Cena pluginu se odvíjí od počtu uživatelů a typu licence (viz tabulka č. 4.2).

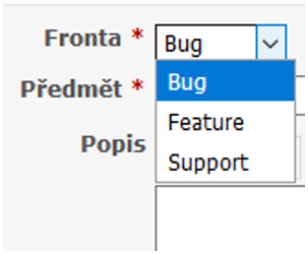
Typ licence	**Počet uživatelů**	**Cena**	**Období**
Cloud	25	1 789 Kč (78 $)	měsíční
	50	2 707 Kč (118 $)	měsíční
	100	5 460 Kč (238 $)	měsíční
Server	25	28 400 Kč (1 238 $)	roční
	50	56 386 Kč (2 458 $)	roční
	100	113 278 Kč (4 938 $)	roční

Tabulka 4.2: Cena pluginů Easy Redmine

Kompletní instalace je popsána na oficiálních stránkách nástroje [10]. Požadavky k instalaci jsou operační systém typu Unix, Linux, macOS nebo Windows. Dále je potřeba Ruby interpreter a databáze buď MySQL nebo PostgreSQL nebo Microsoft SQL Server nebo SQLite 3. Online ukázka je k dispozici na adrese *http://demo.redmine.org/*. Uživatelům je umožněno vytvářet vlastní projekty a vyzkoušet funkce nástroje. Vzhled aplikace lze přizpůsobit pomocí volně dostupných témat (skinů).

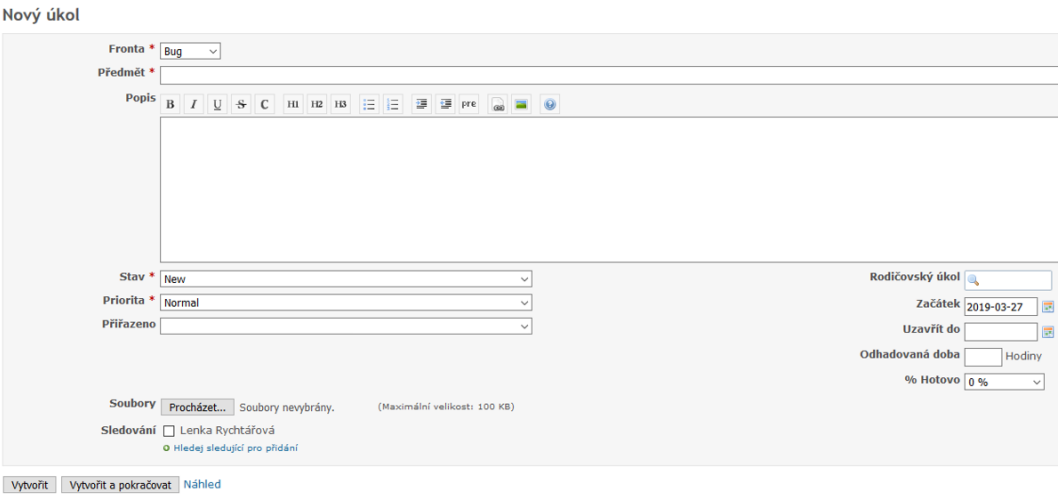
Použití Redmine pro správu úloh

V projektu je umožněno definovat tzv. moduly a fronty. Mezi moduly patří sledování úkolů, sledování času, kalendář, Gantt, diskuse, novinky, dokumenty, soubory a wiki. Frontou je označován typ úlohy. Lze zvolit úlohu typu "Bug", "Feature"a "Support"a další (viz obr. č. 4.9). Možnosti se odvíjí od konfigurace nástroje.



Obrázek 4.9: Redmine: Typ úlohy

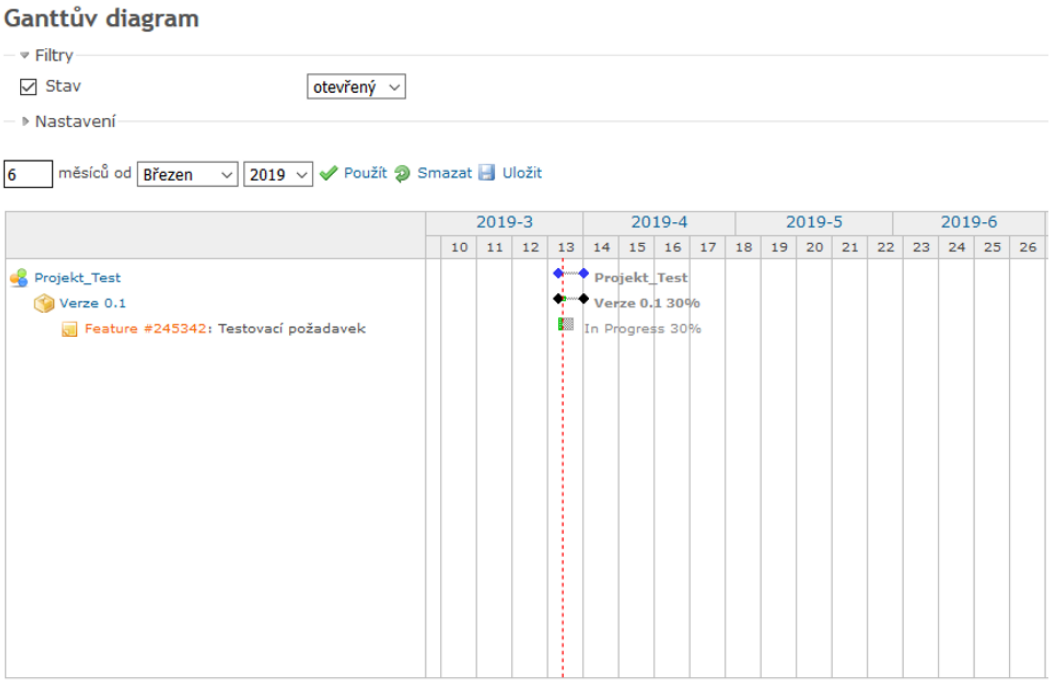
Úloha obsahuje atributy jako například fronta, předmět, popis, stav, priorita, přiřazeno a další (viz obr. 4.10). Atributy jsou opět závislé na konfiguraci nástroje. Na úloze lze sledovat procentuální vyjádření plnění a sledovat čas strávený nad požadavkem.



Obrázek 4.10: Redmine: Založení úlohy / úkolu

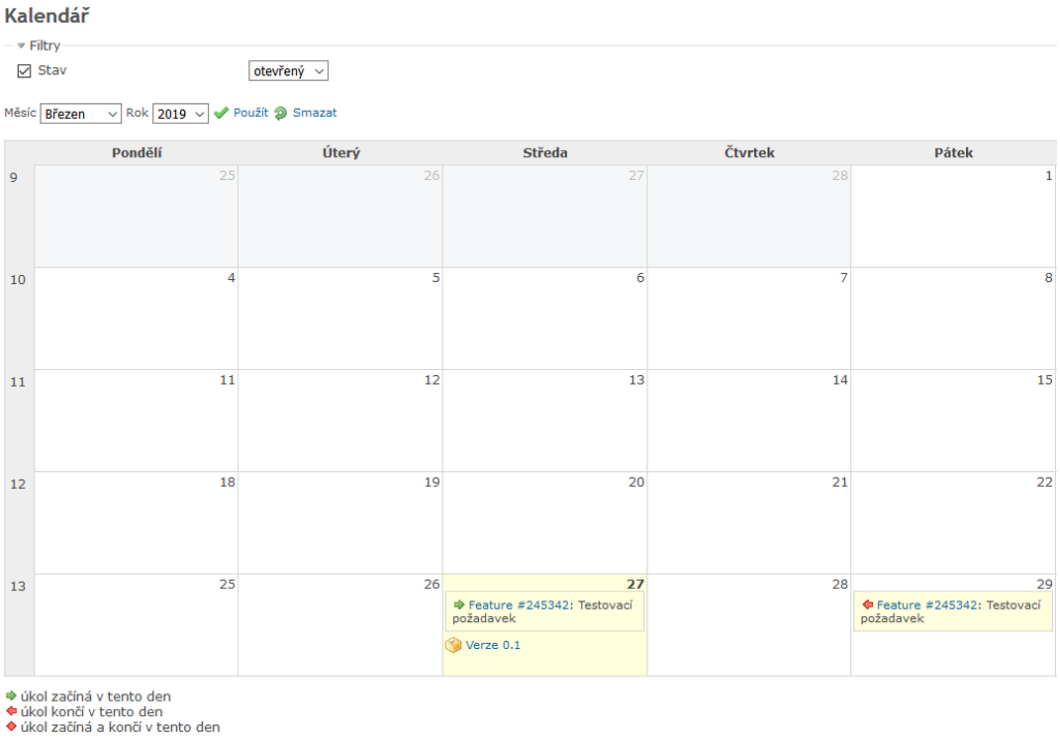
Redmine umožňuje v administrační sekci konfiguraci stavů úloh. Speciální funkcí je například dědičnost definic projektů na podprojekty.

Nástroj umožňuje definovat filtry na základě atributů požadavků. K filtru se poté zobrazuje Granttův diagram, viz obr. č. 4.11.



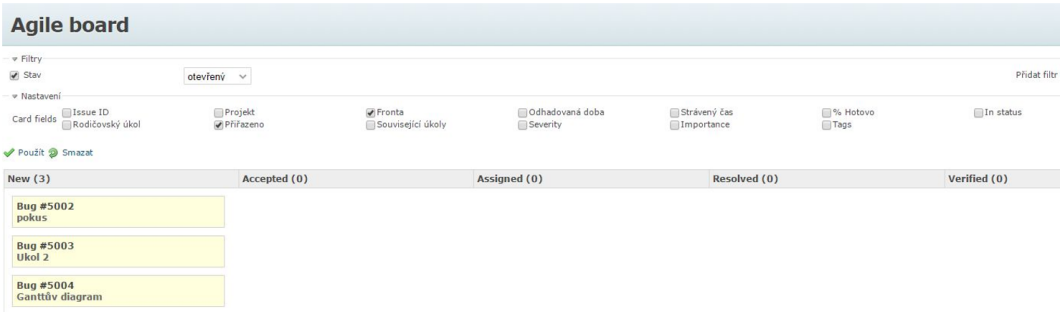
Obrázek 4.11: Redmine: Ganttův diagram

Na základě nastaveného filtru se také zobrazuje kalendář, ve kterém jsou uvedené relevantní aktivity (obr. č. 4.12).



Obrázek 4.12: Redmine: Kalendář

Pomocí modulu Agile lze vytvářet nástěnky. Na obr č. 4.13 je znázorněna nástěnka s úlohami, které jsou zobrazeny na základě stavů úlohy.



Obrázek 4.13: Redmine: Agile board

Nástroj Redmine je uživatelsky přívětivý a intuitivní. Kladně je hodnocen kalendář a Ganttův diagram. Lze tak jednoduše sledovat naplánované aktivity. Negativně je hodnoceno upravení požadavku, kdy se zobrazí kompletní formulář požadavku. Jednotlivá pole nelze upravovat.

4.2.3 Trello

Trello je nástroj používaný k řízení projektů pomocí systému Kanban. Kanban byl vytvořen výrobní společností Toyota [1]. V systému je použita objednávková karta "Kanban"jako plánovací systém. V softwarovém inženýrství je přístup využíván k vizualizaci pracovního postupu. Nástroj je založen na vizuální prezentaci informací. Na nástěnkách jsou pomocí karet zobrazeny úkoly [4]. Jsou definovány sloupce, ve kterých se dle systému Kanban jednotlivé úkoly nacházejí.

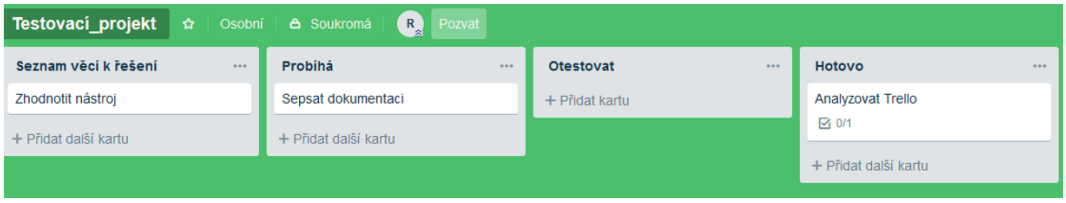
V následující tabulce 4.3 jsou uvedeny ceny nástroje [4]. Rozdíl mezi typem licence, která je zdarma, a Business Class spočívá hlavně v poskytování podpory od prodejce a v počtu vložených vylepšení. Nástroj v základním nastavení nenabízí mnoho funkcí. Pro uživatelské potřeby jsou k dispozici tzv. "vylepšení", které rozšíří či doplní stávající funkčnosti. Rozdíl je dále v definování vlastních pozadí a nálepek, v množství týmových nástěnek, v nastavení emailové notifikace nebo v možnostech exportu dat. Rozdíl mezi typem licence Business Class a Enterprise spočívá například v definování nástěnky viditelné pro celou organizaci nebo v přizpůsobeném náboru.

Typ licence	**Počet uživatelů**	**Cena**	**Období**
Zdarma	neomezené	zdarma	neomezené
Business Class	25	68 776 Kč (2 997 $)	roční
	50	137 553 Kč (5 994 $)	roční
	100	275 335 Kč (11 988 $)	roční
Enterprise	25	143 405 Kč (6 249 $)	roční
	50	286 809 Kč (12 498 $)	roční
	100	573 619 Kč (24 996 $)	roční

Tabulka 4.3: Cena nástroje Trello

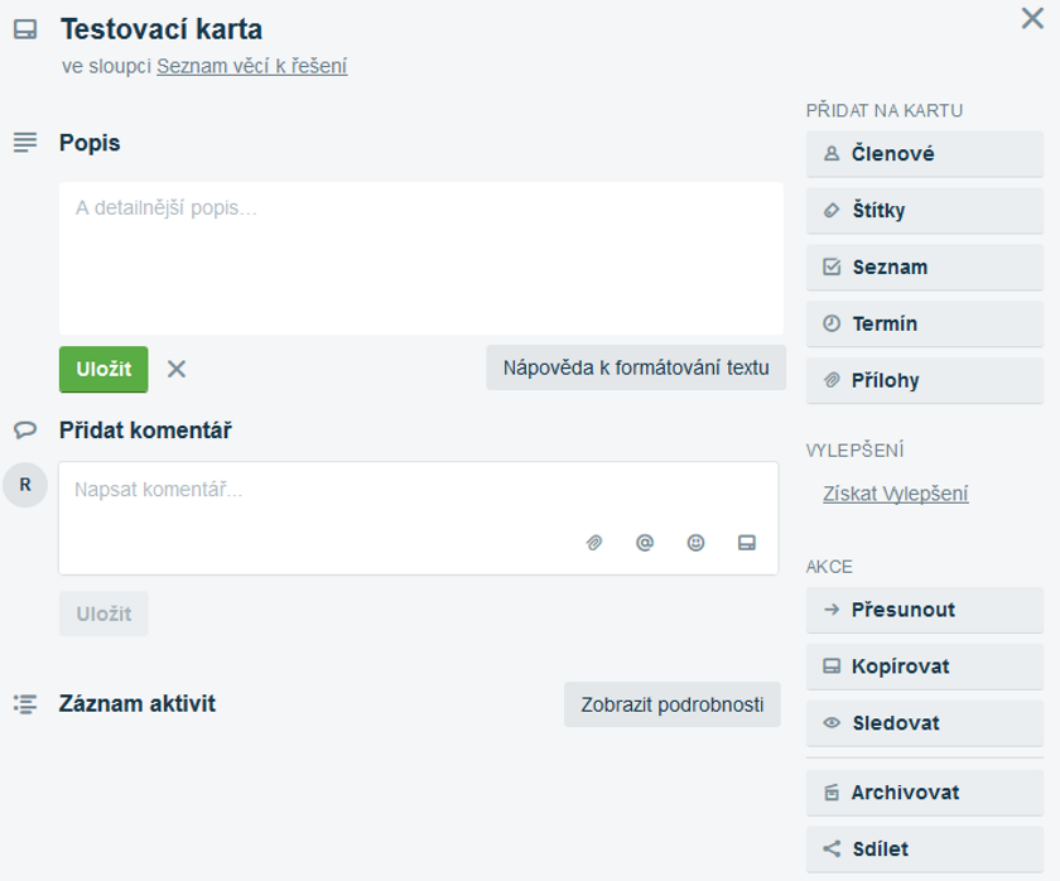
Použití Trello pro správu úloh

Každý projekt má vlastní Kanban nástěnku s libovolně definovanými sloupci, které značí stav úlohy (viz obr č. 4.14).



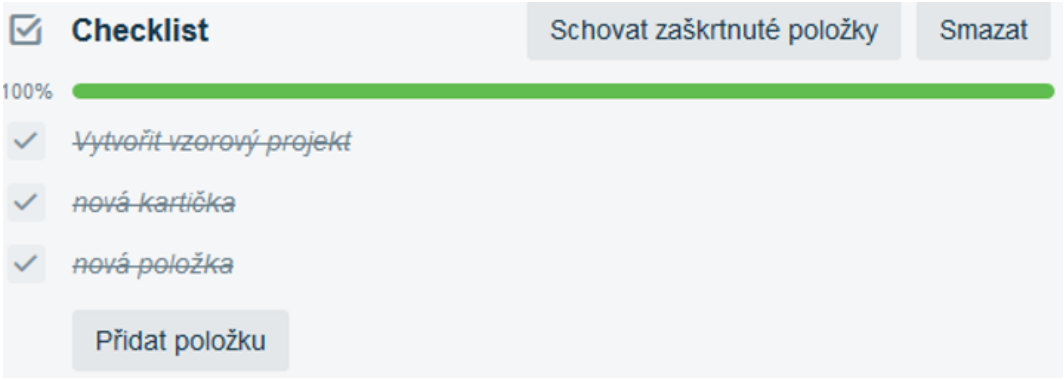
Obrázek 4.14: Trello: Kanban nástěnka

Při vytváření úlohy, nebo-li karty, se definují atributy, jako je popis, členové, štítky, termíny nebo přílohy (obr. č. 4.15). Na kartě je umožněno vkládat komentář i přílohu. Dále jsou u karty dostupné akce jako kopírování archivování či sdílení.



Obrázek 4.15: Trello: Karta

V rámci úlohy lze také vytvářet úkoly pomocí tzv. "Checklist"(obr. č. 4.16). Členové týmu tak mohou definovat úkoly k požadavku a sledovat jejich plnění.



Obrázek 4.16: Trello: Checklist

Nástroj je velmi jednoduchý a uživatelsky přívětivý. Kvůli jeho jednoduchosti je účelný na úkolování jednotlivých členů týmu. Nástroj dle autora práce není ale na správu požadavků pro velké projekty vhodný.

4.2.4 Shrnutí funkčností nástrojů

V následující tabulce č. 4.4 jsou shrnuty náklady na pořízení jednotlivých nástrojů. Srovnání je s ohledem na počet padesáti uživatelů.

	JIRA	**Redmine**	**Trello**
Délka licence	neomezená	neomezené	roční
Cena	103 268 Kč	0 Kč	137 553 Kč

Tabulka 4.4: Ceny nástrojů a délka licence pro 50 uživatelů

V tabulce č. 9.1 jsou shrnuty funkčnosti nástroje JIRA Software, Redmine a Trello. Pokud je u funkčnosti uvedeno "rozšíření", bylo dohledáno dostupné řešení, které danou funkčnost poskytuje.

Funkčnost	**JIRA**	**Redmine**	**Trello**
Definice stavu úloh	ano	ano	ano
Sledování stavů úloh	ano	ano	ano
Definice vývojového procesu úlohy	ano	ano	ano
Přikládání příloh	ano	ano	ano
Evidence historie úlohy	ano	ano	ano
Nastavení emailových notifikací	ano	ano	ano
Definice přehledů a statistik	ano	ano	ano
Kopírování úloh	ano	ano	ano
Sledování SLA	rozšíření	rozšíření	ne
Stanovení časového odhadu k úloze	ano	ano	rozšíření
Sledování odpracovaného času u úlohy	ano	ano	rozšíření
Reportování (excel a jiné)	ano	ano	ano
Integrace na GIT	rozšíření	ano	ne
Integrace na testování	rozšíření	ne	ne
Full-text vyhledávání	ano	ano	ano
Vytváření filtrů	ano	ano	ano
Vytváření pracovních ploch	ano	ano	ano
Řízení přístupu	ano	ano	ano
Použití v mobilní aplikaci	ano	rozšíření	ano
Sledování a řízení projektu	ano	ano	ano
Plánování verzí	ano	ano	ne

Tabulka 4.5: Shrnutí funkčností nástrojů

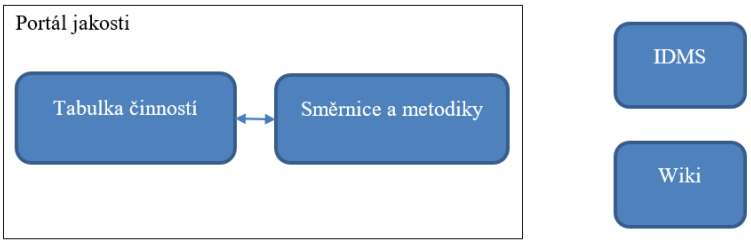
5 Metodika vybrané společnosti

Jedním z cílů práce je seznámit se s metodikou vybrané společnosti. V kapitole jsou popsány dokumenty Projektový zákon a Tabulka činností, které vycházejí z interní dokumentace.

5.1 Profil společnosti

Společnost je poskytovatelem informačních systémů se zaměřením na vývoj aplikací na míru, systémovou integraci, správu dokumentů a byznys inteligenci. Ve společnosti převládají servisní projekty, na kterých je uplatňován vodopádový model, popsaný v kapitole 2.1. Vývojové projekty inklinují taktéž k vodopádovému modelu, ačkoli je prosazován iterativní přístup. Metodika společnosti vychází z Rational Unified Process (viz kapitole 3.1). Metodika byla na základě specifik projektů a zkušeností upravena společností dle potřeb.

Společnost disponuje rozsáhlou dokumentací uloženou na online úložišti Portál jakosti. Výchozím dokumentem je Projektový zákon, který obsahuje Tabulku činností. V dokumentu Tabulka činností jsou popsány různé typy událostí, vykonávané činnosti nad událostmi a jejich řešitelé. K většině činností jsou mimo jiné k dispozici šablony. Projektová dokumentace je uložena v platformě SharePoint IDMS. Návody a příručky pro zaměstnance jsou vytvářeny na wiki stránkách společnosti. Schéma uložení dokumentace je na obr. č. 5.1.



Obrázek 5.1: Schéma uložení dokumentace společnosti

5.2 Projektový zákon

Cílem Projektového zákona je popis pravidel řízení projektů společnosti. Projekty jsou tvořeny fází zahájení, přípravy, konstrukce a nasazení. Samostatnou fázi tvoří Řízení projektu. Fáze jsou dále děleny na kroky. Model životního cyklu se v praxi realizuje tvz. Plánem projektu. V Plánu projektu jsou definované kroky jednotlivých fází.

Projektový zákon a Tabulka činností jsou výchozími dokumenty pro řízení jak vývojových, tak servisních projektů. Vzhledem k různým specifikům projektů se kroky Plánu projektu konfigurují dle potřeby.

