VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

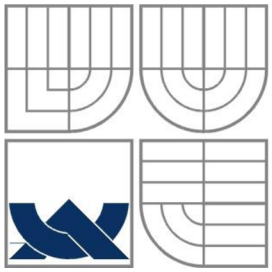
ZÁSUVNÝ MODUL SYSTÉMU JIRA PRO AUTOMATICKÉ PŘIŘAZOVÁNÍ TIKETŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

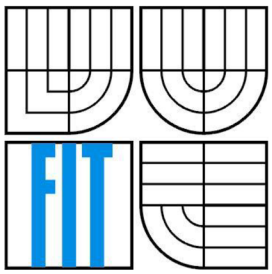
AUTOR PRÁCE
AUTHOR

Bc. PAVLA KŮRKOVÁ

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÁSUVNÝ MODUL SYSTÉMU JIRA PRO AUTOMATICKÉ PŘIŘAZOVÁNÍ TIKETŮ

JIRA PLUG-IN FOR AUTOMATIC TICKET ASSIGNING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVLA KŮRKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. JITKA KRESLÍKOVÁ, CSc.

BRNO 2013

Abstrakt

Tato práce se věnuje tvorbě zásuvného modulu pro systém JIRA společnosti Atlassian. Modul byl vytvořen na základě požadavků reálně společnosti a dodáván jako projekt, odpovídající firemním procesům.
Hlavními cíli bylo v souladu s pravidly společnosti rozšířit funkcionalitu používaného systému a souběžně navrhnout úpravy procesu dodání projektu pro projekty podobného typu.

Klíčová slova

Projektové řízení, IPMA, PMBOK, PRINCE2, malý projekt, JIRA, Atlassian, zásuvný modul, Velocity, Active Objects, i18n.

Abstract

This work is aimed on creating plug-in module for the system Atlassian JIRA. The module was developed based on the requirements of real company and provided as a project corresponding with business processes.
The main objective was to extend the functionality of used system and to design project delivery process modifications for projects of similar type.

Keywords

Project management, IPMA, PMBOK, PRINCE2, small project, JIRA, Atlassian, Plug-in, Velocity, Active Objects, i18n.

Citace

Kůrková Pavla: Zásuvný modul systému JIRA pro automatické přiřazování tiketů. Brno, 2013, diplomová práce, FIT VUT v Brně.

Zásuvný modul systému JIRA pro automatické přiřazování tiketů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením paní docentky Jitky Kreslíkové.
Další informace mi poskytli odborní konzultanti společnosti, pro niž byla práce vytvořena.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Pavla Kůrková
22. 5. 2013

Poděkování

Mé poděkování patří především vedoucí této práce, paní docentce Kreslíkové, která mi poskytla cenné odborné rady, podklady k práci a vycházela mi maximálně vstříc při tvorbě práce.
Dále bych chtěla poděkovat konzultantům a zaměstnancům společnosti, ve které práce vznikala, za odborné rady, zpřístupnění interních informací a systémů a také za umožnění tvorby práce přímo ve společnosti a poskytnutí všech potřebných zdrojů.

Obsah

1 Úvod

Tato práce se věnuje implementaci zásuvného modulu systému JIRA pro reálnou společnost. Tvorba modulu je zasazena do kontextu projektového řízení, protože musí být dodána ve formě projektu odpovídajícího projektovým procesům společnosti.

Následující kapitola obsahuje metodiky a znalostní oblasti projektového managementu, včetně popisu procesu řízení projektů v reálné společnosti. Svou část zde má teorie vycházející ze standardů řízení i poznatky založené na praktických zkušenostech z dřívějších projektů společnosti.

Další kapitola je věnována samotnému systému JIRA, zahrnuje stručný popis systému jako celku a také několik podkapitol, které se detailněji věnují částem stěžejním pro tuto práci – tiketům, událostem a zásuvným modulům.

Čtvrtá kapitola navazuje na teoretický základ z druhé kapitoly, aplikuje získané poznatky na aktuální stav řízení ve společnosti. Výsledkem je návrh nového postupu pro řízení malých interních projektů ve společnosti.

Předposlední kapitola se věnuje samotné realizaci projektu, tj. aplikaci navržených úprav v reálném prostředí. Jsou zde zahrnuty výstupy z řízení projektu, popisy technických částí i nové poznatky získané během realizace. Nedílnou součástí je i vyhodnocení projektu.