Zaměstnanci jsou dle Projektového zákona obsazováni do rolí:

- **sponzor projektu** – osoba s celkovou zodpovědností za uspokojení potřeb zákazníka,
- **vedoucí projektu** – osoba zodpovědná za realizaci a organizační chod projektu,
- **hlavní projektant** – osoba zodpovědná za návrh řešení SW a věcnou stránku projektu,
- **hlavní návrhář-programátor** – osoba zodpovědná za vývoj navrženého SW,
- **hlavní konzultant** – osoba zodpovědná za implementaci SW v rozsahu dle dokumentace,
- **konzultant** – osoba zodpovědná za implementaci jednotlivých částí (modulů) SW,
- **test analytik a tester** – osoby zodpovědné za otestování produktu,
- **vedoucí obchodního případu** – osoba zodpovědná za smluvní zastřešení všech aktivit probíhajících na projektu,
- **ředitel úseku** – není členem projektového týmu, zodpovídá za přidělení dostatečného množství pracovníků do projektu a za jejich kvalitu.

5.3 Tabulka činností

Fáze projektu jsou podrobněji popsány v Tabulce činností. Tabulka obsahuje profese, události, popis činností a odkazy na relevantní dokumentaci. V následujících podkapitolách jsou fáze a kroky popsány.

Tabulka činností je dle společnosti relevantní jak pro servisní, tak pro vývojové projekty. Vzhledem k faktu, že ve společnosti převládají a dlouhodobě přetrvávají servisní projekty, je metodika uzpůsobena spíše potřebám servisních projektů. Z tohoto důvodu je obtížné metodiku aplikovat na velké vývojové projekty. V průběhu vypracovávání této práce společnost vysoutěžila zakázku na implementaci rozsáhlého informačního systému. Na základě zkušeností z vývojového projektu jsou sbírány náměty a návrhy na vylepšení dané metodiky, které budou po skončení projektu zapracovány.

5.3.1 Fáze Zahájení

Cílem fáze Zahájení je připravit vlastní zahájení projektu. Fáze je tvořena kroky: Projektový záměr, Příprava startovní dokumentace a Specifikace účelu a cílů projektu.

Projektový záměr slouží k vyhodnocení realizovatelnosti projektu. Výstupem je dokument Projektový záměr, pro který je dostupná šablona. Za činnost je zodpovědná profese vedoucí projektu.

Krok Příprava startovní dokumentace má za cíl vytvořit úvodní dokumentaci k projektu. V kroku je realizováno například zřízení projektového prostoru, připravení plánu projektu, analýza rizik a další. V případě servisních projektů je postup zjednodušen. Za činnost je zodpovědná profese vedoucí projektu.

V posledním kroku je specifikován účel a cíle projektu. Je nutné zohlednit, že rozsah u některých vývojových projektů je v tomto kroku již smlouvou stanoven. Krok však může být součástí tvz. "Specifikačního projektu". Specifikační projekt je založen na vyžádání zákazníka za účelem stanovení rozsahu projektu. Za činnost je zodpovědný vedoucí projektu.

5.3.2 Fáze Příprava

Záměrem fáze Příprava je příprava výstupů spojených se specifikací projektu. Výstupy z fáze slouží jako vstup pro další fázi Konstrukce. Fáze je tvořena kroky: Specifikace požadavků, Studie proveditelnosti, Analýza a návrh pro zákazníka, Posouzení analýzy v zákaznické podpoře(Posouzení ANZ v ZP), Posouzení nezaujatým pracovníkem, Analýza a návrh pro vývojáře, Architektura SW, Test analýza – plán a Test analýza – testovací scénáře.

Specifikace požadavků má za cíl definovat funkční a nefunkční požadavky. Funkční požadavky popisují funkčnost systému, zejména výstupy, přehledy a interakci s okolím. Nefunkční požadavky definují vlastnosti systému jako

celku a zahrnují omezení na systém. Za krok je zodpovědný analytik / projektant.

Studie proveditelnosti slouží k prokázání, že byla vybrána nejvhodnější varianta na řešení systému, a to jak z ekonomického, tak z technického hlediska. Na základě této studie se investor rozhoduje, zda dojde k realizaci projektu. Krok může být součástí "Specifikačního projektu". Za projekt je zodpovědný analytik / projektant.

V analýze a návrhu pro zákazníka se vyjasní s klíčovými uživateli jejich požadavky na systém. Jsou provedeny náčrty obrazovek tak, aby uživatel získal reálnou představu o řešení systému. V některých případech jsou dodány i odhady pracností. Za krok je zodpovědný analytik / projektant.

Zpracování posouzení ANZ v ZP je ověření správnosti zpracování analýzy z pohledu zákaznické podpory (ZP) jako zástupce zákazníka. Za krok je zodpovědný konzultant ZP. Posouzení nezaujatým pracovníkem zpracovává také Konzultant ZP, který ověří správnost navrženého řešení. Posouzení je provedeno osobou, která není v řešitelském týmu projektu.

Krok Zpracování analýzy a návrhu pro VV vytváří a udržuje specifikaci funkcí systému. Za krok je zodpovědný analytik / projektant.

Cílem kroku Architektura SW je zpracovat architekturu softwaru. Krok probíhá paralelně s tvorbou analýzy. Dokument je dán šablonou a zpracovává jej programátor. Test analýza – plán stanovuje základní plán testování. Za krok je zodpovědný test analytik, stejně tak jako za krok Test analýza – testovací případy. Testovací případy jsou v kroku zpracovány a připraveny pro další fáze.

5.3.3 Fáze Konstrukce

Účelem fáze Konstrukce je vytvořit a ověřit softwarový produkt. Fáze je tvořena následujícími kroky: Programování, Ověření v AP, Testování v ZP, Správa technického prostředí a model nasazení, Projekt zavedení u zákazníka.

V kroku Programování je vytvořen softwarový návrh. Následně je realizována implementace kvalifikovanými programátory. Za krok je zodpovědný programátor. Krok Ověření v AP realizuje analytik / projektant, který ověří, zda je implementace v souladu s analýzou.

Krok Testování v ZP má za cíl ověřit kvalitu produktu, snížit chybovost dodávaného softwaru a prokázat, že systém je ve shodě s požadavky a návrhy. V Tabulce činností ve fázi Konstrukce není popsáno, jak jsou případné neshody řešeny.

Krok Správa technického prostředí a model nasazení je v kompetenci

programátora. Výstupem je dokument, kde je popsáno prostředí aplikace, popis instalace a další.

Za krok Projekt zavedení u zákazníka je zodpovědný vedoucí projektu. Výstupem je dokument, kde je specifikováno, jak bude produkt u zákazníka zaveden. Dále jsou uvedeny potřebné součinnosti a možná rizika spojená se zavedením.

5.3.4 Fáze Nasazení

Fáze nasazení má za cíl nasadit již vytvořený produkt k zákazníkovi. Fáze je tvořena kroky: Instalace, Výroba dokumentace k produktu, Školení, Převzetí zákazníkem do ověřovacího provozu, Ověření pilotními uživateli, Plošné nasazení, Předání do rutinního provozu, Uzavření projektu.

Administrátor ručí za instalaci aplikace v prostředí zákazníka. Dokumentace k produktu obsahuje výstupy: uživatelská dokumentace a systémové dokumentace. Systémová dokumentace je určena pro správce produktu na straně zákazníka. Obsahuje rámcový popis systému, jeho architekturu a popis klíčových uživatelů. Za krok je zodpovědný konzultant ZP.

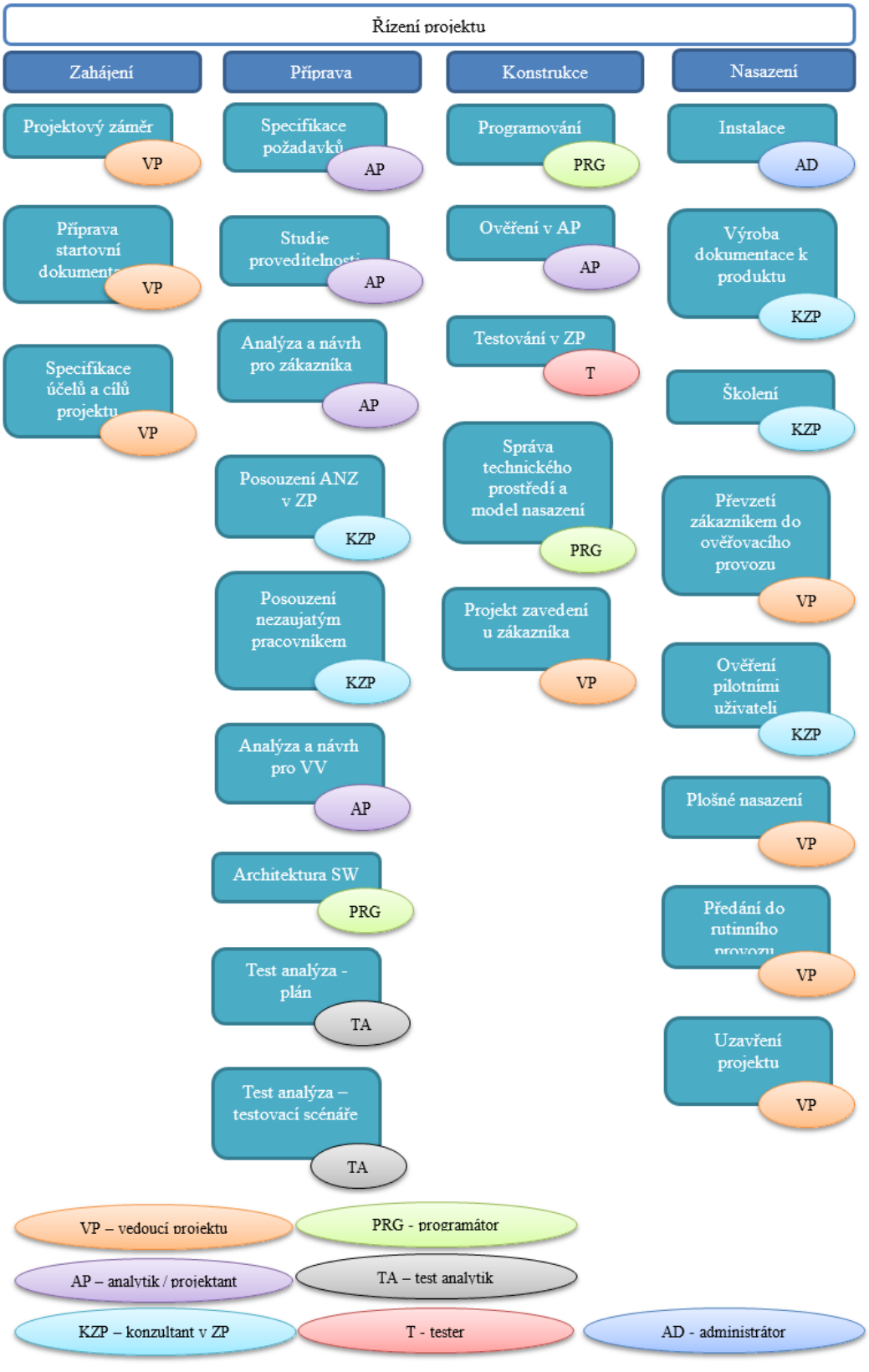
Cílem kroku Školení je připravit plán na proškolení uživatelů systému a následně školení realizovat. Za krok je zodpovědný konzultant ZP. V kroku Převzetí zákazníkem do ověřovacího provozu je systém předán zákazníkovi. Podmínkou pro jeho zahájení je dokončení vývoje systému, instalace, zpracování uživatelské dokumentace a školení. Za splnění kroku ručí vedoucí projektu.

Ověření pilotními uživateli probíhá v souladu se smluvní projektovou dokumentací. Zodpovědný pracovník je konzultant ZP. Případné problémy jsou aktivně řešeny s vedoucím projektu. Následuje krok Plošné nasazení, kdy je produkt nasazen ve všech organizacích zákazníka. Krok je řízen vedoucím projektu. Dalším krokem je Předání do rutinního provozu. Produkt je již v rutinním užívání a nastává formální ukončení instalace systému u zákazníka. Za krok je opět zodpovědný vedoucí projektu. Posledním krokem je Uzavření projektu. Jedná se o interní ukončení projektu. Projekt se převádí do servisního režimu. V rámci ukončení projektu je zhotoven dokument "Poučení z projektu", který je představen na poradách vedoucích projektů a uložen v projektové dokumentaci. Vzhledem faktu, že servisní projekty nemají stanovený konec, poučení z projektu je vypracováno každý rok. U vývojových projektů se poučení z projektu vypracovávají i průběžně dle potřeb. Odpovědnou osobou je vedoucí projektu.

5.3.5 Fáze Řízení projektu

Fáze Řízení projektu začíná zahájením projektu a končí jeho uzavřením. Cílem je projekt řídit tak, aby byl dokončen včas, zcela a za stanovenou cenu. V průběhu řízení projektu vedoucí projektu organizuje a vykonává metodické kroky. K úkolování členů týmu je využíván systém ISZA a ke sdílení informací úložiště IDMS. Ve fázi jsou definované činnosti, jako je řízení projektu na cíl, řízení lidských zdrojů a další.

Fáze s kroky a odpovědnými osobami jsou znázorněné na následujícím obr. č. 5.2.



Obrázek 5.2: Schéma fází a kroků metodiky společnosti

6 Analýza a návrh úprav metodiky

V následujících podkapitolách jsou identifikované problémy a návrhy řešení. Uvedená zjištění reflektují stav metodiky a procesů společnosti za období od září 2018 do ledna 2019.

6.1 Postup analýzy

Ve společnosti byla provedena kritická analýza poznatků nabytých nejen studiem metodiky, ale i rozhovory. Nejprve se konaly rozhovory s ředitelem úseku Metodik a směrnic, v rámci kterých byla metodika společnosti objasňována. Z rozhovorů již vyplývaly problémy, jež jsou popsány v této kapitole a v kapitole 8. Dále se konaly rozhovory společně s ředitelem úseku Metodik a směrnic a s ředitelem úseku Vývoje a inovací. Během rozhovorů byly představovány různé návrhy řešení, které byly diskutovány a podrobněji analyzovány. Například byly debatovány principy agilních přístupů a implementace agilních metodik na vývojových projektech. Návrhy, které byly z pohledu zástupců společnosti irelevantní, nejsou v této práci dále rozebírány.

Z jednotlivých schůzek byly vypracovány zápisy, které nejsou z důvodů ochrany obchodních tajemství společnosti součástí této práce. Identifikované problémy a možné řešení byly dále diskutovány se zástupci jednotlivých rolí. Konaly se dva pracovní semináře společně se dvěma řešitelskými týmy. Přítomni byli například vedoucí projektů, vedoucí vývojářů, analytici, testeři, konzultanti či programátoři. Jedním z cílů pracovních seminářů bylo ověřit identifikované problémy. Dalším cílem bylo identifikovat problémy další. Zápisy ze seminářů jsou uvedeny v přílohách A, B.

Návrhy řešení popsané v této kapitole jsou v souladu s odbornou literaturou dané problematiky. Většina řešení byla průběžně diskutována s ředitelem úseku Metodik a směrnic.

6.2 Správa metodiky

Metodika společnosti a jednotlivé postupy jsou uchovány v několika dokumentech. Každý dokument je ve správě tzv. garanta. Dokumenty nejsou spravovány na pravidelné bázi, ale příležitostně, pokud je zaznamenána po-

třeba změny. Dokument Projektový zákon, který definuje řízení procesů vývoje, byl naposledy aktualizován v roce 2017. Dokument Evidence a členění požadavků byl naposledy aktualizován v roce 2016. Dokument Logická architektura byla vytvořena 1. 2. 2018 a od té doby nebyla nad dokumentem provedena revize.

Metodika v současné době prochází přezkoumáním. Na novém projektu jsou ověřovány nové procesy. Plánované dokončení revizí je na konci roku 2019. Přezkoumání není v současné době formalizované, nevede se evidence změn a jejich vyhodnocování. Samotné používání metodiky není dostatečně monitorováno. Pro správu hlášení a úkolování jednotlivých členů týmu je předepisován nástroj ISZA. V praxi se využívá nástroj Trello. Používání jiného nástroje reflektuje nespokojenost s nástrojem ISZA na specifických projektech.

Společnost nemá zavedené řízené zlepšování jednotlivých procesů vývoje softwaru na pravidelné bázi. Procesy jsou diskutovány na úrovni středního managementu, ze kterých vznikají požadavky na změny metodik či jejich objasnění. Konají se pravidelné pohovory jednou za tři měsíce, kde jsou procesy diskutovány, ale pouze okrajově.

Zaměstnanci organizace mají možnost zasílat své náměty na vylepšení metodik a procesů na zákaznickou linku organizace. Tato možnost není aktivně využívána. Případná spokojenost či nespokojenost s metodikami nebo nástroji není mezi zaměstnanci proaktivně diskutována.

Jednotlivé dokumenty nejsou terminologicky konzistentní. Například Projektový zákon obsazuje pracovníky do rolí, kdežto Tabulka činností do profesí. V projektovém zákoně nejsou specifikovány všechny role (profese), které jsou uvedené v tabulce činností. Například Projektový zákon definuje role Hlavní konzultant a Konzultant, kdežto tabulka činností obsahuje pouze roli Konzultant ZP. Chybí zde definice pojmů jako je událost, činnost nebo Specifikační projekt.

6.2.1 Návrh řešení - Metodická znalostní skupina

Reakcí na slabé místo správa metodik popsané v kapitole výše je vytvoření tzv. Metodické znalostní skupiny (MZS). Skupina by měla za cíl být plně obeznámena s metodikami firmy, spravovat metodiky, monitorovat jejich používání, zlepšovat stávající procesy a šířit možnosti změny.

Znalosti se stávají jedním ze strategických zdrojů organizací. Disciplína znalostního managementu systematicky umožňuje vytváření hodnoty informací a znalostí. Problematika znalostního managementu - strategie, přístupy, znalostní skupiny a mnoho dalších jsou popsány v publikaci The New

Edge in Knowledge od autorů Carl O'Dell a Cindy Hubert [6].

Složení metodické znalostní skupiny (MZS)

Autoři O'Dell a Hubert doporučují vytvořit skupinu, která se bude problematice věnovat na plný úvazek [6]. Skupina se skládá z rolí MZS vůdce, MZS specialista, MZS komunikační direktor a MZS analytik.

Role MZS vůdce má na starosti celý program. Cílem je obhájit strategii a cíle u vedení organizace a také získat potřebné prostředky. Hlavní činnosti jsou:

- podpora strategie, rozvoj vize, poslání a cílů,
- řízení vývoje programu,
- propagace, prosazování stejného přístupu napříč odděleními,
- spolupráce s vedoucími oddělení na vytvoření společného porozumění a zaměření,
- hledání příležitostí, kde sdílení znalostí přidává hodnotu organizaci.

Role MZS specialista by měl disponovat expertní znalostí metodik. Osoba navrhuje a implementuje nové přístupy. Neméně důležitým úkolem je šířit metodiky jako organizační praktiky a jako systém podpory zaměstnanců. Hlavní činnosti jsou:

- monitorovat používání metodik napříč organizací,
- monitorovat a reportovat pokrok v používání metodik MZS vůdci,
- hledat společně se MZS vůdcem nové příležitosti a problémy v metodikách,
- navrhovat, implementovat a vylepšovat přístupy, procesy,
- monitorovat efektivitu a použití nástrojů a systémů,
- informovat o programu spolupracovníky.

Role MZS komunikační direktor má za úkol vyvíjet a řídit komunikaci se všemi zúčastněnými v programu. Osoba spolupracuje s ostatními členy skupiny a podporuje udržování povědomí o metodikách. Vyhledává a publikuje úspěšné příběhy a zajišťuje komunikaci směrem k vedení společnosti. Hlavní činnosti jsou:

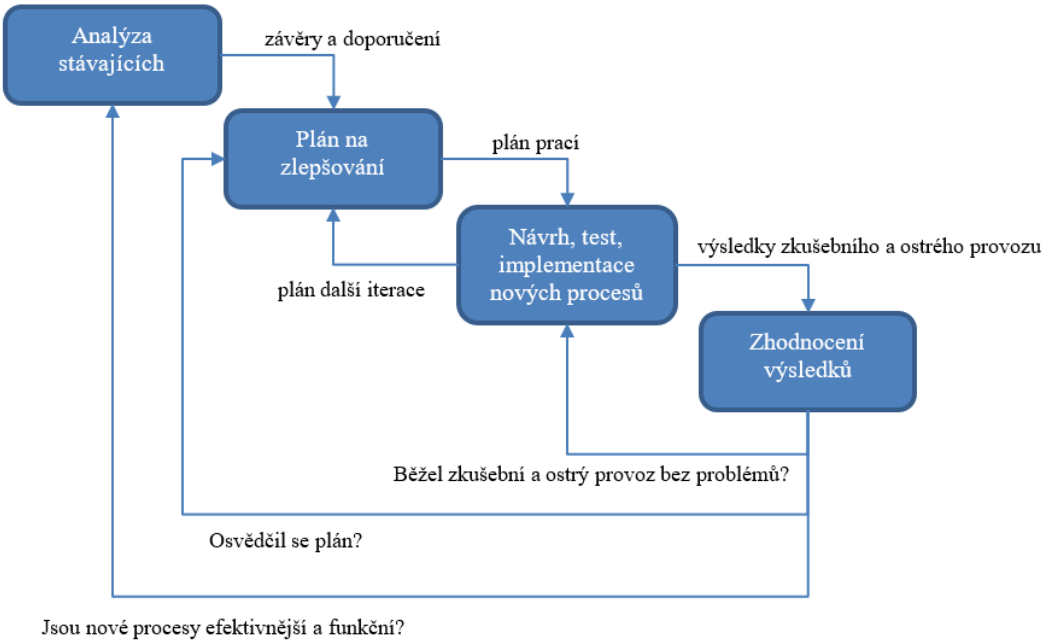
- plánování a vedení komunikačních strategií, usiluje o posílení povědomí o výhodách metodik, procesů a nástrojích,
- vede a pořádá schůzky, školení, konference či speciální události,
- řídí očekávání zaměstnanců.

Poslední role v týmu MZS analytik spojuje IT znalosti s očekáváním jednotlivých oddělení, zaměstnanců a spolupracovníků v metodické skupině. Díky spolupráci se všemi zainteresovanými osobami analytik zajišťuje, že potřeby organizace jsou naplněny, zaměstnanci dostávají adekvátní technologickou podporu, a že nástroje jsou efektivně využívány. Hlavní činnosti jsou:

- vyvíjet, implementovat a podporovat infrastrukturu ve spolupráci se znalostní skupinou,
- uspokojovat potřeby oddělení, zaměstnanců a vedení potřebnými nástroji.
- podporovat činnosti v oblasti vzdělávání v technologiích,
- modelovat osvědčené postupy a zavádět osvědčené procesy při používání podpůrných technologií [6].

Zlepšování procesů

Metodická znalostní skupina by měla v kompetenci i zlepšování procesů, které by dle [21] mělo probíhat postupně, cyklicky a nepřetržitě. Prvním krokem je analýza procesů, dále detekce problémů. Určení priority problémů a následně nalezení řešení. Řešení by zpočátku programu měla být méně nákladná, ale za to efektivní. Závěrečným krokem je vyhodnocení výsledků provedených změn. Schéma je na obr. č. 6.1.



Obrázek 6.1: Schéma zlepšování procesů

Prvním krokem je analýza stávajících procesů, jejich výhod a nedostatků. Systematický přístup nabízí strukturované dotazníky. Přesnější a podrobnější analýzy se docílí pomocí rozhovorů a diskuzí se členy týmu.

Zlepšování procesů je klasický projekt, po analýze následuje plánování. Strategický plán popíše zlepšování procesů jako celek. Taktické plány se zaměřují na konkrétní oblasti. Každý plán by měl uvést cíl navrhovaného zlepšení, jeho účastníky a seznam prací. V každém akčním plánu by mělo být nanejvýš deset položek, aby se dal dokončit během dvou až tří měsíců.

Pro většinu nových procesů a šablon by měl být naplánován zkušební provoz, který se zhodnotí. Pro zkušební provoz by měli být vybráni lidé, kteří jsou proaktivní a mají zájem změny vyzkoušet. Pro interpretování výsledků musí být kvantifikována kritéria, podle kterých se zkušební provoz zhodnotí. Účastníci by měli pravidelně posílat informace o zkušebním provozu. Do zkušebního provozu by mělo být zapojeno co nejvíce lidí, zvýší se tak povědomí o změnách.

Posledním krokem je hodnocení výsledků zlepšovacího cyklu. Hodnocení zajistí neustálé zdokonalování. Nejdůležitější je zjistit, jak nově nasazené procesy vedou k požadovaným cílům. U nových procesů je nutné zohlednit křivku učení. Krátkodobě klesá produktivita, jak je vidět na obr. č. 6.2. Po přivlastnění si procesu organizací produktivita vzroste a návratnosti investice je kladná [21].

Obrázek 6.2: Křivka učení

Metodická znalostní skupina by v prostředí společnosti měla za cíl spravovat metodiku a zlepšovat stávající procesy a postupy. Frekvence schůzek je doporučena na jednou měsíčně. Na každé schůzce byl měl být vytvořen akční plán (viz příloha C), ze kterého je zřejmé, jaké kroky budou učiněny a jaká je stanovená metrika k vyhodnocení úspěšnosti plánu.

Role MZS komunikační direktor má za úkol schůzky plánovat, případně plánovat plynoucí aktivity ze schůzek. Role MZS vůdce má za cíl plány a případné změny obhajovat u generálního ředitele společnosti. Dále reportovat pokrok a výhody plynoucí z vylepšování procesů a metodiky společnosti. Ve spolupráci MZS specialistou by mělo probíhat zlepšování a ověřování procesů na konkrétních projektech. MZS specialista by měl mít povědomí o přístupech softwarového inženýrství a jednotlivých metodikách. V závislosti na charakteristikách konkrétních projektů by měla být metodika přizpůsobena dle potřeb. MZS analytik by se měl v prostředí společnosti zaměřit na používané nástroje - ISZA, Trello, Microsoft Project a další nástroje s podobnými funkčnostmi používanými v softwarovém inženýrství. MZS analytik by měl být schopen doporučit vhodný nástroj v závislosti na používanou metodiku.

K analýze stávající metodiky by měl být využit současný vývojový projekt na implementaci informačního systému. Ověřování nových postupů může probíhat taktéž na tomto projektu či na drobných rozvojích u servisních projektů.

6.3 Řešitelský tým

Jedním z nejvýznamnějších aspektů projektu jsou lidé. Kniha The Rational Unified Process Made Easy od autorů Kroll a Krutchen uvádí, že vývoj softwaru se stal týmovým sportem [14]. Tradičně, mnoho společností má funkční organizaci, analytici, vývojáři a testeři jsou v oddělených skupinách. Ačkoli jsou budována kompetenční centra, nevýhodou je neefektivní komunikace mezi skupinami. Dle autorů Kroll a Krutchen je ještě horší než funkční organizace maticová organizace společnosti, kde zaměstnanec pracuje na několika různých projektech a ve skutečnosti není součástí týmu. Tato organizace vede k tomu, že nejvýznamnější aspekt společnosti – lidé, se stává nenahraditelným.

Funkční organizace jsou přijatelné pro dlouhodobé projekty, kde je aplikován vodopádový model [14]. V případě iterativního vývoje je nezbytná větší soudržnost a komunikace mezi členy týmu. Tým by měl být mezifunkční, obsahující analytiky, vývojáře, architekty, testery a další. Týmu by měly být dodány nástroje pro efektivní spolupráci. Členové týmu by měli mít postoj: "Budu dělat to, co je potřeba k získání k vysoce kvalitního softwaru". Každý člen týmu by měl mít přístup k požadavkům, chybám, testům a dalším. Dále by měly být zredukovány nároky na dokumentaci. Místo toho by měla být umožněna efektivní komunikace tváří v tvář. Tým by měl sdílet zodpovědnost a měl by pracovat společně na naplnění cílů.

Společnost má zavedenou maticovou organizační strukturu. Řešitelský tým je ve společnosti stanoven za účelem operativního řízení projektu. Obvykle je vzhledem k rozsahu projektu stanoveno více řešitelských týmů. Členové jsou specialisté jednotlivých agend. Tým se podílí na analýze i vlastní implementaci.

Zaměstnanci zpravidla pracují ve více řešitelských týmech a na více projektech. Vazba je naznačena na obr. č. 6.3.

Obrázek 6.3: Relace řešitelský tým, pracovník, projekt

Na workshopu byl identifikován problém s alokací kapacit. Při plánování kapacit je nutné zohledňovat, že pracovník spravuje více systému ve více řešitelských týmech. Nástroj ISZA neumožňuje přehledné zobrazení plánovaných prací, vzniká zde obtížná komunikace jak mezi pracovníkem a zadavatelem

úkolu, který stanovuje jednotlivé termíny, tak mezi vedoucími projektů, kteří jsou zodpovědní za alokaci kapacit. Dále byl shledán problém v prioritizaci úkolů napříč projekty.

6.3.1 Návrh řešení - Řešitelský tým

Návrhem je docílit, aby pracovník pracoval v jednom řešitelském týmu. Vazba je znázorněna na obr. č. 6.4. Vzhledem k množství a charakteru projektů společnosti a její organizační struktuře nelze v současné chvíli docílit ideálního stavu, kdy pracovník pracuje v jednom řešitelském týmu na jednom projektu.

Obrázek 6.4: Návrh relace řešitelský tým, pracovník, projekt

V rámci jednoho řešitelského týmu je usnadněna komunikace mezi členy. Tým se může potkávat na pravidelných schůzkách, kde budou řešeny jednotlivé projekty a prioritizovány úkoly. Zlepší se plánování alokace pracovníků a prioritizace úkolů, která již nebude napříč týmy a systémy.

6.4 Míra ceremonie, iterativní / vodopádový přístup

Procesy dle [14] lze srovnat na základě míry ceremonie a použitého přístupu. Ceremonie je chápána jako metodická zátěž (složitost postupů, dokumentační náročnost).

Charakteristické rysy vysoké ceremonie jsou dle [14]:

- důkladné plány,
- mnoho artefaktů souvisejících s řízením, požadavky, designem, testováním,
- dokumenty k řízení, požadavkům, designu, testování jsou detailní a procházejí revizemi,
- je vytvářené a udržované propojení mezi požadavky, designem a testováním,

- je vyžadován schvalovací proces na změny,
- výsledky kontrolních mechanismů jsou pečlivě zaznamenávány a sledovány.

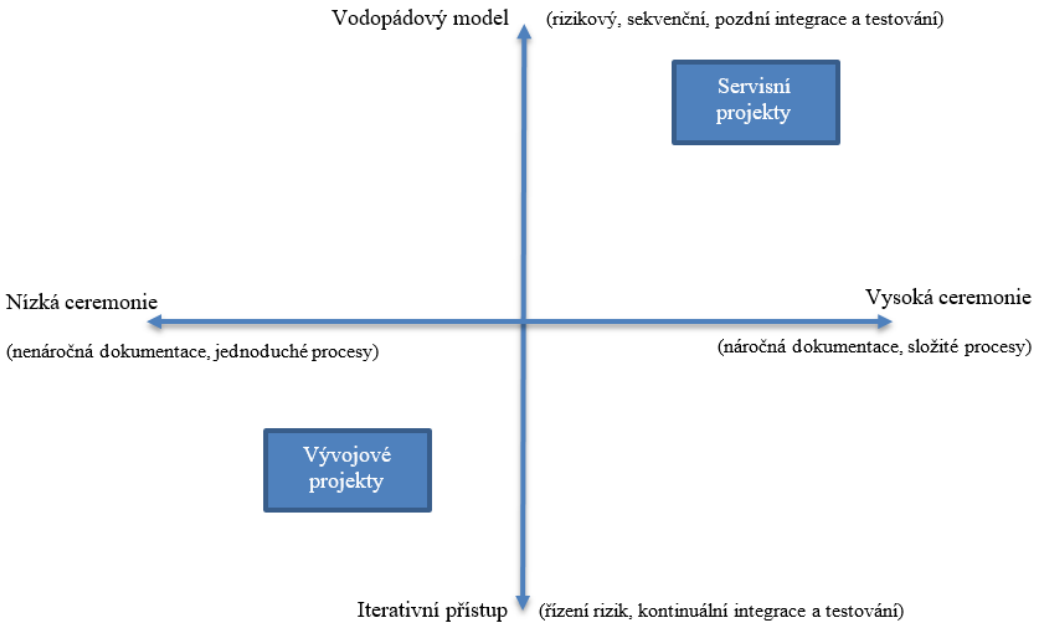
Metodiky s vysokou ceremonií jsou vhodné pro vývoj složitých systémů a pro velké a distribuované týmy [14]. Často je produkovaný systém vysoké kvality, který je snadno udržovatelný. Výrobní náklady jsou typicky vyšší a doba uvedení na trh je pomalejší než u přístupů s nízkou ceremonií.

Metodiky s nízkou ceremonií jsou více flexibilní a mají schopnost se lépe přizpůsobit k vyvíjejícímu se prostředí [14]. Je kladen větší důraz na produkování softwaru než na vytváření rozsáhlé a obsáhlé dokumentace. Výhodou je rychlá reakce na změny, ke kterým během procesů dochází.

Existuje konsenzus mezi představiteli softwarového průmyslu o podpoře přínosů iterativního a rizikově orientovaného vývoje. Iterativní přístup umožňuje lepší předvídatelnost. Včasné zjištění, že se překročí náklady nebo nedodrží harmonogram. Dle [14] inklinují společnosti k vodopádovému modelu kvůli:

- nedostatečnému know-how,
- nekvalitní procesní podpoře,
- nekvalitní podpoře nástrojů.

Na obr. č. 6.5 je znázorněna procesní mapa, ve které jsou zařazeny servisní a vývojové projekty společnosti. Zařazení projektů probíhalo ve spolupráci s ředitelem úseku Metodik a směrnic a odráží charakter současných projektů. Vývojové projekty byly kategorizovány zejména podle současného vývojového projektu, na kterém je uplatňován iterativní přístup. Projekt však není řízen standardně tak, jak popisuje metodika společnosti.



Obrázek 6.5: Procesní mapa servisních a vývojových projektů

Jak je z obr. č. 6.5 zřejmé, na vývojových projektech je implementován iterativní přístup. V Projektovém zákoně či Tabulce činností není iterativní přístup popsán. Nejsou zde uvedeny iterace v jednotlivých fázích projektu ani možné počty iterací. Z rozhovorů vyplynulo, že i přes snahu aplikovat iterativní přístup na vývojovým projektech často dojde k inklinaci k vodopádovému modelu.

Ačkoli jsou servisní a vývojové projekty kategorizovány zcela odlišně, v metodice není tento rozdíl zřejmý. Ve společnosti převažují servisní projekty, a tak jsou metodiky přizpůsobeny spíše jim. V odborné literatuře [5] je monolitický softwarový proces vnímán jako anti-pattern. Jsou zvýšena rizika:

- překročení nákladů,
- předčasné ukončení projekt,
- vývoj nekvalitního produktu,
- technické selhání.

Častým jevem je, že je proces ignorován. Tento jev je zřejmý i ve společnosti. Na vývojových projektech se používají rozdílné nástroje, než definuje metodika. K úkolování jednotlivých pracovníků se na některých projektech

využívá nástroj Trello namísto ISZA. Dále jsou implementovány některé klíčové principy metodiky Scrum (viz kapitola 3.2) jako jsou například *Daily Meetings*.

6.4.1 Návrh řešení - Metodika pro vývojové projekty

Návrhem je nepoužívat monolitický vývojový proces, ale definovat metodiku pro vývojové projekty. V metodikách by se měly definovat iterace dle Rational Unified Process popsaném v kapitole 3.1. Každá iterace by měla obsahovat kroky, jako jsou sběr požadavků, analýza, návrh či programování.

Metodika vývojových projektů by měla být upravena dle charakteristik jednotlivých projektů. Vypracování metodiky pro vývojové projekty by mělo být ve správě metodické znalostní skupiny (viz kapitola 6.2.1).

6.5 Priority řešení

Metodika je dle společnosti chápána jako popis daného procesu. Autor Jiří Voříšek ve své publikaci Informační systémy a jejich řízení popisuje, aby byla metodika efektivně využita při vývoji informačního systému, měla by splňovat základní požadavky, a to zejména [20]:

- musí jasně deklarovat soubor hodnot, na kterých je založena (minimální náklady tvorby, nejkratší možná doba řešení, zahrnutí sociální hlediska a další),
- musí určovat postup řešení, aby bylo možné celý proces vývoje plánovat (čas, lidi, finance, techniku a jiné),
- musí určovat priority řešení,
- měla by doporučovat metody, nástroje a techniky, které je vhodné využít v jednotlivých fází řešení [20].

V metodice Tabulka činností, která je vnímána jako klíčová pro jednotlivé pracovníky řešitelských týmů, nejsou uvedené priority řešení. Pracovníkovi poté není zřejmé, jaké činnosti a události jsou v Tabulce činností klíčové, a jaké výstupy je nutné zpracovat.

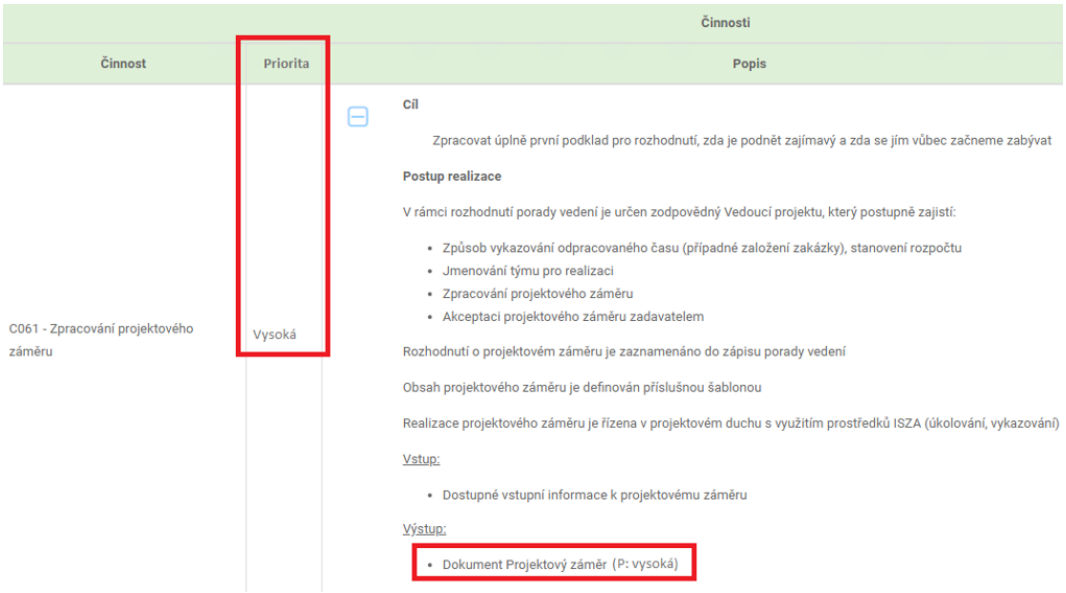
Metodika Tabulka činností předepisuje použití nástroje ISZA. Autor J. Voříšek popisuje, že metodika by měla nástroje pouze doporučovat. V případě, že se společnost rozhodne změnit nástroj ISZA za jiný, bude nutné metodiku Tabulku činností velmi významně změnit.

6.5.1 Návrh řešení - Stanovení priority

Dle [20] by se v metodice měly objevit priority řešení. Doporučení je u jednotlivých výstupů a činností definovat prioritu (viz tabulka č. 6.1, obr. č. 6.1).

Priorita	**Popis**
Vysoká	Výstup je vždy nutné zpracovat.
Střední	Výstup by se měl zpracovat, v případě relevantních důvodů, není nutný.
Nízká	Výstup je doporučeno zpracovat.

Tabulka 6.1: Priority řešení



Obrázek 6.6: Priorita v Tabulce činností

7 Inženýrství úloh vybrané společnosti

V dokumentu Tabulka činností je mimo jiné popsán systém zpracování hlášení. Jedná se o návod, jak zpracovat jednotlivé typy úloh a kdo je za činnosti zodpovědný. V kapitole je popsán systém zpracování hlášení tak, jak uvádí Tabulka činností. Dále je popsáno inženýrství požadavků vycházející z dokumentu Evidence a členění požadavků. V závěru kapitoly je popsán nástroj ISZA určený ke správě úloh.

7.1 Systém zpracování hlášení

Hlášení dle definice společnosti představuje záznam o komunikaci s osobou, která přichází do styku s produkty společnosti. V kontextu této práce je hlášení identické s pojmem úloha. Hlášení je typu: dotaz, požadavek / objednávka vývoje, vada, havárie nebo návrh na zlepšení.

Dle společnosti je systém zpracování hlášení relevantní jak pro vývojové, tak pro servisní projekty. Jak již bylo zmíněno, tak ve společnosti dlouhodobě převládají servisní projekty. V rámci servisních projektů jsou na systémech implementovány nové funkčnosti, které jsou řiditelné níže popsanými postupy. V případě některých velkých vývojových projektů nejsou v praxi postupy uplatňovány. Systém zpracování hlášení popisuje způsob práce v nástroji ISZA. U nového vývojového projektu je využíván pro řízení projektu nástroj Microsoft Project a k úkolování členů týmu nástroj Trello. ISZA slouží pracovníkům pouze k vykazování pracovního času.

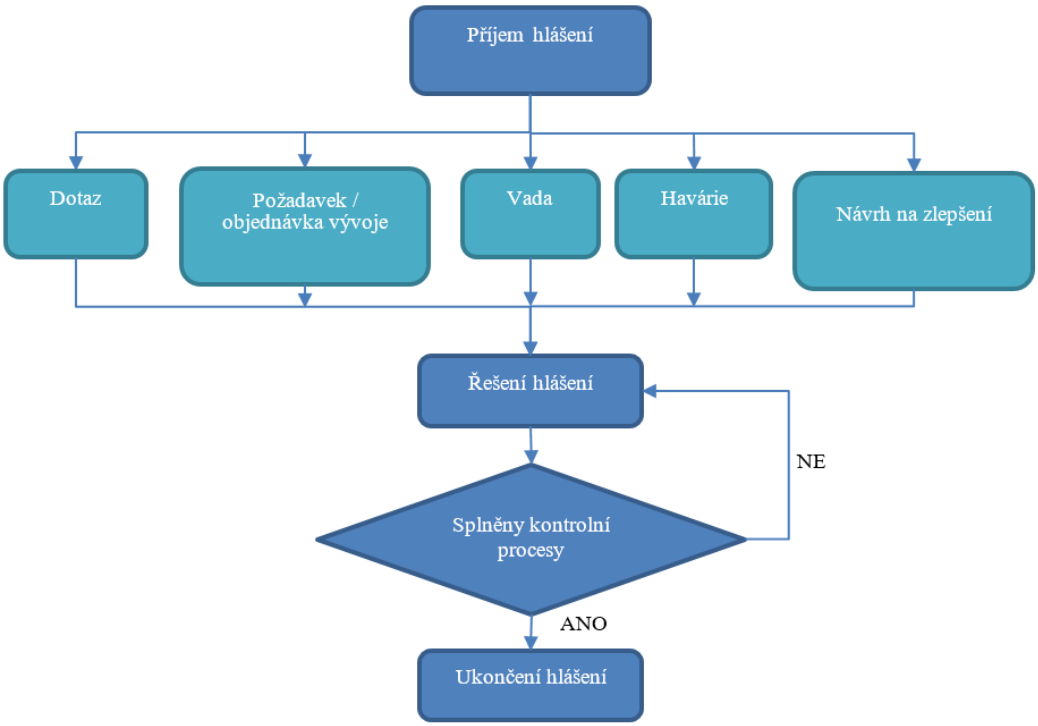
V této kapitole je popsán systém zpracování hlášení tak, jak uvádí Tabulka činností.

Dotaz je hlášení řešící dotaz na ovládání, případně funkčnost aplikace. Požadavek / objednávka vývoje je požadavek na novou funkčnost či změnu funkčnosti. Typ vada je hlášení, které řeší neshodu s platnými návody, standardy, šablonami či požadavky, které jsou platné v rámci společnosti nebo požadované zákazníkem. Jedná se například o chybu programu či nedostatečnou dokumentaci. Hlášení typu havárie řeší stav informačního systému, který znemožňuje užívání aplikace nebo jeho části k účelu, pro který byl vytvořen. Havárii je nutné neprodleně řešit a uvést systém do stavu, který nebrání uživateli aplikaci používat. Návrh na zlepšení je hlášení s požadav-

kem na úpravu systému, které není vázáno smlouvou.

Každý typ hlášení může být dále členěn na druhy dle číselníku druhů hlášení, který je dán smlouvou se zákazníkem. Druhy hlášení jsou specifikovány smlouvou se zákazníkem. Druh hlášení je stanoven terminologií a prostředím zákazníka. Například typ hlášení požadavek / objednávka vývoje má jako druh hlášení "Service request"nebo "Change request".

Zpracování jednotlivých typů hlášení je v Tabulce činností rozčleněno tak, jak je uvedeno na obr. č. 7.1.



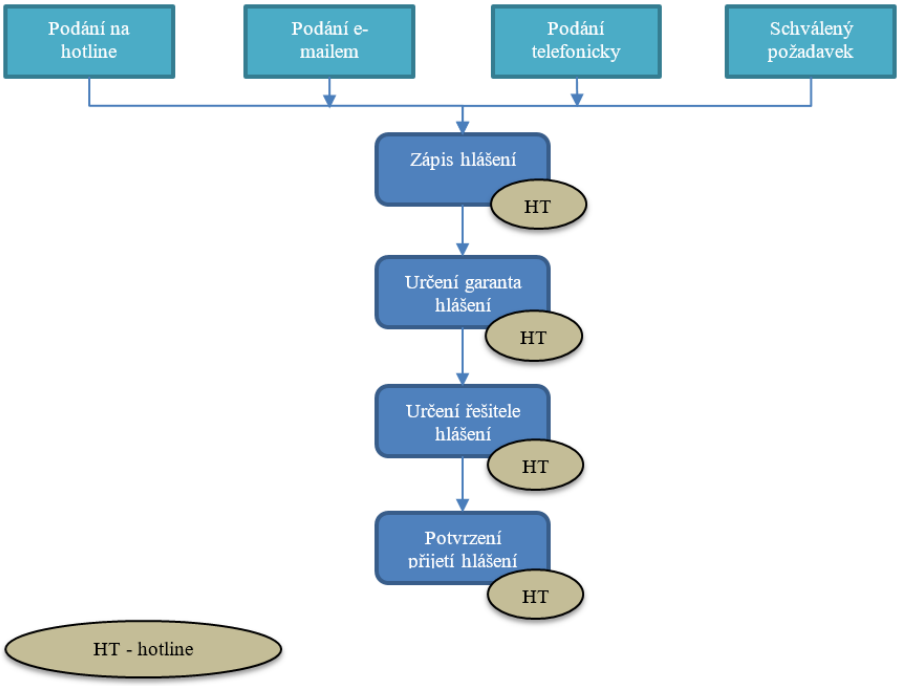
Obrázek 7.1: Schéma systému zpracování hlášení

7.1.1 Příjem hlášení

Hlášení je možné přijmout čtyřmi způsoby: podáním hlášení na hotline, podáním prostřednictvím e-mailu, podáním telefonicky nebo vedoucí projektu obdržel závazně objednaný požadavek.