Závěrečná kapitola přehledně shrnuje obsah a průběh práce jako celku. Dosažené výsledky jsou stručně posouzeny a doplněny o možnosti dalšího použití a rozvoje.

Kapitola věnovaná projektovému řízení vychází z popisu řízení, uvedeném v semestrálním projektu. V této práci je téma dále rozšířeno o standardy řízení a specifika malých projektů. Popis systému JIRA ve třetí kapitole prakticky odpovídá obsahu semestrálního projektu, doplněna byla pouze poslední podkapitola a několik drobných detailů v ostatních podkapitolách.

Původní kapitola Návrh realizace projektu byla pro potřeby diplomové práce rozdělena na dvě kapitoly – Návrh realizace malých projektů a Projekt AutoAssign. Tato kapitola také doznala největších změn a rozšíření, popisuje totiž praktickou realizaci, která nebyla součástí teoretických podkladů v semestrálním projektu.

2 Projektové řízení

Důležitou součástí každého projektu je jeho řízení, proto můžeme nalézt řadu postupů managementu projektů, od neformálních rad zkušenějších až po standardizované a uznávané komplexní metodiky. Tato kapitola uvádí základní pojmy oblasti, popisuje vybrané obecné standardy projektového řízení a také jejich konkrétní aplikaci přímo ve společnosti. Poslední část se pak věnuje specifikům řízení malých projektů.

2.1 Základní pojmy

V literatuře se můžeme setkat s řadou definic odborných pojmů managementu projektů, základní rysy však bývají velmi podobné. Pro účely této práce vyjdeme z následujících definic pojmů:

Projekt je časově, nákladově a zdrojově omezený proces realizovaný za účelem vytvoření definovaných výstupů [1]. Projekt má tedy tyto parametry [2]:

- začátek a konec (ohraničený časový harmonogram),
- jednoznačně určené cíle a výstupy,
- omezené zdroje a
- je v daném místě a čase jedinečným nástrojem inovace.

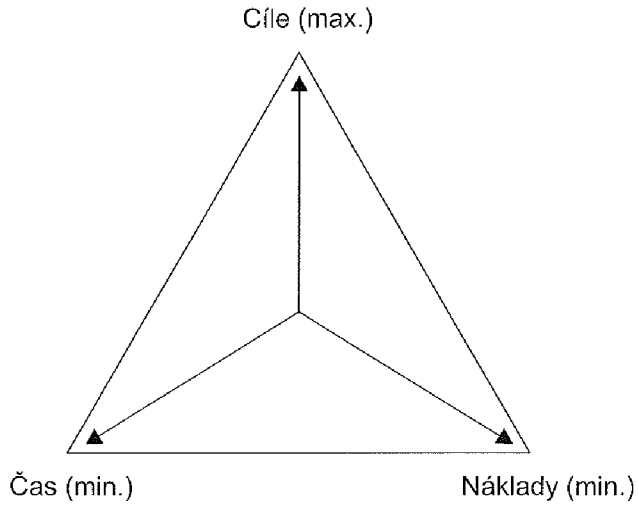
Cílem projektu je vytvořit odsouhlasené výsledky v požadovaném časovém rámci, rozpočtu a akceptovatelných parametrů rizika [1]. Lze jej chápat jako stav, do kterého se chceme dostat. Dobře stanovený cíl splňuje SMART pravidla – je specifický, měřitelný, akceptovaný, realistický a termínovaný [3].

Projektový trojimperativ popisuje cíl projektu jako vztah tří provázaných základních veličin – výsledku (též kvality, cílů), času a zdrojů (též nákladů) [2], [3]. Často je zobrazován jako trojúhelník, viz Obrázek 1.

Projektové řízení (též management projektů nebo projektový management) je aplikace znalostí, dovedností, nástrojů a technik na projektové aktivity tak, aby bylo dosaženo cílů projektu [4]. Jako účel řízení můžeme chápat též redukci rizika neúspěšného zakončení projektu [5].

Úspěšnost řízení projektu dle zdroje [1] chápeme jako kladné ocenění výsledků projektu různými zainteresovanými stranami. Často je tento pojem zaměňován s **úspěšností projektu,** tu však chápeme jako splnění kritérií úspěchu. Obecně se jedná o dva odlišné pojmy, neúspěšný projekt mohl být úspěšně řízen a naopak [3].