Proces přijmutí hlášení podané prostřednictvím hotline je znázorněno na obr. č. 7.2. Hlášení je evidováno v nástroji ISZA. Následně se určí garant hlášení, což je osoba zodpovědná za vyřešení hlášení včetně akceptace od zákazníka. Řešitel hlášení je osoba zodpovědná za naplánování a provedení činností, které vedou k vyřešení hlášení v příslušné kvalitě. Řešitel

dle Tabulky činností zakládá Plán projektu. V praxi se Plán projektu zakládá téměř vždy. Poslední činnost je potvrzení přijetí hlášení. Nástroj ISZA automaticky generuje potvrzovací e-mail o přijetí.

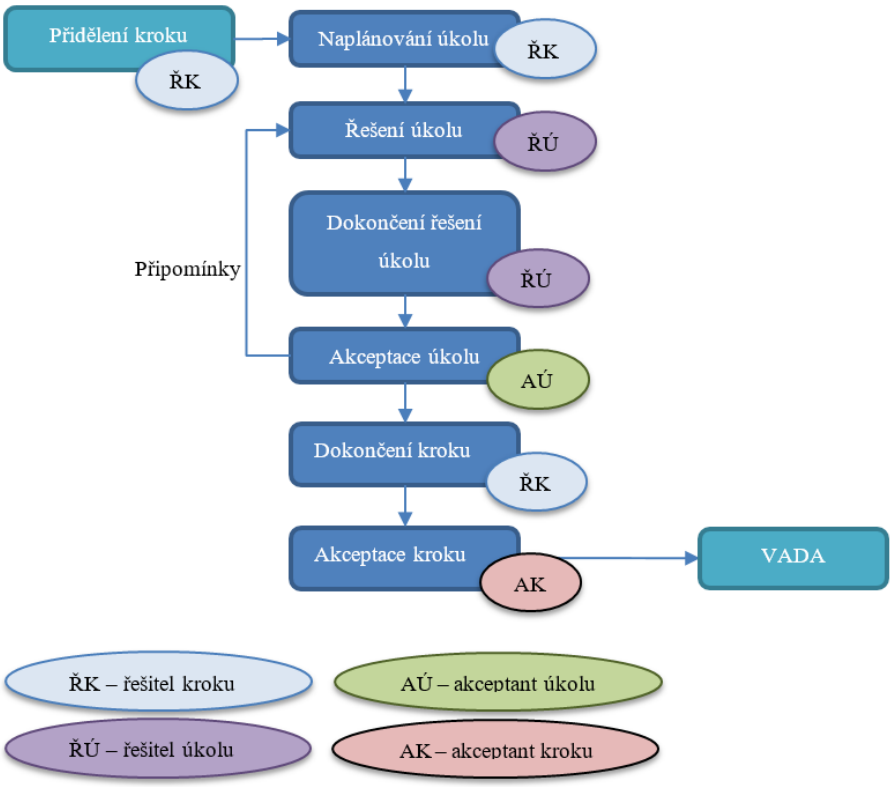


Obrázek 7.2: Schéma příjmu hlášení

7.1.2 Řešení hlášení

Zpracování hlášení je v metodice Tabulka činností popsáno obecně i pro jednotlivé typy hlášení. Obecný proces znázorňuje obr. č. 7.3.

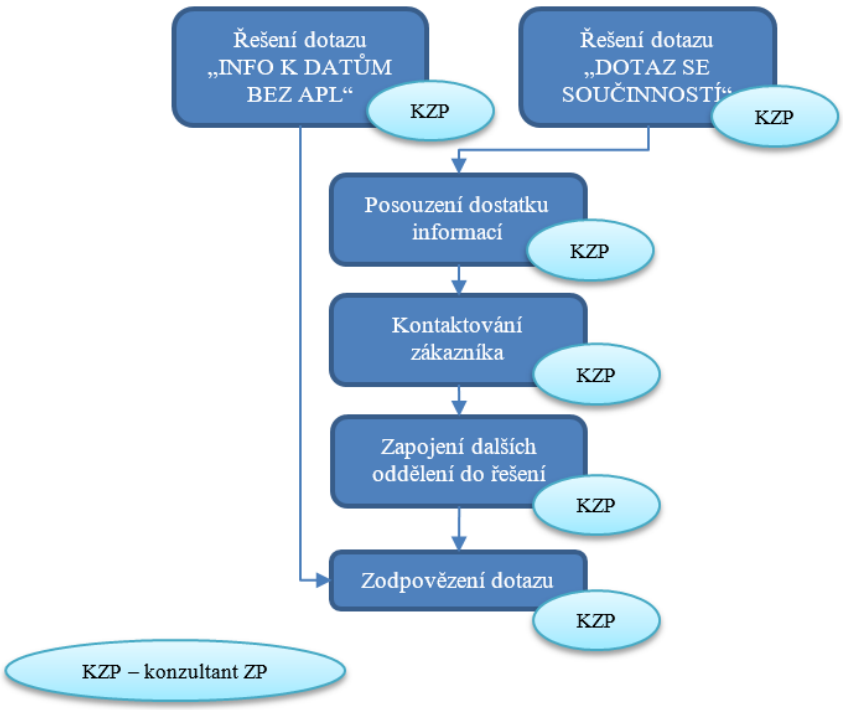
V kroku lze naplánovat úkoly. Řešitel kroku je zodpovědný za správné vypracování všech úkolů. Úkoly jsou zadávány s odhadnutými pracnostmi, s termíny zahájení řešení, doručení a termíny akceptace úkolů. Úkol se zobrazí na pracovním stole řešitele (viz kapitola 7.3.5). Před samotným řešením se na pracovním stole vyznačí termín zahájení řešení. Výstupy jsou připojovány jako přílohy buď přímo k úkolu, nebo k nadřazenému kroku. Následuje činnost Dokončení řešení úkolu, ve které pracovník zkontroluje, zda byly připojeny všechny výstupy, zapíše popis řešení úkolu a vyznačí datum dokončení. Úkol přebírá akceptant, který dle kvality splnění úkol akceptuje či předá zpět k řešení. Pokud je úkol akceptován, řešitel kroku zkontroluje, zda jsou všechny úkoly v kroku vyřešeny a krok předá akceptantovi kroku k akceptaci. Pokud řešení obsahuje vady, akceptant je zaznamená do nástroje ISZA prostřednictvím hlášení typu vada.



Obrázek 7.3: Schéma řešení hlášení

Hlášení typu Dotaz

Proces zpracování hlášení typu Dotaz je znázorněno na obr. č. 7.4. Dotaz je dělen na "Dotaz se součinností", pro jehož řešení je nutná spolupráce třetí strany, a "Dotaz k datům bez aplikace". V případě dotazu se součinností se do hlášení uvádí, že byl zapojen třetí subjekt. Zodpovědný řešitel je konzultant, který komunikuje s dodavatelem. Konzultant nejprve posoudí, zda je k dispozici dostatek informací pro zodpovězení dotazu. Pokud není, kontaktuje zákazníka pro poskytnutí relevantních informací. V případě, je-li nutná konzultace ostatních oddělení, konzultant vytvoří k hlášení úkol a předá jej na kompetentního řešitele. U dotazu k datům bez APL jsou nejčastěji kladeny otázky typu: "Zjistěte mi v jakém stavu je funkčnost X", "Jaká je celková částka na faktuře"a jiné. Jsou-li získané podklady pro odpověď, konzultant odpoví zákazníkovi a vyřeší hlášení.



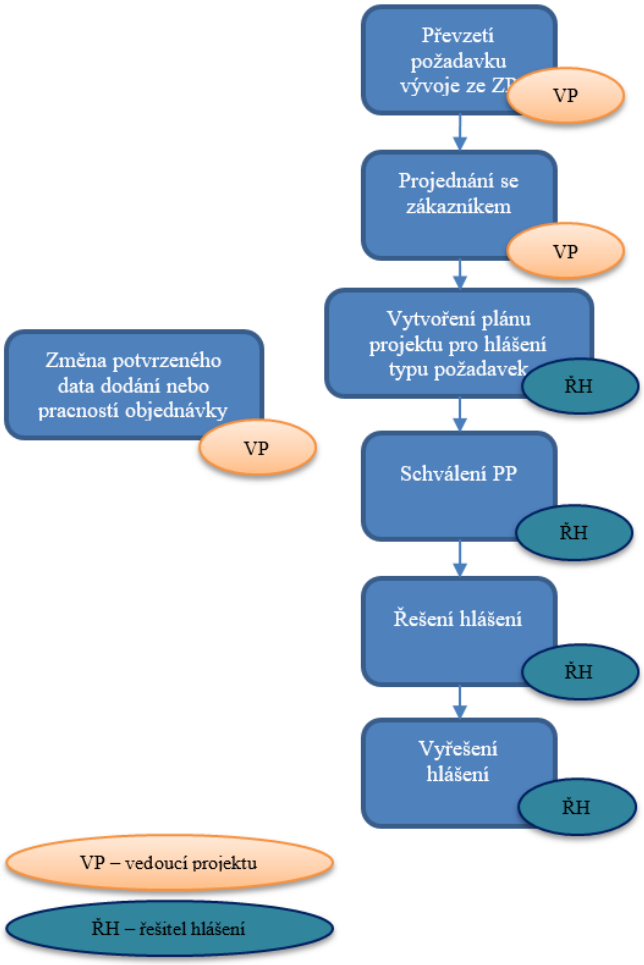
Obrázek 7.4: Schéma řešení hlášení typu Dotaz

Hlášení typu Požadavek / Objednávka vývoje

Proces zpracování hlášení typu Požadavek je znázorněno na obr. č. 7.5. Vedoucí projektu obdrží informaci o hlášení a dle povahy požadavku rozhodne o jeho zpracování nebo jej zařadí do seznamu požadavků k projednání se zákazníkem. Na jednání se zákazníkem se rozhodne o prioritě a termínu zpracování. Následně je vytvořen Plán projektu řešitelem hlášení. V Plánu projektu se dle potřeb nakonfigurují jednotlivé kroky popsané v kapitole 5.3. Pro skupinu požadavků na vývoj lze vytvořit jeden Plán projektu. Hlášení se pak seskupí pod jedno mateřské hlášení. Plán projektu s kapacitami, termíny a personálním obsazením se schvaluje ve schvalovacím režimu, je-li pracnost u požadavku odhadnuta na více než 75 člověkodní. Řešitel hlášení je povinen v nástroji ISZA evidovat každý kontakt se zákazníkem v rozsahu datum, typ a zjednodušený popis komunikace. V případě, že je vyžadována spolupráce na řešení, zadá se formou úkolu na kompetentního pracovníka. Pokud jsou dokončeny všechny kroky a úkoly Plánu projektu, řešitel hlášení zapíše stav k požadavku a uvede datum vyřízení.

Vedoucí projektu má možnost u hlášení změnit potvrzené data dodání či pracnosti. Vedoucí projektu zadá nový termín data dodání, odhadovanou

pracnost a důvod změny. O změně informuje zákazníka a k hlášení připojí danou komunikaci.

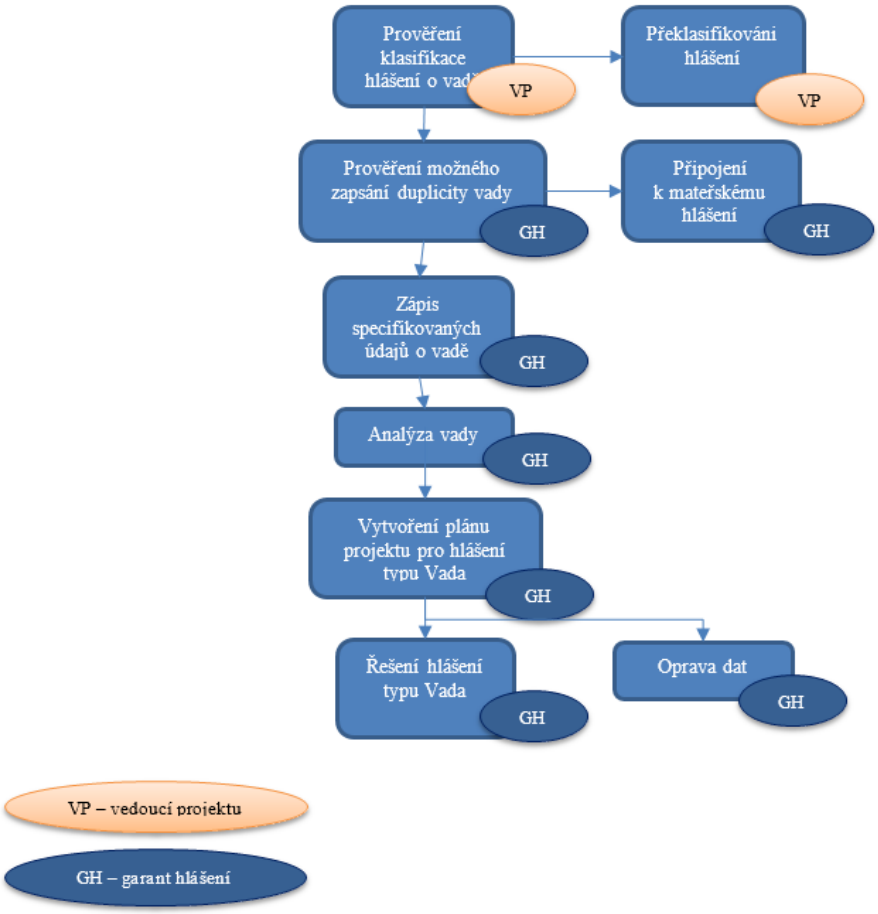


Obrázek 7.5: Schéma řešení hlášení typu Požadavek / Objednávka vývoje

Hlášení typu Vada

Proces zpracování hlášení typu Vada je znázorněno na obr. č. 7.6. Garant hlášení zkontroluje hlášení a posoudí, zda se opravdu jedná o vadu. Pokud se jedná o jiný typ hlášení, po domluvě se zákazníkem se typ hlášení změní. Dále je nezbytné prověřit, zda vada již nebyla nahlášena, pokud ano, hlášení se připojí k již existujícímu. Pokud neexistuje obdobná či stejná vada, garant hlášení se stává zároveň řešitelem. Provede se analýza vady a vytvoří se plán projektu, kde se stanoví priorita, určí se termín distribuce opravy. U jednotlivých kroků projektu se definují řešitelé, pracnosti a termíny zahájení, doručení a akceptace. Následně se vada vyřeší. Jedná-li se o opravu

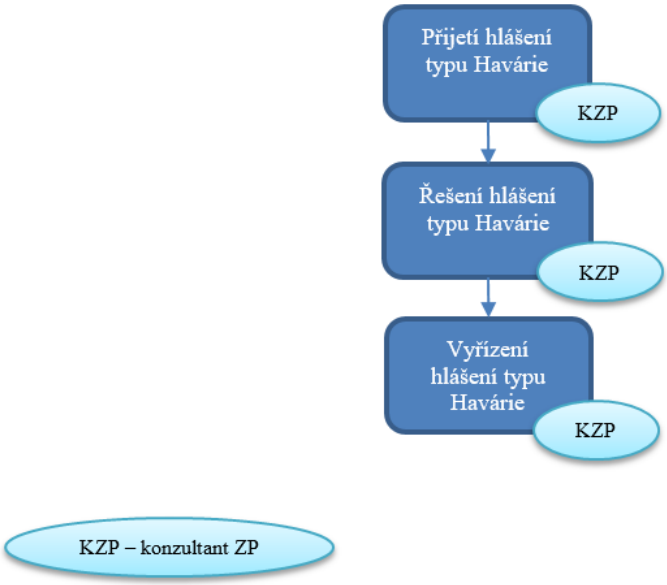
dat, je nutné zohlednit možné příčiny chybných dat. V případě databázového skriptu je nezbytné vyplnit šablony na zásah do databáze.



Obrázek 7.6: Schéma řešení hlášení typu Vada

Hlášení typu Havárie

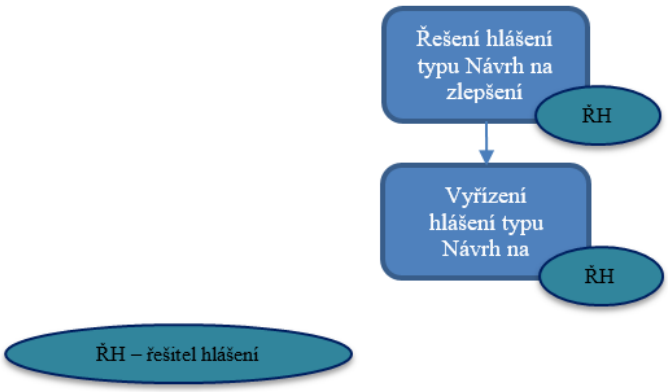
Hlášení přijímá konzultant ZP, který je povinen zjistit maximum informací a aktivně komunikovat se zákazníkem. Havárie je řešena ve spolupráci s hlavním programátorem a vedoucím projektu. Vyřešením havárie je zajištěna zpětná vazba zákazníkovi. Proces je shrnut na obr. č. 7.7.



Obrázek 7.7: Schéma řešení hlášení typu Havárie

Hlášení typu Návrh na zlepšení

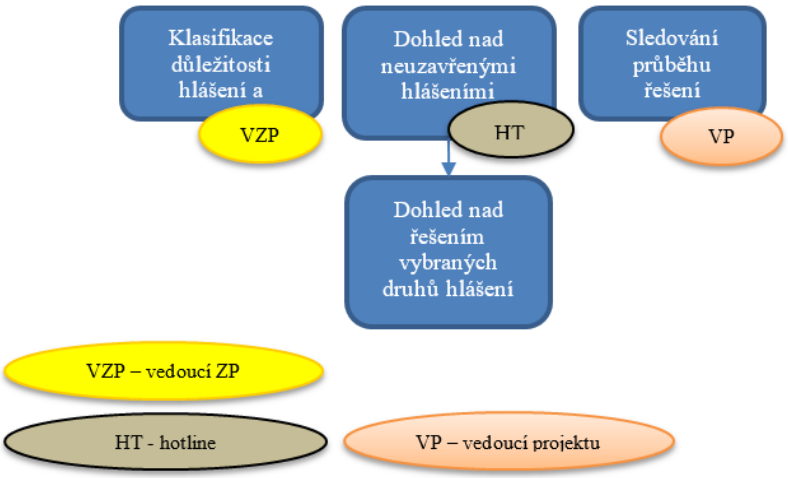
Přijde-li hlášení typu návrh na zlepšení, vedoucí projektu rozhodne, zda se návrh na zlepšení bude realizovat. Pokud se bude implementovat, přiřadí se buď k již existujícímu hlášení, v rámci kterého bude řešeno, nebo se vytvoří plán projektu a dále se postupuje standardně (viz kapitola 7.1.2). Proces je shrnut na obr. č. 7.7.



Obrázek 7.8: Schéma řešení hlášení typu Návrh na zlepšení

7.1.3 Kontrolní proces

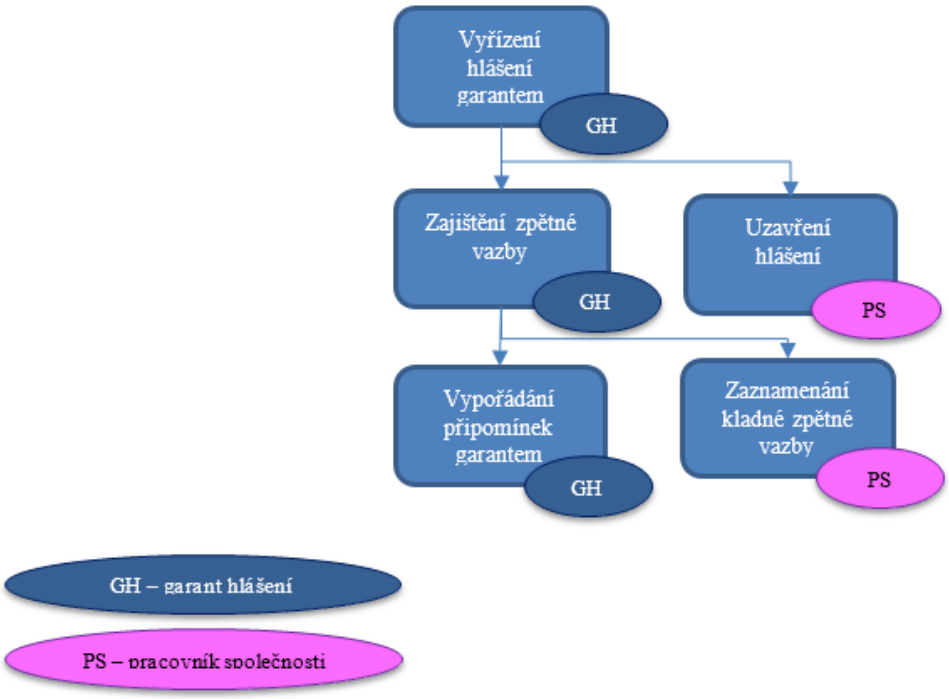
V rámci kontrol řešení hlášení jsou zavedeny kontrolní procesy (viz obr. č. 7.9). Vedoucí oddělení ZP kontroluje na denní bázi přijatá hlášení a přiřazuje priority. Hlášení s vysokou prioritou se v nástroji ISZA označí příznakem "důležité". Hotline provádí dohled nad neuzavřenými hlášeními. Dále se jednou týdně zpracovává seznam nevyřešených hlášení, který je předkládán jednotlivým garantům. Vedoucí projektu je povinen sledovat průběh řešení hlášení typu Havárie.



Obrázek 7.9: Schéma kontrolních procesů u hlášení

7.1.4 Ukončení hlášení

Pokud jsou splněny všechny kroky a úkoly, garant hlášení označí hlášení jako vyřízené a uvede datum. Pokud není stanoveno jinak, automaticky se odesílá zákazníkovi mail s žádostí o vyplnění zpětné vazby. V případě, že zákazník projeví nespokojenost s řešením, hlášení se neuzavře. Je-li zaznamenaná kladná zpětná vazba, pracovník Hotline zaznamená kladnou zpětnou vazbu a hlášení uzavře. Proces je shrnut na obr. č. 7.10.

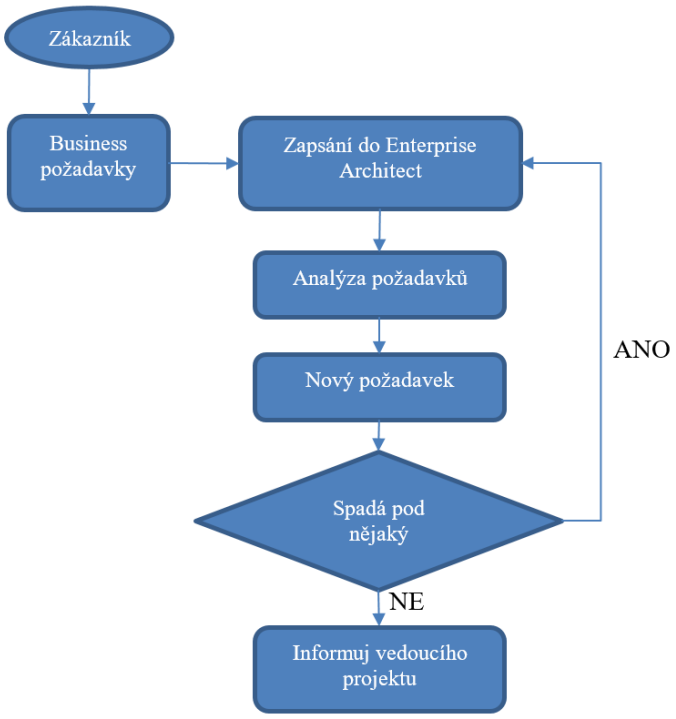


Obrázek 7.10: Schéma ukončení hlášení

7.2 Inženýrství požadavků

Proces evidence a členění požadavků je realizován při plnění kroku Seznam požadavků ve fázi Příprava (viz kapitola 5.3.2). Vývoj i správa požadavků probíhá v nástroji Enterprise Architect, potažmo v nástroji Microsoft Excel.

Proces vývoje požadavků je zobrazen na obr. č. 7.11. Proces je převzatý z metodiky a jak je z obrázku zřejmé, není kompletní.



Obrázek 7.11: Proces vývoje požadavků

7.2.1 Členění požadavků

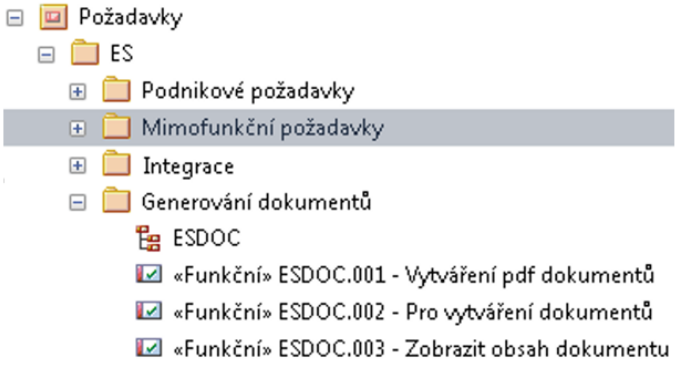
Požadavky jsou členěny na:

- business – dle Wiegerse jsou to podnikatelské požadavky (viz kapitola 4), stanovují účely projektu,
- funkční – definují funkce systému,
- nefunkční – definice obecných parametrů (požadavky na bezpečnost, rychlost, vzhled, rozsah, dostupnost a jiné).

Nefunkční požadavky jsou dále členěny na:

- architektonické – požadavky na architekturu systému,
- implementační – požadavky na způsob zavedení systému (školení),
- legislativní – požadavky vyplývající ze zákonů, norem,
- bezpečnostní – požadavky na bezpečnost systému

Požadavky jsou hierarchicky členěny do struktury podle subsystémů a modulů. Příklad členění v nástroji Enterprise Architect je uveden na obr. č. 7.12.



Obrázek 7.12: Členění požadavků v Enterprise Architect

7.2.2 Číslování požadavků

Každý požadavek je označen kódem a pořadovým číslem v rámci kódu a jednoznačným názvem. Příklad je uveden v následující tabulce č. 7.1.

Kód	**Popis**
ES	kód systému
DOC	kód subsystému
ESDOC001	kód modulu

Tabulka 7.1: Příklad číslování požadavků

7.2.3 Stavy požadavků

Stavy požadavků jsou psány v tabulce č. 7.2.

Kód	**Popis**
Nový	Nový požadavek, zatím nezačalo zpracování
K zapracování	Požadavek byl schválen k zapracování
Analyzováno	Požadavek byl analyzován, vznikl návrh
Implementováno	Požadavek byl zapracován
Ověřeno	Zpracování požadavku bylo ověřeno
Zastaveno	Zastavený nebo zrušený požadavek
Duplicita	Duplicitní požadavek, bude řešen jinde

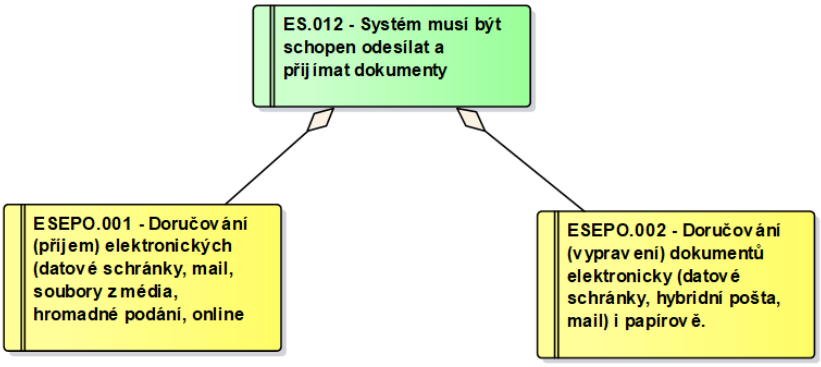
Tabulka 7.2: Stavy požadavků

7.2.4 Pokrytí požadavků, kontrola

Během procesu evidence požadavků jsou definovány aktivity:

- **pořizování požadavků** – profese AP, analýza a návrh požadavků, pokrytí požadavků funkcemi,
- **plánování realizace požadavků do výroby** – profese VP, založení požadavků jako hlášení v nástroji ISZA,
- **kontrola plnění požadavků** – profese VP, kontrola plnění v reportu Report pokrytí požadavků.

Pokrytím požadavku se dle dokumentace rozumí vyznačení způsobu jeho vyřešení v evidenci. Požadavek může být pokryt buď hlášením, nebo jinými požadavky. Hlášení znamená, že je naplánovaná realizace a řízení je realizováno standardním postupem popsaným v kapitole 7.1. Pokud je požadavek pokryt jiným požadavkem, znamená to, že se požadavek rozpadá. Například se business požadavek rozpadá na několik funkčních požadavků a pokrytí je zřejmé. V nástroji Enterprise Architect se rozpad naznačuje vazbou Aggregate (viz obr. č. 7.13). Nadřazený požadavek je splněn v případě, že jsou splněny všechny požadavky podřízené.



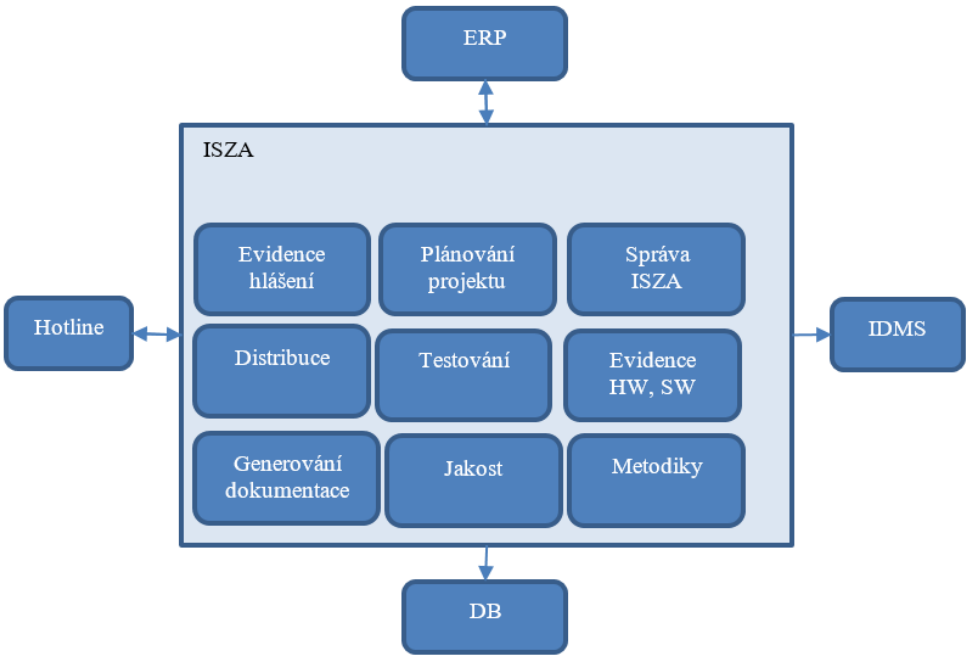
Obrázek 7.13: Agregace požadavků v Enterprise Architect

Report pokrytí požadavků je předkládán vedoucím projektu na vyžádání ředitele společnosti. Nad dokumentem s požadavky je připravené makro, kterým je vyhodnoceno, zda jsou požadavky pokryty hlášeními v nástroji ISZA.

7.3 Nástroj určený pro správu úloh

Automatizovaná správa úloh je ve společnosti realizována prostřednictvím nástroje ISZA (viz kapitola 7.1). Nástroj byl implementován v devadesátých letech vývojáři firmy. Mezi hlavní funkce, které nástroj poskytuje je

evidence a správa hlášení, organizace a správa úkolů pracovníka, plánování projektů a další. Blokové schéma nástroje je uvedeno na obr. č. 7.14. Nástroj je v současné chvíli přepisován do webového rozhraní, které je uživatelsky přívětivější. Vzhledem k nedokončené implementaci je v kapitole popsána desktopová verze aplikace.



Obrázek 7.14: Moduly nástroje ISZA

Nástroj ISZA eviduje a spravuje hlášení (více v kapitole 7.1). Dále se pomocí nástroje spravují Plány projektů, definují se kroky a úkoly. Správa ISZA je modul obsahující pomocné programy, které zajišťují provoz nástroje. V modulu Distribuce se evidují verze softwaru, které jsou předávány zákazníkovi. Jedná se o jednoduchý systém na správu verzí. Modul Testování slouží k řízení testování, evidenci plánů a jednotlivých testovacích případů. Evidence HW a SW slouží pro technické pracovníky k získání přehledu o hardware a software používaný zaměstnanci společnosti. Dále jsou v modulu evidované prostředí zákazníků, na kterých jsou produkty společnosti spravovány. Modul Generování dokumentace je již zastaralý a není v současné době aktivně využívaný. V modulu Jakost jsou definovány otázky na jednotlivé výstupní dokumenty z Tabulky činností. Otázky slouží pracovníkovi jako návod, co všechno by měl výstupní dokument obsahovat. Otázky jsou zároveň určeny akceptantovi kroku či úkolu k zajištění řádné kontroly (viz kapitola 7.1.3). Modul Metodiky umožňuje editaci metodik.

Nástroj je propojený s webovým rozhraním zákaznické linky. Hlášení evidované prostřednictvím webového formuláře je v nástroji ISZA automaticky zaznamenáno. Automaticky jsou také evidované emaily příslušného formátu zaslané na adresu zákaznické linky.

Systém ERP obsahuje moduly mimo jiné sloužící k vykazování pracovního času zaměstnanců. Pracovní čas lze vykazovat prostřednictvím ISZA. Data jsou se systémem ERP synchronizována. ISZA je dále propojen s projektovým úložištěm IDMS. V případě, jsou-li v nástroji ISZA nahrané přílohy, automaticky se ukládají do projektového prostoru.

7.3.1 Zápis hlášení v nástroji ISZA

Hlášení lze v nástroji zadat několika způsoby:

- přímým zápise na formuláři *HAP001F - Evidence hlášení*,
- zápisem přes tlačítko *Nové hlášení* na formuláři *HAP001F - Evidence hlášení*,
- zápisem přes tlačítko *Nová vada* z formuláře *HAP001F - Evidence hlášení* nebo přes tlačítko *Vada* z jiných formulářů,
- vygenerováním hlášení z formuláře *HAP008F - Hlášení z WWW*,
- vygenerováním hlášení z formuláře *HAP009F - Evidence hotline hovorů*,
- zapsáním hlášení zákazníkem přes webový formulář v aplikaci Helpdesk,
- automatickým zpracováním emailu s formátovaným předmětem zprávy.

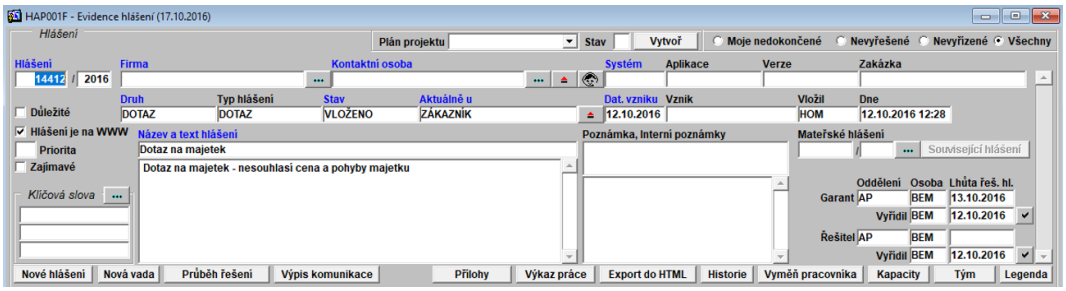
Formulář HAP001F - Evidence hlášení

K evidenci hlášení skrze formulář *HAP001F - Evidence hlášení* je nezbytné vyplnit povinná a volitelná pole:

- *Hlášení* – vyplněno automaticky,
- *Firma* – povinné pole, obsahuje rozbalovací seznam se zákazníky, které jsou evidované v nástroji,
- *Kontaktní osoba* – povinné pole, obsahuje rozbalovací seznam s osobami, které jsou evidované v nástroji,

- *Systém* – povinné pole, obsahuje rozbalovací seznam se systémy, které jsou evidované v nástroji,
- *Aplikace* – volitelné pole, obsahuje rozbalovací seznam s aplikacemi, které jsou evidované v nástroji ke spravovanému systému zákazníka,
- *Zakázka* – volitelní pole, obsahuje rozbalovací seznam se zakázkami, které jsou evidované v nástroji,
- *Druh* – povinné pole, obsahuje rozbalovací seznam s druhy hlášení,
- *Stav* – vyplněno automaticky,
- *Aktuálně* – vyplněno automaticky,
- *Důležité* – vyplněno automaticky v případě, že zákazník má příznak *Důležitý zákazník*, jinak volitelné pole definující prioritu hlášení,
- *Hlášení je na WWW* – volitelné pole, v případě zaškrtnutí se hlášení zobrazuje v aplikaci Helpdesk a automaticky se generují notifikace o založení, změně a vyřešení hlášení zákazníkovi,
- *Zajímavé* – volitelné pole, v případě zaškrtnutí nástroj uživateli umožní rychlé vyhledání hlášení,
- *Klíčová slova* – volitelné pole, obsahuje rozbalovací seznam s klíčovými slovy hlášení,
- *Oddělení* – vyplněno automaticky na základě hodnot v polích Systém a Druh,
- *Osoba* – vyplněno automaticky na základě hodnot v polích Systém a Druh,
- *Garant* – vyplněno automaticky na základě hodnot v polích Systém a Druh,
- *Řešitel* – vyplněno automaticky na základě hodnot v polích Systém a Druh,
- *Lhůta řešení hlášení* – povinné pole vyplněno dle druhu hlášení.

Formuláře je zobrazen na obr. č. 7.15.

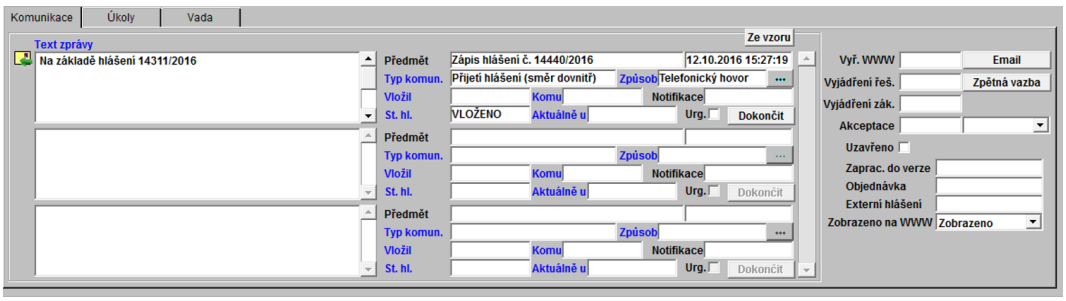


Obrázek 7.15: Formulář - Evidence hlášení

Po vyplnění hlavičky hlášení se vyplní první záznam v záložce Komunikace (viz obr. č. 7.16). Vyplňují se pole:

- *Text zprávy* – povinné pole, vložení textu hlášení,
- *Předmět* – vyplněno automaticky,
- *Typ komunikace* – vyplněno automaticky,
- *Způsob* – povinné pole, obsahuje rozbalovací seznam se způsoby přijetí hlášení,
- *Vložil* – vyplněno automaticky,
- *Komu* – vyplněno automaticky,
- *Notifikace* – volitelné pole, vyplňují se emailové adresy, na které jsou automaticky zasílány notifikace,
- *Stav hlášení* – vyplněno automaticky,
- *Aktuálně u* – vyplněno automaticky.

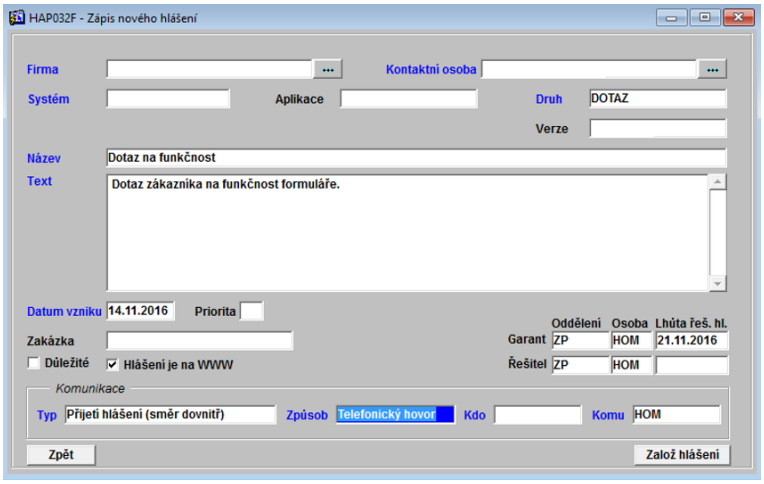
K hlášení je umožněno přes tlačítko *Přílohy* vkládat přílohy. Přes tlačítko *Výkaz práce* lze vykázat čas strávený nad řešením hlášení.



Obrázek 7.16: Formulář - Komunikace

Formulář HAP001F – Nové hlášení

Zjednodušený zápis hlášení je umožněn přes tlačítko *Nové hlášení*, které je na formuláři *HAP001F - Evidence hlášení* (obr. č. 7.17).

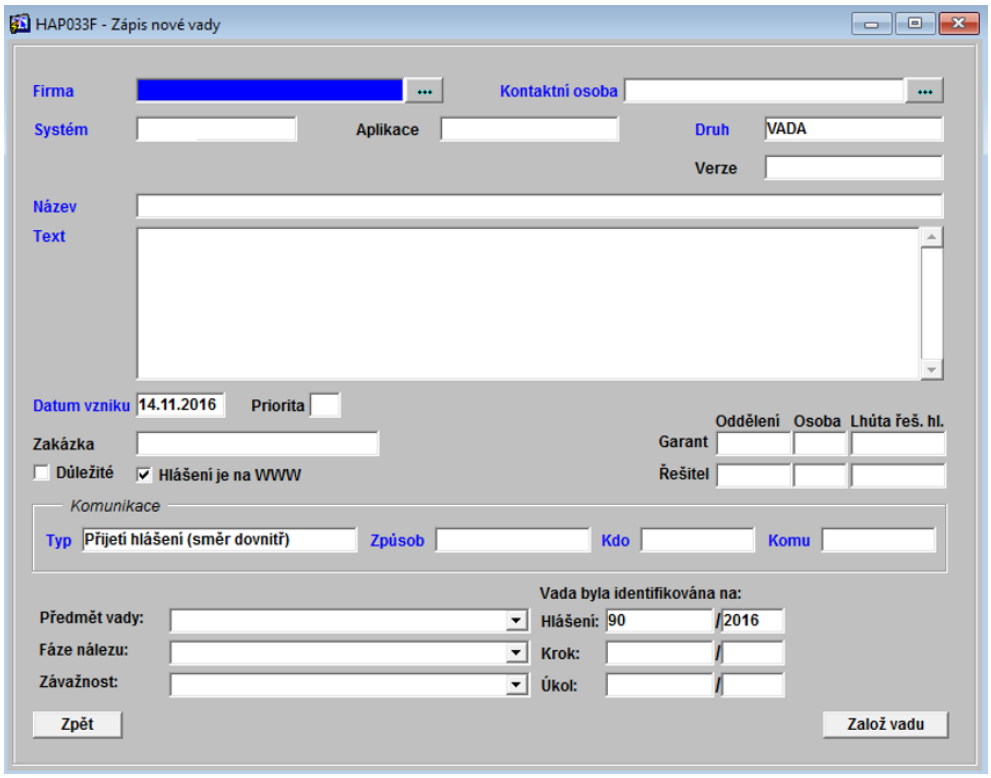


Obrázek 7.17: Formulář HAP001F – Nové hlášení

Po vyplnění povinných polí a stisku tlačítka *Založ hlášení* se hlášení objeví na standardním formuláři *HAP001F - Evidence hlášení*.

Formulář HAP001F – Nová vada

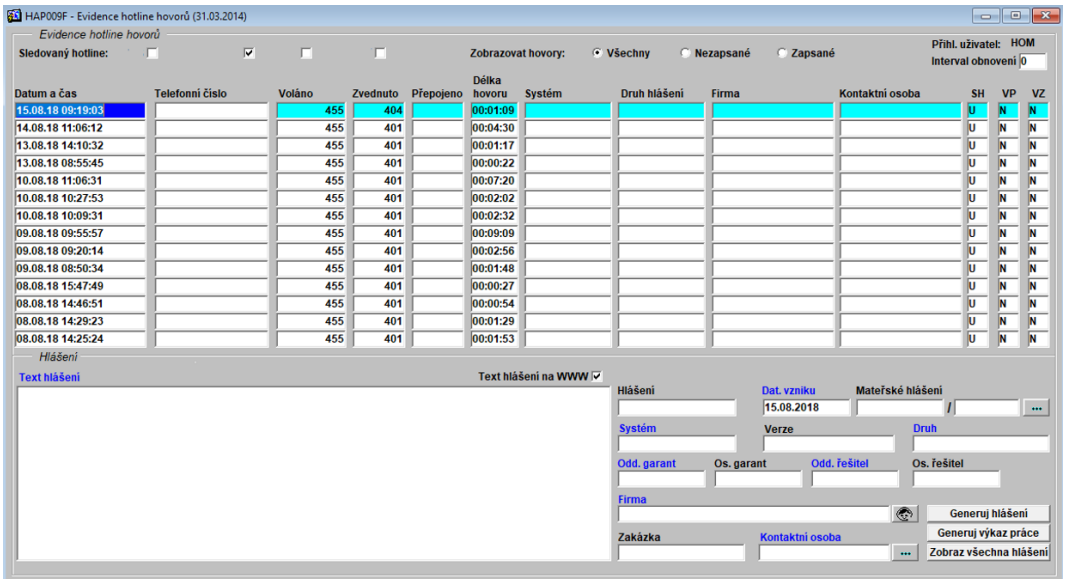
Zjednodušený zápis hlášení typu Vada je umožněno přes tlačítko *Nová vada* na formuláři *HAP001F - Evidence hlášení* (obr. č. 7.18). Je-li na formuláři *HAP001F - Evidence hlášení* vybrán záznam s hlášením a následně stisknuto tlačítko *Nová vada*, některé pole jsou při zápisu vady předvyplněné.



Obrázek 7.18: Formulář HAP033F – Nová vada

Formulář HAP0009F – Evidence hotline hovorů

Telefonické hovory, které jsou přijímány zákaznickou linkou, jsou evidovány na formuláři *HAP009F - Evidence hotline hovoru* (viz obr. č. 7.19). Ze záznamů je umožněno založit hlášení stiskem tlačítka *Generuj hlášení*.



Obrázek 7.19: Formulář HAP0009F – Evidence hotline hovorů

Zápis hlášení přes aplikaci Helpdesk

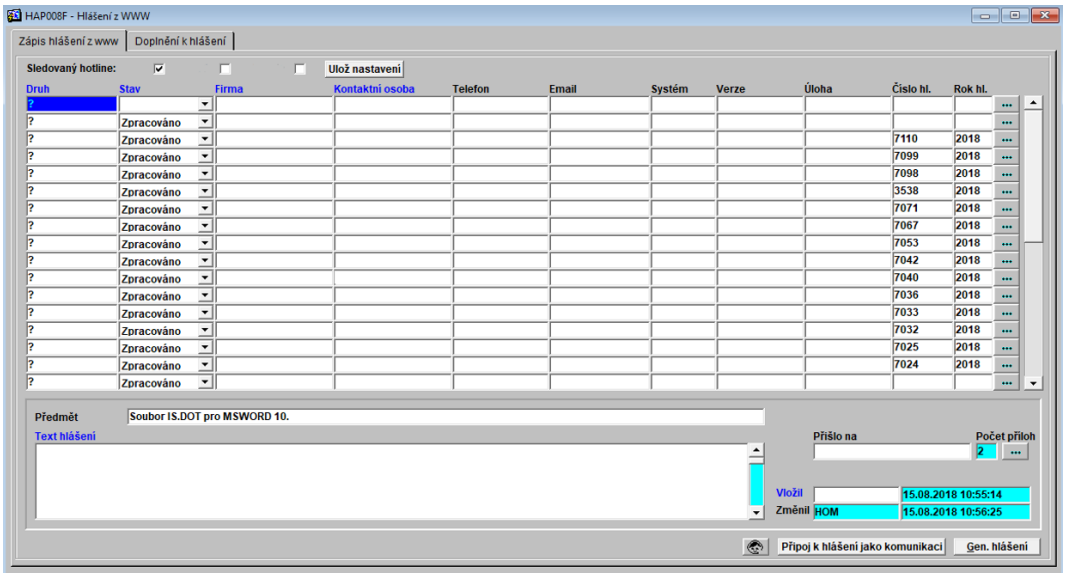
Zákazník má možnost zapsat hlášení přes aplikaci Helpdesk. Nástroj ISZA automaticky vygeneruje nové hlášení. O vytvoření je garant hlášení notifikován prostřednictvím emailu. Hlášení v nástroji ISZA je nutné editovat, aby splňovalo příslušný formát.

Zápis hlášení přes email

Hlášení v nástroji ISZA lze evidovat prostřednictvím zaslání emailu na adresu zákaznické linky společnosti. Podmínkou je správně vyplněný formát. Nástroj ISZA následně automaticky hlášení zaznamená.

Formulář HAP0008F – Hlášení z WWW

Nástroj ISZA eviduje záznamy o hlášeních, které jsou evidovány prostřednictvím zaslání emailu na zákaznickou linku společnosti, ale nesplňují definovaný formát (obr. č. 7.20). Pracovníci zákaznické linky poté doplní nezbytné informace a zadají hlášení.

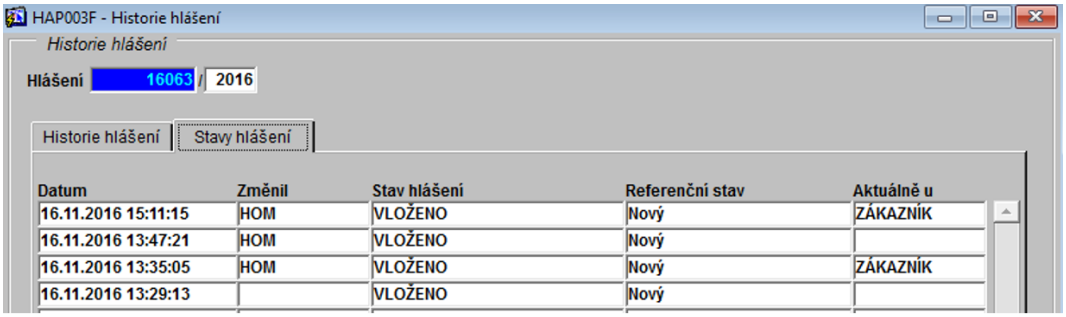


Obrázek 7.20: Formulář HAP0008F – Hlášení z WWW

Z formuláře je umožněno prostřednictvím tlačítka *Gen. hlášení* vygenerovat nové hlášení, kde jsou automaticky vyplněny některé hodnoty.

7.3.2 Stav hlášení

U jednotlivých hlášení jsou chronologicky zobrazeny stavy (viz obr. č. 7.21). Změna stavu se provádí přímo v hlavičce hlášení změnou hodnoty polí *Stav* a *Aktuálně u*.



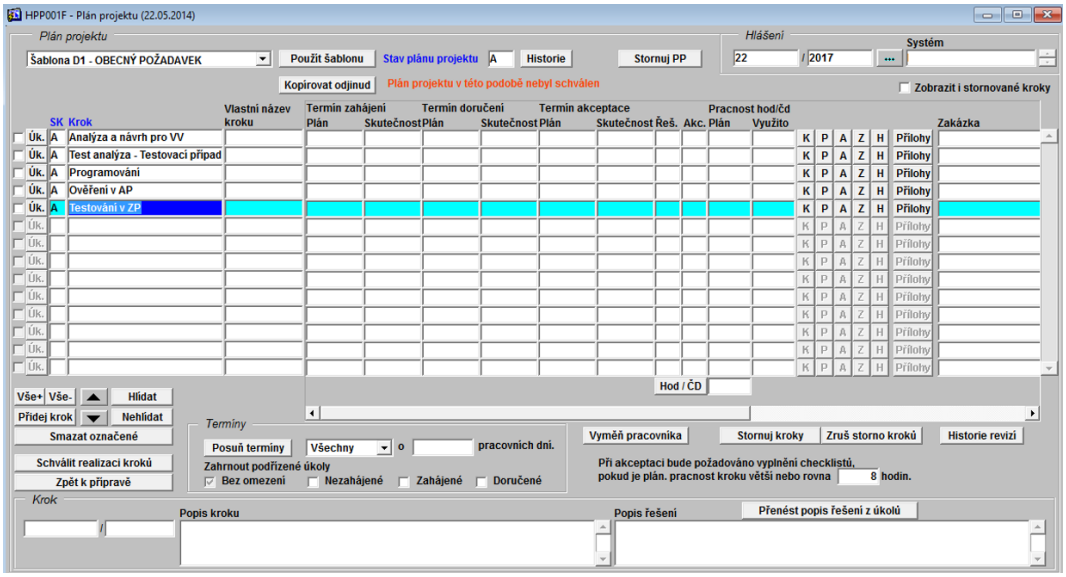
Obrázek 7.21: Formulář HAP0003F – Historie hlášení

Dále je umožněno zobrazit historii k hlášení pod záložkou *Historie*.

7.3.3 Založení Plánu projektu

Plán projektu se vytváří z formuláře *HAP001F - Evidence hlášení* stiskem tlačítka *Vytvoř*. Otevře se formulář *HPP001F - Plán projektu*, kde se kroky

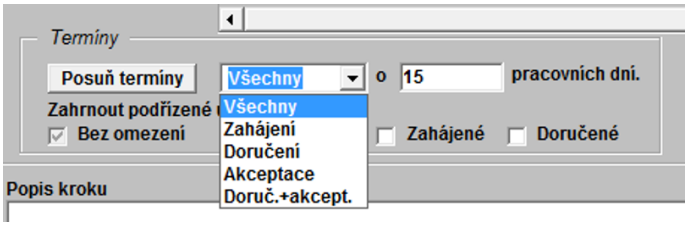
plánu zadávají buď ručně, nebo nahráním šablony (viz obr. č. 7.22).



Obrázek 7.22: Formulář HPP001F – Plán projektu

K dokončení plánu projektu je nutné doplnit u jednotlivých kroků plánované termíny zahájení, doručení, akceptace, řešitele, akceptanty a plánovanou pracnost.

Plán projektu je umožněno zkopírovat z již existujícího hlášení prostřednictvím tlačítka *Kopírovat odjinud*. Naplánované termíny lze posunout o definovaný počet pracovních dní. Obrazovka na posun termínů je na obr. č. 7.23.



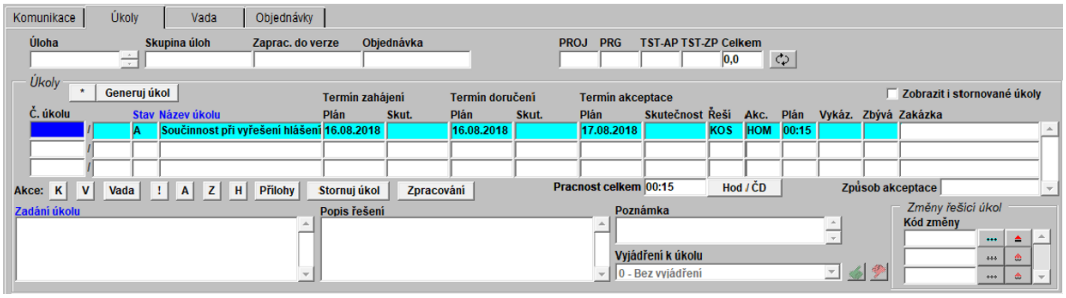
Obrázek 7.23: ISZA: Přeplánování projektů

7.3.4 Definice úkolů

K jednotlivým krokům v plánu projektu se evidují úkoly na pracovníky. U úkolu se eviduje:

- *Číslo úkolu* – vyplněno automaticky,

- *Stav* – povinné pole, úkol nabývá stavů: A-příprava, B-předán ke zpracování, C-zahájena práce na úkolu, D-úkol je vyřešen, E-úkol je akceptován, F-úkol je stornován
- *Termín zahájení* – povinná pole, vyplňuje se plánovaný a skutečný termín zahájení
- *Termín doručení* – povinná pole, vyplňuje se plánovaný a skutečný termín doručení,
- *Termín akceptace* – povinná pole, vyplňuje se plánovaný a skutečný termín akceptace,
- *Řeší* – povinné pole, řešitel úkolu,
- *Akc.* – povinné pole, akceptant úkolu,
- *Plán* – povinné pole, plánovaný čas na řešení,
- *Vykáz.* – vyplněno automaticky v případě zadání odpracovaného času nad úkolem,
- *Zbývá* – vyplněno automaticky, zbývající čas, který lze strávit nad řešením úkolu,
- *Zakázka* – vyplněno automaticky.



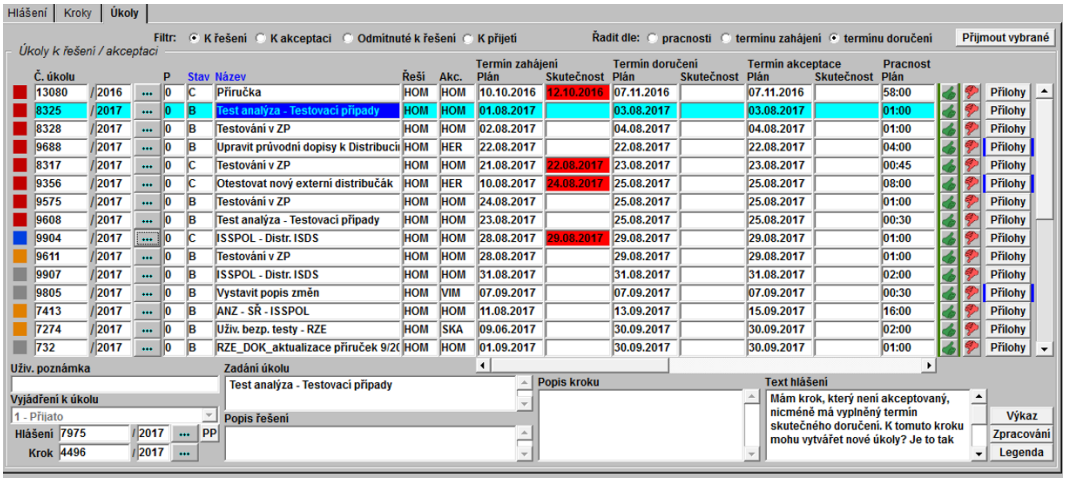
Obrázek 7.24: ISZA: Úkoly

Seznam úkolů pod jedním krokem je možné sledovat zobrazit na formuláři *HPP002F - Seznam úkolu jednotlivých kroku projektu.*

7.3.5 Pracovní stůl

Pracovní stůl obsahuje přehled hlášení, kroků a úkolů přiřazených zaměstnanci. Nástroj umožňuje zobrazit pracovní stůl podřízeného pracovníka. Na záložce *Hlášení* jsou evidovány hlášení, ke kterým je zaměstnanec přiřazen jako řešitel nebo garant. Zobrazení lze uzpůsobit dle zvoleného filtru.

Pod záložkou *Kroky* jsou zaznamenány kroky, u kterých je zaměstnanec řešitelem, případně akceptantem. Záložka *Úkoly* se sestává z úkolů, u kterých je zaměstnanec řešitelem nebo akceptantem (viz obr. č. 7.25).



Obrázek 7.25: ISZA: Pracovní stůl

7.3.6 Shrnutí funkčností nástroje ISZA

Funkčnosti nástroje jsou shrnuty v následující tabulce č. 7.3. Funkčnosti jsou totožné jako při srovnání nástrojů v kapitole 4.2.4.

Funkčnost	**ISZA**
Definice stavu úloh	ano
Sledování stavů úloh	ano
Definice vývojového procesu úlohy	ano
Přikládání příloh	ano
Evidence historie úlohy	ano
Nastavení emailových notifikací	ano
Definice přehledů a statistik	ano
Kopírování úloh	obtížné
Sledování SLA	ano
Stanovení časového odhadu k úloze	ano
Sledování odpracovaného času u úlohy	ano
Reportování (excel a jiné)	ano
Integrace na GIT	ne
Integrace na testování	ano
Full-text vyhledávání	částečně
Vytváření filtrů	ano
Vytváření pracovních ploch	ne
Řízení přístupu	ano
Použití v mobilní aplikaci	ne
Sledování a řízení projektu	částečně
Plánování verzí	ano

Tabulka 7.3: Shrnutí funkčností nástroje ISZA

Kopírování hlášení je v nástroji obtížné. Hlášení lze kopírovat standardní funkčností dané technologie, která je uživatelsky nepřívětivá. Pomocí funkčnosti vyhledávání lze vyhledat název a text hlášení, text komunikace a obsah přiložených dokumentů. Sledování a řízení projektů je efektivní pouze u servisních projektů. Společnost pro řízení vývojových projektů využívá nástroj Microsoft Project.

8 Analýza a návrh úprav správy úloh

Kritická analýza poznatků byla provedena studiem příruček a procesů nástroje ISZA (viz kapitola 4.2). Analýza opět probíhala ve spolupráci se středním managementem a s dvěma řešitelskými týmy. Postup analýzy je popsán v kapitole 6.1.

Nástroj ISZA neumožňuje flexibilní spolupráci v rámci projektů. Byla identifikována slabá místa nástroje, která jsou vypsána v následujících podkapitolách 8.1 až 8.6. Návrh řešení je popsán v kapitole 8.8.

8.1 Administrativní zátěž – plán projektů

Nástroj ISZA umožňuje zadat plán projektu, jak je popsáno v kapitole 7.3.3. K jednotlivým krokům plánu projektu se musí zadávat plánované termíny zahájení, doručení a akceptace. U každého kroku se tedy vyplňují tři plánované termíny. Nejčastěji má plán projektu pět kroků, a to: Analýza a návrh pro VV, Test analýza – Testovací případ, Programování, Ověření v AP a Testování v ZP. Při založení standardního plánu projektu je tedy nutné vyplnit patnáct plánovaných dat.

Nastávají situace, kdy je nutné termíny přeplánovat. Nástroj umožňuje posun pouze o definovaný počet pracovních dní. Při posouvání není možné zobrazit kalendář. Přeplánování se tak stává obtížnější.

Je-li nutné posunout některé termíny zahájení a doručení jednotlivých kroků, musí uživatel termíny přepisovat ručně. Přeplánování je velmi administrativně náročné. Z rozhovoru s vedoucím projektu vyplynulo, že je velmi pracné udržovat termíny hlášení aktuální. Do dat plánů je zanášen šum. Šum se projeví i v úkolech a pracovníci nemají relevantní informace na pracovním stole.

8.2 Plánování alokací

Nástroj ISZA neumožňuje efektivní plánování alokací jednotlivých pracovníků na projektech. Plánování zdrojů probíhá skrze nástroj u jednotlivých kroků, potažmo úkolů, které mají stanoveného řešitele, estimaci a plánované termíny zahájení, dokončení a akceptace. Vzhledem k zanesenému šumu,

který vzniká problémem popsaným v předchozí podkapitole 8.1, jsou plány nepřesné. Pracovní stůl zaměstnance, který zobrazuje frontu úkolů k řešení s přiřazenými prioritami, nemusí obsahovat aktuální data. Plánování zdrojů se tak stává velmi obtížné.

Nástroj neumožňuje přehledně zobrazit pracovníka s naplánovanou prací na jednotlivých úkolech. Dále chybí funkčnost zobrazení estimace požadavků v čase.

8.3 Grafické uživatelské rozhraní

Nástroj byl vyvinut v roce 1997. Obsahuje složité, zastaralé formuláře. Na workshopech se zástupci jednotlivých pozic bylo dosaženo konsensu, že pro nově nastoupivše zaměstnance je velmi obtížné naučit se s nástrojem ISZA pracovat. I přes absolvování náročného školení není uživatel plně seznámen s funkcemi nástroje. Pro uživatele je k dispozici uživatelská příručka i dokument Uživatelská příručka - ovládání, kde jsou shrnuty osvědčené postupy práce s nástrojem. Příručky však nezajistí rychlou orientaci v nástroji a efektivní a intuitivní práci.

Byla stanovena hodnotící kritéria uživatelské přívětivosti nástroje ISZA. Autorem práce společně s uživatelm nástroje byla uživatelská přívětivost zhodnocena. Jednotlivá hodnotící kritéria jsou následující:

- jednoznačnost – rozhraní je bez textových či grafických dvojznačností,
- přehlednost – rozhraní působí přehledně,
- stručnost – je zobrazen pouze nezbytný text,
- podobnost – rozhraní jsou podobné předchozím zkušenostem uživatelů,
- intuitivnost – chování aplikace je logické a předvídatelné,
- estetičnost – rozhraní je vizuálně příjemné,
- efektivita – rozhraní zvyšuje produktivitu svým designem či zkratkami,
- nápověda – rozhraní má dostupnou nápovědu k chování aplikace.

Tabulka č. 8.1 uvádí hodnocení stanovených kritérií následovně:

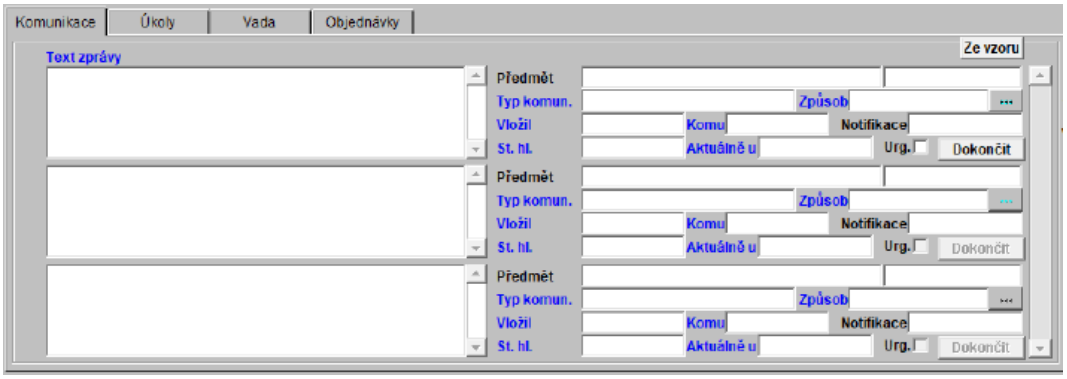
- 1 – uživatelsky přívětivé,
- 2 – uživatelsky přívětivé s drobnými nedostatky,

- 3 – uživatelsky nepřívětivé,
- 4 – uživatelsky nepřívětivé s velkými nedostatky.

Stanovená kritéria	**ISZA**
Jednoznačnost	3
Přehlednost	4
Stručnost	3
Podobnost	1
Intuitivnost	3
Estetičnost	4
Efektivita	1
Nápověda	2
Průměr	**2,6**

Tabulka 8.1: Hodnocení uživatelské přívětivosti nástroje ISZA

Nástroj obsahuje textové a grafické dvojznačnosti, příkladem je formulář *HAP001F - Evidence hlášení*, kde je v záložce komunikace třikrát zobrazeno pole pro vložení zprávy (viz. obr. č. 8.1). Formuláře poté působí velmi nepřehledně a uživatel se ztrácí v tom, jaké pole vyplnit.



Obrázek 8.1: HAP001F - Evidence hlášení, záložka komunikace

Na formulářích jsou zobrazeny zbytečně dlouhé formulace. Příkladem je název pole *Lhůta řeš. hl. (Lhůta řešení hlášení)*, které by se dalo nahradit pojmem *Termín*. Další příklad je název pole *St. hl. (Stav hlášení)*. Výstižnější by byl název *Stav*.

Podobnost je hodnocena kladně, jelikož pracovníci mají s formuláři zkušenosti z projektů, kde je použita stejná technologie. Nástroj je hodnocen jako neintuitivní, na formulářích je změť polí a textu. Uspořádání formuláře

uživateli nenapovídá, jak má být vyplněn. Rozhraní působí zastarale a je hodnoceno jako vizuálně nepříjemné. Rozhraní zvyšuje produktivitu zkratkami, které umožňuje daná technologie. Pokud je uživatel v systému znalý, dokáže pracovat velmi pohotově. Nápověda k chování aplikace je dostupná v příručkách, které jsou uživatelům k dispozici. Příručky dobře popisují jednotlivá pole a funkčnosti nástroje.

Z tabulky vyplývá, že nástroj ISZA je hodnocen spíše jako uživatelsky nepřívětivý. Ke stejnému závěru došly i řešitelské týmy, se kterými byl problém diskutován.

8.4 Nastavení notifikací

Jednou z funkčností nástroje je posílání emailových notifikací pracovníkům. Na schůzkách s uživateli byl identifikován problém v nastavení notifikací. Uživatelům není umožněno nastavit notifikace dle preferencí. Například byl dotazován vedoucí projektu, kterému současné nastavení vyhovuje, protože dle notifikací sleduje aktivitu nad hlášeními. Oproti tomu dotázaný vývojář e-mailové notifikace zcela ignoruje. Převážná většina notifikací zaslaných vývojáři je irelevantní.

8.5 Integrace nástroje s okolními systémy

Rozšíření či změny nástroje jsou implementovány zaměstnanci společnosti. Vzhledem k rychle se vyvíjejícímu prostředí společnosti nedokáže nástroj změny dostatečně rychle reflektovat.

V současné době není nástroj integrován s verzovacím systémem GIT, který je využíván pro správu verzí. Případná integrace zajistí, pokud je úkol řešen úpravou zdrojového kódu aplikace, že se daný commit automaticky propojí s úkolem. Integrace s nástrojem GIT je ve frontě požadavků na nástroj ISZA.

8.6 Pracovní plochy, kolaborativní systém

Nástroj ISZA neumožňuje zobrazovat nástěnky jako například nástroj JIRA. Jednotlivé úkoly, kroky, hlášení či projekty nelze v nástroji přehledně zobrazit. Pomocí nástěnek lze sledovat aktivitu na projektech napříč týmy. Týmu je tak poskytován důležitý přehled o celém dění na projektech.

Na základě rozhovorů s vedoucím výroby by byla uvítána funkčnost, která by umožnila členům týmu zobrazit všechna hlášení na projektech, jejich

řešitele, stavy a další atributy.

8.7 Závažnost identifikovaných problémů

Problémy, jež byly identifikované, jsou v následující tabulce 8.2 seřazeny dle míry závažnosti. Závažnost byla stanovena na základě rozhovorů a výstupů z rozhovorů.

Problém	**Závažnost**
Administrativní zátěž - plán projektů	vysoká
Plánování alokací	vysoká
Grafické uživatelské rozhraní	střední
Nastavení notifikací	střední
Integrace nástroje s okolními systémy	nízká
Pracovní plochy, kolaborativní systém	vysoká

Tabulka 8.2: Identifikované problémy a jejich závažnost

Z tabulky je zřejmé, že problémy: administrativní zátěž – plán projektů, plánování alokací a pracovní plochy jsou závažné a je nezbytné je řešit.

8.8 Návrh řešení – nástroj JIRA Software

Návrhem na řešení všech problémů popsaných výše je nahrazení nástrojem ISZA za nástroj JIRA Software popsaném v kapitole 4.2.1. I přes to, že je nástroj ISZA přepisován do webového rozhraní, je nezbytné brát v potaz náklady spojené s přepisem a také náklady obětované příležitosti. Dále je potřebné zohlednit náklady na další rozšíření či změny nástroje. Požadavky, které byly na nástroj v rámci práce sesbírány, jsou následující:

- integrace s verzovacím systémem,
- zjednodušení plánování projektů,
- nastavení notifikací,
- zobrazení pracovních ploch,
- zobrazení alokací pracovníka,
- jednodušší kopírování hlášení,
- integrace webového rozhraní na modul testování,

- zpřehlednění pracovního stolu – na první pohled viditelné nové úkoly a úkoly s vysokou prioritou,
- podpora agilního řízení nástrojem (kanban nástěnka a další),
- doplnění návazností kroků a úkolů.

Funkčnosti, kterými disponuje nástroj ISZA, disponuje i nástroj JIRA Software společně s doplňujícími aplikacemi. V tabulce č. 8.3 je uvedena cena nástroje JIRA Software s hostingem na hardwaru společnosti (typ licence Server) pro padesát uživatelů. Jsou zde uvedené i další aplikace, které obsahují funkčnosti řešící problémy výše a také pokrývají funkčnosti ISZA.

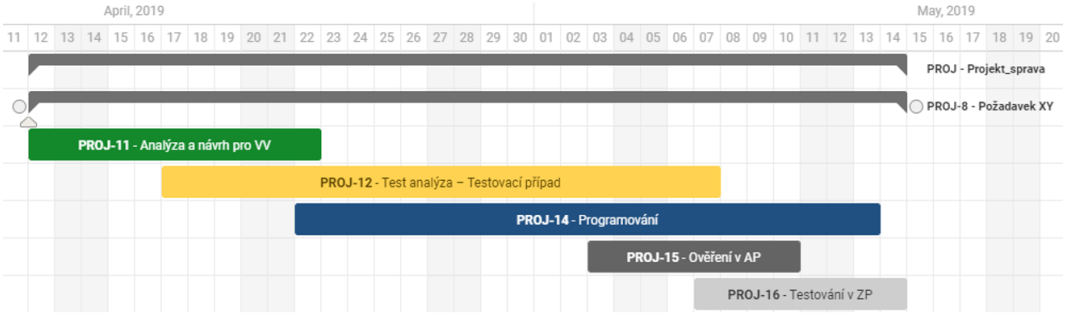
Název aplikace	**Cena pro 50 uživatelů**
JIRA Software	103 268 Kč (4 500 $)
Git Integration for JIRA	17 550 Kč (765 $)
BigPicture Project Management and PPM	20 646 Kč (900 $)

Tabulka 8.3: Cena nástroje JIRA Sotware s rozšířeními pro 50 uživatelů

Zakoupení nástroje JIRA Software s doplňujícími aplikacemi je vyčísleno na 141 464 Kč. Jedná se o jednorázovou investici. V ceně je zahrnuta také dvanáctiměsíční údržba (podpora a aktualizace verzí).

8.8.1 Plán projektů

Návrhem řešení problému popsaného v kapitole 8.1, je využití pomocné aplikace BigPicture - Project Management and PPM [3]. V aplikaci je možné vytvořit Gantt diagram a efektivně tak plánovat jednotlivé kroky plánu projektu. Vzhledem k nesmírným možnostem nastavení nástroje JIRA Software lze u jednotlivých kroků definovat termíny zahájení, doručení a akceptace. Doporučením je stanovit pouze termín vyřešení. Na následujícím obrázku č. 8.2 je znázorněn ukázkový Gantt diagram. Je zde vidět požadavek "PROJ-8 – Požadavek XY", který obsahuje jednotlivé kroky plánu projektu.



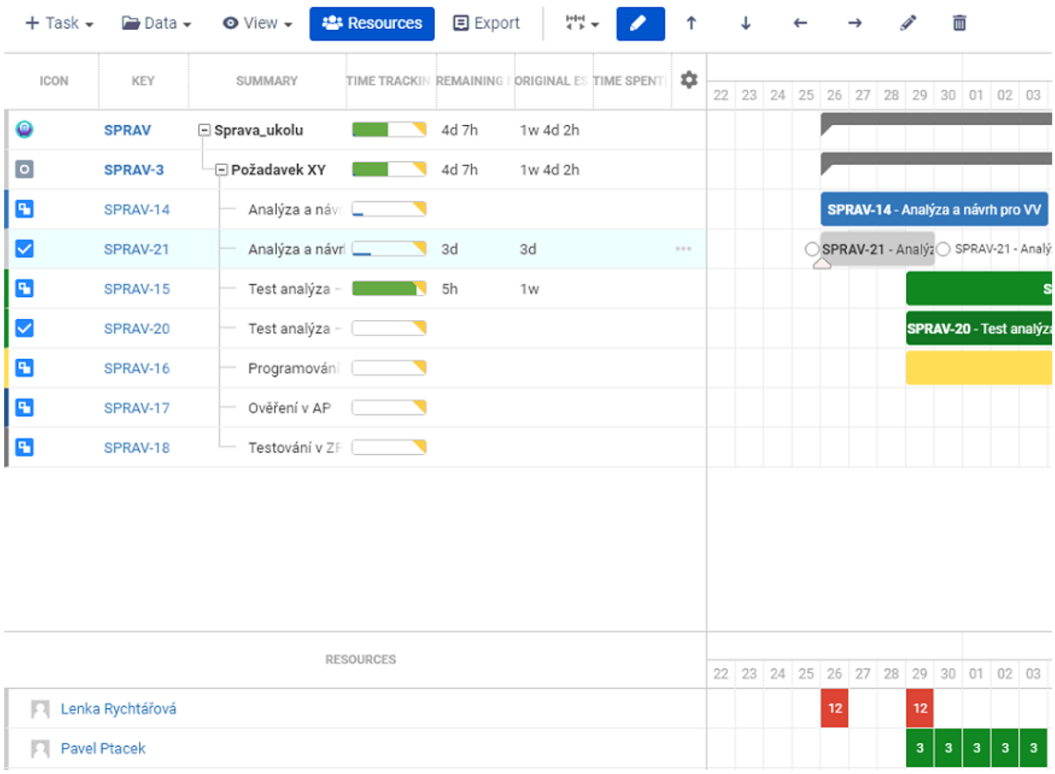
Obrázek 8.2: BigPicture - Project Management and PPM: Gantt diagram

Kroky plánu projektu nejsou v nástroji JIRA generovány automaticky. Vytváření jednotlivých kroků tak může vést opět k administrativní zátěži. Doporučením je vytvářet plány projektů u požadavků, které přesáhnou pracnost 75 člověkodní. Jedná se o požadavky, u kterých plán projektu prochází procesem schvalování. U požadavků, které mají menší pracnost, by vytvoření plánu projektu bylo na řešiteli požadavku. Dalším možným řešením je zakoupit aplikaci na Atlassian Marketplace, například Automation for Jira[1] či ScriptRunner for Jira[2], která automatickou generaci kroků zajistí.

8.8.2 Plánování alokací

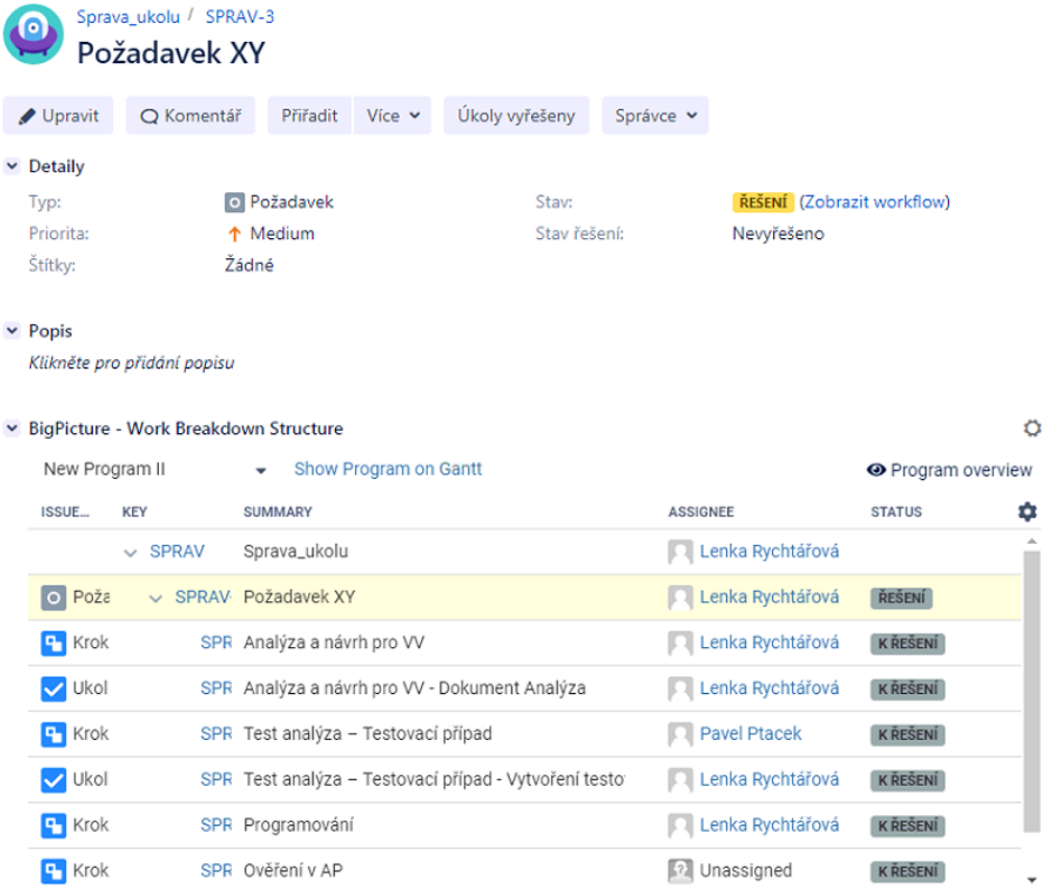
Nástroj umožňuje plánovat lidské zdroje pomocí integrovaných aplikací. Řešení reflektují problém popsaný v kapitole 8.2. Například v aplikaci BigPicture - Project Management and PPM lze u jednotlivých požadavků kromě vytváření Gantt Diagramu plánovat zdroje na projektu [3]. Jak je vidět na obr. č. 8.3, u úloh je stanovený odhad doby řešení. Z obrázku je evidentní, že pro uživatele "Lenka Rychtářová" je na 26. – 29. 4. 2019 naplánováno více práce, než je stanovená denní kapacita. Úlohy jsou v Gantt diagramu interaktivní, takže začátek i dokončení úlohy lze různě měnit.

[1] https://marketplace.atlassian.com/apps/1215460/automation-for-jira
[2] https://marketplace.atlassian.com/apps/6820/scriptrunner-for-jira



Obrázek 8.3: BigPicture - Project Management and PPM: Resource Management

Aplikace má navíc k dispozici samostatnou záložku na plánování alokací, kde jsou různé možnosti zobrazení. Pomocí aplikace se vytváření tzv. Programy, do kterých jsou zahrnuty různé projekty a týmy. V rámci jednotlivých programů se poté spravují alokace a Gantt digramy. Aplikace mimo jiné umožňuje zobrazit na konkrétním požadavku WBS (viz. obr. č. 8.4).



Obrázek 8.4: WBS na úloze

Aplikace BigPicture - Project Management and PPM není jediná, která umožňuje plánování zdrojů. Další jsou například Tempo Planner[3] či AIM Resource Management for Jira. Aplikace AIM Resource Management for Jira[4] například umožňuje poskytování kontinuální zpětné vazby na základě analýzy plánované versus reálné práce. Aplikace BigPicture byla vybrána z důvodu, že dokáže vytvářet i Gantt diagramy a WBS.

8.8.3 Grafické uživatelské rozhraní

Nástroj JIRA Software je vydáván s pravidelnými upgrade zhruba jednou měsíčně. Nástroj je velmi intuitivní a velmi dobře dokumentován. Na zhodnocení grafického uživatelského rozhraní nástroje byla použita stejná hodnotící kritéria jako v kapitole 8.3. Hodnocení je v následující tabulce č. 8.4.

[3]https://marketplace.atlassian.com/apps/1211881/tempo-planner
[4]https://marketplace.atlassian.com/apps/1215177/aim-resource-management-for-jira

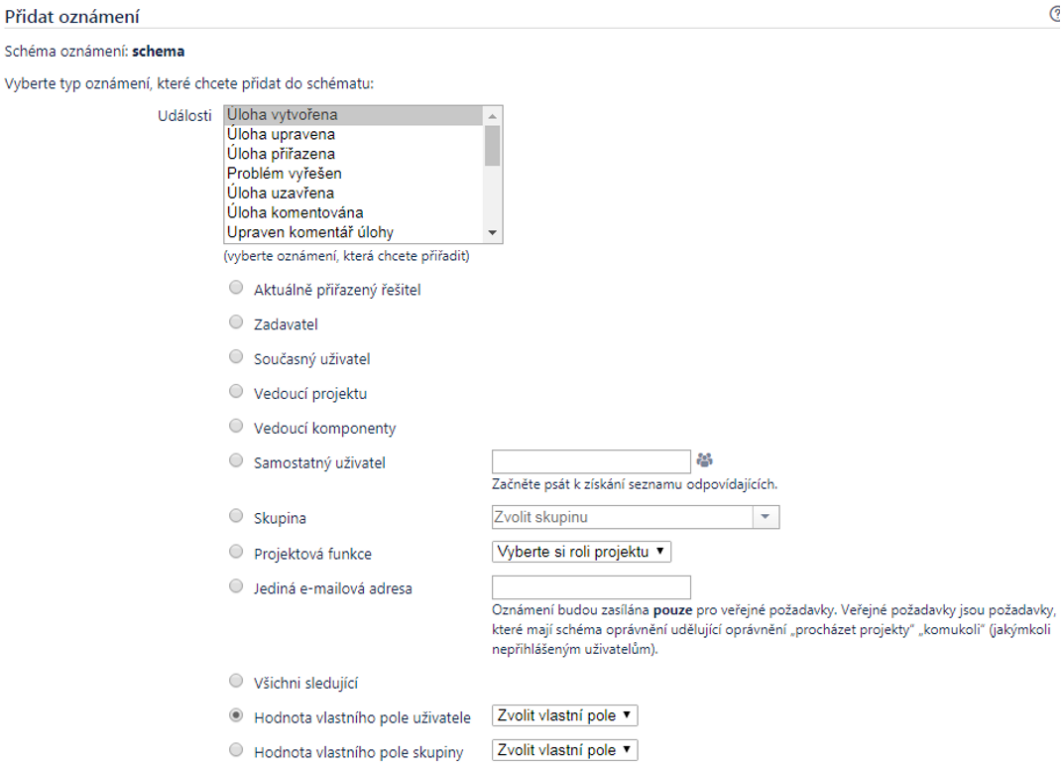
Stanovená kritéria	**ISZA**
Jednoznačnost	1
Přehlednost	1
Stručnost	1
Podobnost	2
Intuitivnost	1
Estetičnost	1
Efektivita	2
Nápověda	1
Průměr	**1,3**

Tabulka 8.4: Hodnocení uživatelské přívětivosti nástroje JIRA

Nástroj JIRA neobsahuje dvojznačnosti, rozhraní působí přehledně a je zobrazen pouze nezbytný text. V rozhraní je nutné se nejprve zorientovat. Z toho důvodu je hodnocena podobnost s drobnými nedostatky. Rozhraní zvyšuje produktivitu svým designem i zkratkami. Negativně je hodnocena editace formulářů, kdy se nejprve načítá formulář. Editace by mohla probíhat přímo na stránce. Nástroj má veřejně dostupnou rozsáhlou dokumentaci. Mimo jiné nabízí možnosti diskuzí a komunit.

8.8.4 Nastavení notifikací

Nástroj JIRA umožňuje nastavení notifikací dle definovaného schématu. Notifikace lze nastavit na události jako například vytvoření úlohy, upravení úlohy, komentování úlohy a další. Obrazovka nástroje s nastavením schématu je zobrazena na obr. č. 8.5.



Obrázek 8.5: Schéma nastavení notifikací

Notifikace můžou být správcem aplikace nastaveny dle jednotlivých preferencí rolí nebo konkrétních uživatelů.

8.8.5 Integrace s okolními systémy

Nástroj JIRA Software lze rozšířit o různé aplikace dostupné na Atlassian Marketplace. Aplikace Git Integration for Jira[5] nabízí integraci s verzovacím systémem GIT. Aplikace nabízí integrace s nástroji určenými ke správě verzí zobrazenými v následující tabulce č. 8.5.

Verzovací nástroj	**Verze**
Fisheye/Crucible	3.3+
Bamboo	5.4+
Bitbucket	4.0+
GitHub	Cloud
GitHub Enterprise	11.10.290+

Tabulka 8.5: Integrace nástroje JIRA s verzovacími systémy

[5]https://marketplace.atlassian.com/apps/4984/git-integration-for-jira

Funkčnosti, které nabízí nástroj JIRA společně s nástrojem Bitbucket:

- automatické zobrazení *branches* souvisejících s úlohou,
- automatické zobrazení *commits* souvisejících s úlohou,
- automatické zobrazení *pull request* souvisejících s úlohou,
- zobrazení *tags* souvisejících s úlohou,
- zobrazení zdrojového kódu v nástroji JIRA,
- vytváření *branches* a *pull request* prostřednictvím nástroje JIRA,
- srovnání *branches* a *tags* v nástroji Bitbucket,
- vkládání komentářů k úloze prostřednictvím nástroje Bitbucket,
- vkládání odpracovaného času k úloze prostřednictvím nástroje Bitbucket,
- sledování progresu v nástroji Bitbucket (*commits*, *diffs*, související úlohy),
- sledování historie změn v nástroji JIRA (autor, změna),
- workflow trigger – automatické přesouvání úlohy do definovaného stavu, například při vytvoření *branche* se úloha přesune z jednoho stavu druhého.

Kompletní dokumentace je uvedena na oficiálních webových stránkách tvůrce integrace nástrojů.

9 Souhrn zjištění a doporučení

Analýzou používaných postupů, dokumentů a na základě zkušeností členů řešitelských týmů a zástupců ze středního managementu vyplynuly problémy popsané v kapitolách 6 a 7. Pro přehlednost jsou v následující tabulce problémy uvedeny a zároveň je stanovena jejich závažnost. Závažnost vyplynula z rozhovorů a pracovních seminářů. Byla stanovena autorem práce.

Problém	**Řešení**	**Závažnost**
Správa metodik	Metodická znalostní skupina	Vysoká
Míra ceremonie, iterativní / vod. přístup	Metodika pro vývojové projekty	Střední
Řešitelský tým (M-1-N)	Řešitelský tým (1-1-N)	Nízká
Priorita řešení	Stanovení priority	Nízká
Administrativní zátěž - plán projektů	JIRA Software	Vysoká
Plánování alokací	JIRA Software	Vysoká
GUI	JIRA Software	Střední
Nastavení notifikací	JIRA Software	Střední
Integrace nástroje s okolními systémy	JIRA Software	Nízká
Pracovní plochy, kolaborativní systém	JIRA Software	Vysoká

Tabulka 9.1: Shrnutí problémů a jejich závažností

Z tabulky je patrné, že v oblasti metodik je nejzávažnější problém Správa metodik. Společnost nemá zavedenou řízenou a formalizovanou správu metodik a zlepšování procesů vývoje software. Řešením je vytvořit metodickou znalostní skupinu, která bude mít správu metodik a zlepšování procesů v kompetenci.

Dalším problémem je relace mezi pracovníkem, řešitelským týmem a projektem. Ve společnosti pracovníci pracují zpravidla na více projektech a ve více řešitelských týmech. Pracovníkovi klesá efektivita, má-li přepínat mezi projekty a také nastávají problémy s alokací kapacit, plánováním a prioritizací úkolů. Řešením je optimalizovat sestavení řešitelských týmů tak, aby

pracovník pracoval v jednom řešitelském týmu. V rámci týmu se zefektivní plánování i prioritizace úkolů.

Z analýzy dále vyplynul problém s dokumentem Tabulka činností, který je spíše orientovaný na servisní projekty. Ačkoli je snaha implementovat iterativní přístup na vývojových projektech, není tento přístup v metodikách popsán. Řešením je vytvořit separátní metodiku určenou pouze pro vývojové projekty.

Posledním problémem, jenž byl identifikován v oblasti metodik, je chybějící priorita jednotlivých činností a výstupů v metodice. Řešením je prioritu definovat.

Nejzávažnější problémy, jež byly identifikované v nástroji ISZA, jsou přeplánovávání plánu projektu, alokace kapacit a pracovní plochy. Problémy způsobují zejména administrativní zátěž. Další problémy jako nastavení notifikací, grafické uživatelské rozhraní či integrace nástroje s okolními systémy mají stanovenou prioritu nižší.

Řešením problémů spojených s nástrojem ISZA je použít nástroj JIRA Software. Nástroj je možné přizpůsobit dle preferencí společnosti a lze do něj integrovat aplikace, které rozšíří jeho stávající funkčnosti.

9.1 Ověření s klíčovými uživateli

Identifikované problémy a návrhy řešení popsané výše byly představeny generálnímu řediteli společnosti, řediteli úseku Metodik a směrnic, řediteli úseku Vývoje a inovací a členům dvou řešitelských týmu. Konaly se prezentace, v rámci kterých byli účastníci dotázáni, zda s představeným řešením souhlasí či nikoliv.

V oblasti metodik se řešení Metodická znalostní skupina, které vyplývá z problému Správa metodik, setkalo s největším úspěchem. Všichni dotázaní s řešením souhlasí. Již byly předkládány návrhy, kdo zastane jednotlivé role, které jsou v rámci metodické znalostní skupiny stanoveny.

Řešení problému s řešitelským týmem, kdy pracovník pracuje ve více řešitelských týmech, bylo odsouhlaseno členy řešitelských týmů - testerem, analytikem a programátorem. Management (generální ředitel společnosti, ředitel úseku Metodik a směrnic, ředitel úseku Vývoje a inovací, vedoucí programátorů) tento problém vnímá a navrženého řešení se snaží dosáhnout. Vzhledem k různé charakteristice projektů, jak vývojových, tak servisních, a také k různému zaměření a znalostem jednotlivých profesí není jednoduché sestavit řešitelské týmy tak, jak je definováno v návrhu řešení.

Vytvoření separátní metodiky pro vývojové projekty bylo odsouhlaseno

členy řešitelských týmů, vedoucím programátorů a ředitelem úseku Metodik a směrnic. Ředitel úseku Vývoje a inovací s navrženým řešením souhlasí pouze částečně. Servisní projekty obvykle mívají různou míru rozvoje. Otázkou je, kdy aplikovat metodiku určenou pro servisní projekty a kdy pro vývojové. Dle [5] je rozhodnutí závislé na konkrétních charakteristikách daného rozvoje.

Stanovení priorit událostí a činností u jednotlivých výstupů a činností se nesetkalo s neúspěchem.

Navržené řešení použít nástroj JIRA Software je dle klíčových uživatelů vhodné pouze u vývojových projektů. Vzhledem k přizpůsobení nástroje ISZA pro servisní projekty je velmi obtížné pokrýt všechny funkčnosti a přizpůsobit je tak, aby vyhovovaly nejen požadavkům zákazníků, ale i společnosti. Bylo by nutné vypořádat se se zavedenými reporty vedení, s kontrolními procesy, distribucemi a dalšími funkčnostmi. Na vývojových projektech je vyšší míra flexibility a nástroj lze uzpůsobit potřebám konkrétních projektů. V případě využití nástroje je potřeba zohlednit migraci dat a přizpůsobení nástroje JIRA Software. Nastavení nástroje, případné zaškolení zaměstnanců a další náležitosti související se zavedením nástroje by měly být v kompetenci znalostní metodické skupiny.

10 Závěr

V rámci diplomové práce byla podrobena analýze metodika (proces vývoje softwaru) vybrané softwarové společnosti a nástroj ISZA, jež slouží ke zpracování hlášení. Analýza probíhala ve spolupráci s ředitelem úseku Metodik a směrnic, dále s ředitelem úseku Vývoje a inovací a s dvěma řešitelskými týmy. Hlavní pozornost byla věnována metodice Tabulka činností a nástroji ISZA. Nad nabytými poznatky byla provedena kritická analýza, ze které vyplynuly problémy a jejich závažnost.

Z analýzy vyplynulo, že společnost má metodiku na dobré úrovni zejména pro servisní projekty. Byly identifikovány celkem čtyři problémy, mezi které patří například správa metodik či monolitický vývojový proces aplikovaný na servisní a vývojové projekty. Jako nejzávažnější problém je klasifikována správa metodik.

Mezi nejzávažnější problémy týkající se nástroje ISZA patří přeplánování projektů, alokace kapacit a nedostupné pracovní plochy. Nástroj způsobuje administrativní zátěž a do projektů je zanášen šum, který je zapříčiněn složitostmi nástroje.

Pro vypořádání se se zjištěné problémy byla navrhnuta řešení, která byla následně ověřena klíčovými uživateli, již se podíleli na analýze.

Na základě výsledků této práce by měla společnost založit metodickou znalostní skupinu, která bude spravovat metodiky a zároveň bude upravovat a vylepšovat procesy vývoje software. Dalším závěrem je implementovat nástroj JIRA Software na vývojové projekty. Nastavení nástroje a další náležitosti spojené se zavedením by měly být v kompetenci zmíněné metodické znalostní skupiny.

V návaznosti na tuto práci by bylo vhodné navrhnout metodiku pro vývojové projekty tak, aby odpovídala charakteristikám vývojových projektů a požadavkům vedení. Stejně tak by bylo žádoucí zaměřit se na zavedení nástroje JIRA Software na konkrétních projektech. Dále by měl být nástroj ISZA přizpůsoben servisním projektům tak, aby byla snížena administrativní zátěž.

Seznam pojmů a zkratek

- Betaverze - softwarový produkt, který je stále ve fázi vývoje.
- Branch - nová kopie projektu ve verzovacím systému.
- Commit - potvrzení editačních změn a jejich uložení do repozitáře systému sledování verzí.
- Hotline - zákaznická linka společnosti.
- Pull request - žádost o začlenění změn ve verzovacím systému.
- Tag - označení verze produktu ve verzovacím systému.

Literatura

[1] Ahmad, M. O. – Markkula, J. – Oivo, M. Kanban in software development: A systematic literature review. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, s. 9–16, Sep. 2013. doi: 10.1109/SEAA.2013.28.

[2] Atlassian. *Jira Software* [online]. 2019. Dostupné z: `https://cs.atlassian.com/software/jira`.

[3] Atlassian. *BigPicture - Project Management and PPM* [online]. 2019. Dostupné z: `https://marketplace.atlassian.com/apps/1212259/bigpicture-project-management-ppm`.

[4] Atlassian. *Trello* [online]. 2019. Dostupné z: `https://trello.com/`.

[5] Brown, W. J. *AntiPatterns in Project Management*. John Wiley and Sonsa, Inc., 2000.

[6] Carla O'Dell, C. H. *The New Edge in Knowledge: how knowledge management is changing the way we do business*. John Wiley and Sons, 2011. ISBN 978-0-470-91739-8.

[7] Cunningham, W. *Manifest Agilního vývoje software* [online]. 2001. Dostupné z: `http://agilemanifesto.org/iso/cs/manifesto.html`.

[8] Govardhan, D. A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues 1694-0814*. 09 2010, 7, s. 94–101.

[9] Kadlec, V. *Agilní programování*. Computer Press, a.s., 2004. ISBN 80-251-0342-0.

[10] Lang, J.-P. *Redmine* [online]. 2014. Dostupné z: `https://www.redmine.org/`.

[11] Larman, C. *Agile and iterative development*. Pearson Education, Inc., 2004. ISBN 0-13-111155-8.

[12] Ltd., E. S. *Easy Redmine* [online]. 2018. Dostupné z: `https://www.easyredmine.com/`.

[13] P. Naur, B. R. SOFTWARE ENGINEERING: Report of a conference sponsored by the NATO Science Committee. *Scientific Affairs Division, NATO*. Leden 1969, 231. Dostupné z: `http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF`.

[14] Per Kroll, P. K. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Pearson Education, Inc., 2006. ISBN 0321166094.

[15] Royce, W. W. MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS. *Proceedings, IEEE WESCON*. Srpen 1970, , 1-9. Dostupné z: `http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf`.

[16] S. P. Tayal, M. G. B. A. *Software Engineering and Testing*. Jones and Barlett Publishers, 2010. ISBN 978-1-934015-55-1.

[17] Sabharwal, S. *Software Engineering*. New Age International (P) Limited, 2008. Dostupné z: `https://books.google.cz/books?id=5K1EV57lYfwC`. ISBN 9788122423778.

[18] Schuurman, R. *Robbin Schuurman: Professional Scrum Trainer and Agile Expert* [online]. 2019. Dostupné z: `http://burozeven.nl/robbinschuurman/2017/11/20/10-tips-for-product-owners-on-the-scrum-framework/`.

[19] Sommerville, I. *Softwarové inženýrství*. Albatros Media, 2013. ISBN 978-80-251-3826-7.

[20] Voříšek, J. *Informační systémy a jejich řízení*. 278. Bankovní institut, 1997.

[21] Wiegers, K. E. *Požadavky na software*. Computer Press, 2008. ISBN 978-80-251-1877-1.

A Zápis z pracovního semináře 28. 11. 2018

Účastníci

- Lenka Rychtářová,
- ředitel úseku Metodik a směrnic,
- vedoucí projektů,
- vedoucí programátorů,
- analytik,
- tester.

Program

- Agilní hra "vánoční ozdoby",
- tradiční a agilní přístupy vývoje SW,
- ověření a identifikace problémů.

Zápis

Na úvod semináře byla připravena agilní hra "vánoční ozdoby". Úkolem bylo za stanovený čas vybarvit a vystřihnout vánoční ozdoby. Vedoucí projektu dodal odhad, kolik vánočních ozdob je tým za stanovený čas schopen vyrobit. Cílem bylo poukázat na to, že při vyšším počtu iterací se odhady více blíží realitě.

Byly shrnuty základní principy tradičních a agilních přístupů. Celá skupina se jednoznačně shodla, že na většině projektů se používá vodopádový model vývoje softwaru. U některých vývojových projektů se prosazuje iterativní přístup.

Celý tým primárně pracuje na servisních projektech. Bylo poukazováno na nepřiměřeně dlouhé období mezi fází návrhu softwaru a jeho testováním. Délka byla odhadovaná na dva až tři roky. Dlouhá prodleva poté zapříčiní, že se systém od doby návrhu mění a některé části neodpovídají fázi analýzy. Dále byla shledána nedostatečná komunikace se zákazníkem. Zpočátku je

vyvíjen tlak na rychlé dodání analýzy, ale schválení či připomínky jsou zákazníkem zaslány po měsících až letech. Zákazník nefiguruje v projektu po celou jeho dobu trvání. Členům týmu se nedostává adekvátní zpětná vazba. Tým taktéž nemá zpětnou vazbu od uživatelů, kteří v projektu nijak nefigurují. Tyto problémy jsou způsobeny specifickým prostředím zákazníka.

Na úloh se používá interní systém ISZA, ve kterém celý tým pracuje. Každému účastníkovi byla položena otázka, zda jsou v procesu nějaké problémy či nedostatky.

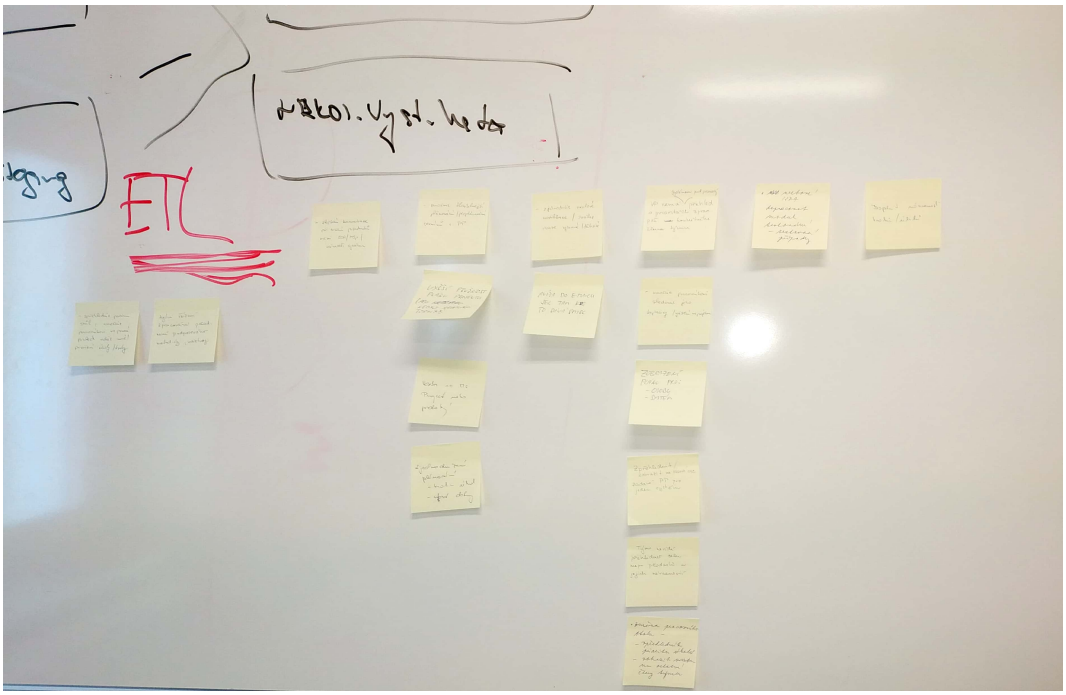
Tester dostává úlohu v momentě, kdy je vytvořen krok na testování. Do systému je možnost vložit komentáře a mezi týmem probíhá neustálá komunikace. Na pracovním stole je vedena evidence úkolů přiřazených pracovníkovi. Nově se pracovní stůl přepisuje do webového rozhraní. Tester webové rozhraní, které je uživatelsky přívětivější nevyužívá, protože není integrováno modul pro testování.

Analytik hodnotil nástroj ISZA jako administrativní zátěž. Nástroj je zastaralý a uživatelsky nepřívětivý. Po nástupu do zaměstnání bylo velmi obtížné se s nástrojem seznámit a naučit pracovat. Dále by byla uvítána možnost nastavení notifikací. V případě, že je založených pět požadavků, jednotlivé události v systému pošlou až šedesát emailových notifikací.

Vývojář poukazoval na časové plány vedené v ISZA, které neodpovídají realitě. Data, která jsou uvedena v systému u jednotlivých požadavků, obvykle neodpovídají skutečnosti.

Vedoucí projektu spravuje v nástroji přes sto požadavků. Bylo konstatováno, že nelze v objemu práce udržovat data aktuální. U požadavků jsou vytvářeny plány projektu, které mají zpravidla šest fází. Termíny fází lze posunou najednou, například o týden. Problém nastává, pokud se každá fáze posouvá o jiný časový úsek. Časový odhad na přeplánování plánu projektu je dvacet minut. Vedoucí projektu spravuje přes třicet systém, na kterých nepracuje stejný tým. V nástroji není umožněn přehled alokací jednotlivých členů týmu. Ostatní plány projektu nejsou na první pohled viditelné a navíc nebývají aktuální.

U některých vývojových projektů se využívá iterativní přístup. K evidenci úloh je používán nástroj Trello. V nástroji ISZA se vede pouze evidence o projektu z důvodů vykazování pracovního času. Na závěr setkání dostali účastníci za úkol sepsat hlavní problémy (viz obr. č. A.1).



Obrázek A.1: Problémy sepsané účastníky semináře

Problémy jsou s počtem výskytů vypsány v následující tabulce č. A.1. Z tabulky je zřejmé, že tým shledává jako největší potíž viditelnost požadavků napříč celým týmem. Nástroj vedoucímu projektu například neumožňuje přehled o požadavcích konkrétního člena týmu.

Název	**Počet výskytů**
Tým nevidí přehledně celou sadu požadavků a jejich návazností.	6
Přeplánování termínu v nástroji ISZA.	3
Možnost nastavení emailových notifikací dle preferencí.	2
Zpřehlednění pracovního stolu. Na první pohled nejsou viditelné požadavky s vysokou prioritou a nové požadavky.	1
Agilní řízení podpořit metodiky a nástroji.	1
Zlepšení komunikace při řešení požadavků se zákazníkem.	1
Chybějící modul testování ve webovém ISZA.	1
Doplnění návazností kroků / úkolů.	1

Tabulka A.1: Problémy sepsané účastníky semináře

B Zápis z pracovního semináře 17. 1. 2019

Účastníci

- Lenka Rychtářová,
- ředitel úseku Metodik a směrnic,
- Programátor,
- vedoucí projektu,
- tester,
- konzultant / analytik.

Program

- hra "zeptej se kolegy",
- tradiční a agilní přístupy vývoje SW,
- ověření a identifikace problémů.

Zápis

Na úvod semináře byla připravena hra "zeptej se kolegy". Účastníci schůzky byly rozděleni do dvojic. Jeden ze dvojice dostal na papíře vytištěný obrázek a zastával roli odpovídač, druhý dostal tužku a papír a zastával roli vyptávač. Úkolem vyptávače bylo za stanovený čas na základě položených otázek nakreslit vytištěný obrázek. Cílem hry je uvědomit si, jak důležitá je formulace otázky i odpovědi. Většina odpovídačů odpovídala na otázky formou ano / ne, i přesto že na začátku nebyla definovaná pravidla otázek ani odpovědí.

Po seznámení byla připravena krátká prezentace shrnující základní principy tradičních a agilních přístupů. Skupina došla k závěru, že je převážně aplikován vodopádový model. Celá skupina je plně obeznámena s metodikou firmy. Portály s metodikami shledávají přehledné.

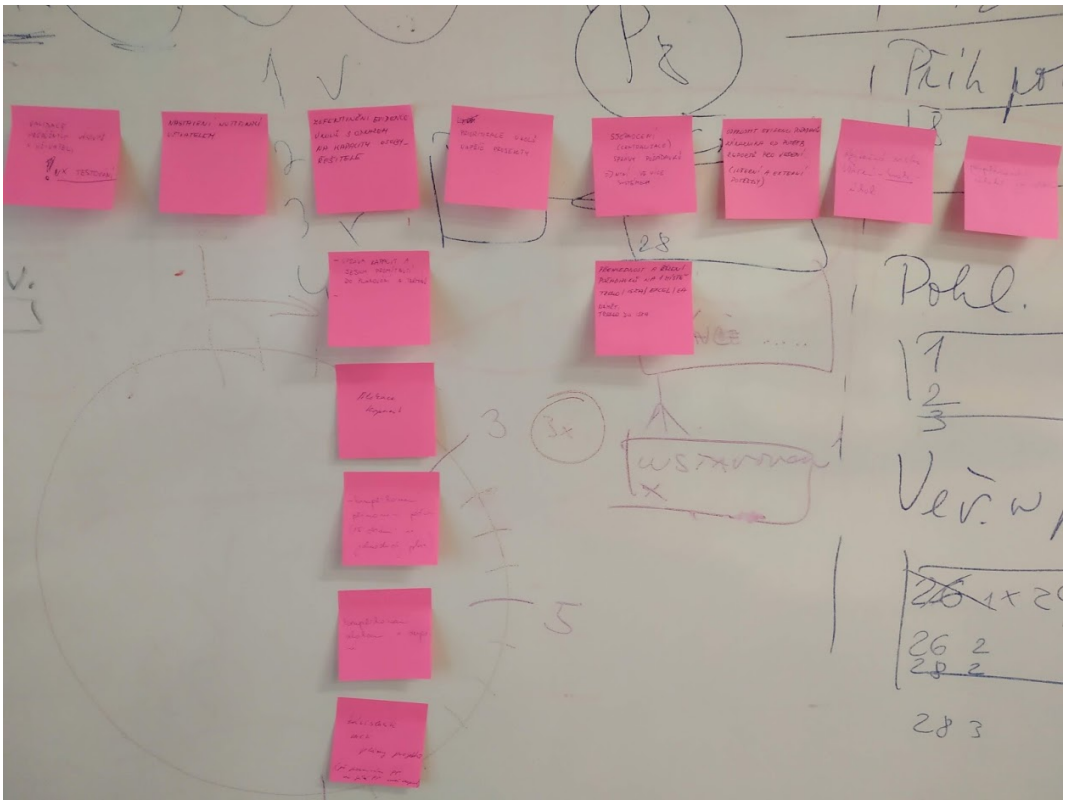
Každý ze zúčastněných měl možnost vyjádřit se k otázce, jak pracuje s hlášeními a zda jsou nějaká slabá místa procesů správy hlášení. Skupina

došla k závěru, že nástroj ISZA je uživatelsky nepřívětivý a zastaralý. Formuláře, které jsou v nástroji používané, jsou nepřehledné a neintuitivní. V případě nástupu nového pracovníka je nutné absolvovat náročné školení. I po absolvování školení je obtížná orientace v systému.

Účastníci by uvítali nastavení notifikací dle jejich preferencí. Vedoucímu projektu současné nastavení notifikací vyhovuje, oproti tomu programátor emailové notifikace ignoruje.

Dále byly diskutovány alokace pracovníků. Nástroj neumožňuje efektivní plánování alokací. Ideální případ by byl, kdyby se dal zobrazit kalendář pracovníka s naplánovanou prací.

Na závěr workshopu měli účastníci za úkol napsat hlavní problémy spojené se správou požadavků (viz obr. č. B.1).



Obrázek B.1: Problémy sepsané účastníky semináře

Jednotlivé problémy jsou vypsány v následující tabulce č. B.1. Z tabulky je evidentní, že skupina shledává jako klíčový problém alokaci kapacit jednotlivých pracovníků.

Název	**Poč. výskytů**
Zlepšení alokace kapacit	6
Správa požadavků v jednom systému (ISZA x Trello)	2
Nastavení notifikací uživatelem	1
Prioritizace úkolů napříč projekty	1
Odstranění kontrolních mechanismů pro vedení	1
Validace prováděných výstupů s uživatelem (UX testování)	1

Tabulka B.1: Problémy sepsané účastníky semináře

C Akční plán založení MZS

Akční plán – založení MZS

Projekt: Založení metodické znalostní skupiny **Datum:** 22. 3. 2019

Cíle:

Cílem je vytvoření metodické znalostní skupiny složené z rolí MZS vůdce, MZS specialista, MZS komunikační direktor a MZS analytik. Účelem skupiny je:

- aktivně spravovat metodiky firmy,
- šířit znalosti o metodikách, nástrojích,
- monitorovat používání metodik,
- šířit možnost změny,
- analyzovat nové přístupy, nástroje,
- zlepšovat stávající procesy.

Měřítko úspěchu:

Vytvoření metodické skupiny, přiřazení jednotlivých rolí.

Dopad změn:

-

Účastníci:

Lenka Rychtářová, ředitel úseku Metodik a směrnic

Sledování hlášení pokroku:

-

Závislosti, rizika, omezení:

Rizikem je vyčlenění osob a získání jejich kapacit. Dále obhájení akčního plánu u managementu firmy.

Odhadované datum dokončení:

5. 4. 2019

Práce:

Práce	Vedoucí	Termín	Úkol	Popis	Výstupy	Potřebné zdroje
1	ředitel úseku Metodik a směrnic	29.3.2019	Založení MZS	Nalezení klíčových pracovníků, kteří by obsadili jednotlivé role.	Sestavený tým.	Maximálně 4 zaměstnanci
2	ředitel úseku Metodik a směrnic	5.4.2019	Komunikace plánu k managementu	Obeznámení vedení o plánované činnosti.	Schválení procesu managementem.	-
3	Lenka Rychtářová	5.4.2019	Představení rolí a činností znalostní skupiny	Uspořádání workshopu, kde budou představeny role skupiny a jednotlivé činnosti. Představení návrhu na zlepšování procesů.	Seznámení o projektu.	Maximálně 4 zaměstnanci, 1 hodina workshopu

Obrázek C.1: Akční plán - založení metodické znalostní skupiny