Zainteresovanou stranou může být jednotlivec, skupina či organizace, která má vliv, je ovlivněna nebo se ovlivněna cítí rozhodnutím, aktivitou nebo výstupem projektu [6].



Obrázek 1 Trojimperativ [převzato z [3]]

2.2 Standardy projektového řízení

Jak bylo řečeno v úvodu, v současné době existuje řada přístupů k projektovému řízení, někdy mají formu doporučení na základě zkušeností („best practices“), jindy se jedná přímo o standardy (např. PMBOK®Guide, IPMA Competence Baseline, PRINCE2, ISO 10006). Ve všech případech je hlavním cílem projektového řízení návrh a realizace úspěšných projektů, tedy úspěšného dosažení cílů projektu v daném čase, nákladech a kvalitě [7]. Vybraným standardům se v této podkapitole budeme věnovat více.

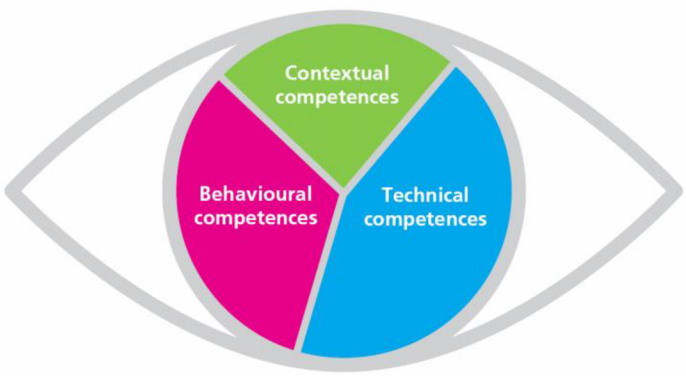
2.2.1 IPMA Competence Baseline

IPMA Competence Baseline (ICB) je především standardem profesionálního chování projektových manažerů a členů týmu. Dnes je k dispozici již třetí vydání této normy [1]. Standard byl vytvořen asociací International Project Management Association (IPMA), u nás je podporován Společností pro projektové řízení o. s. (SPŘ) a zájemci mohou pro tento standard získat také oficiální certifikaci (více informací viz webové stánky SPŘ [8]).

Na rozdíl od jiných standardů se ICB se primárně věnuje tomu, co by měl projektový manažer znát a ovládat – jeho kompetencím[1] – nikoliv konkrétním postupům řízení [9]. Kompetence jsou rozděleny do tří velkých skupin – technické, behaviorální, a kontextové kompetence, viz Obrázek 2. Každá z těchto skupin potom obsahuje několik dílčích kompetencí,

[1] Kompetencí se rozumí prokázaná schopnost použít znalosti a/nebo dovednosti a ve vhodných případech i prokázat patřičné osobní vlastnosti [1].

které jsou blíže popsány ve standardu. Popis zahrnuje např. možné procesní kroky elementu, dotčená témata, hlavní vztahy s dalšími elementy, apod. Následováním doporučení a aplikací postupů uvedených v jednotlivých kapitolách by potom mělo být dosaženo úspěšného řízení projektů.



Obrázek 2 Oko kompetencí [převzato z [1]]

Kromě standardu samotného je u nás dostupný také Výkladový slovník pojmů (k dispozici také online [10] a kniha Projektové řízení podle IPMA od autorů J. Doležala, P. Máchala a B. Lacka [3]. Tato přidává k jednotlivým částem standardu detailnější vysvětlení, konkrétní příklady a ukázky, je proto vhodnou studijní literaturou řízení podle IPMA.

Standard ICB se nevěnuje pouze projektům, ale zahrnuje i jejich širší okolí. Tím jsou koncepty *program* a *portfolio.* Programem chápeme skupinu věcně souvisejících a společně řízených projektů. Portfolio potom zahrnuje projekty a programy, které nemusí být propojeny, jsou však seskupeny za účelem řízení, kontroly a koordinace [1]. Lze zde tedy sledovat jakousi hierarchii a vzájemné vztahy uvedených tří konceptů.

Nedílnou součástí standardu jsou nástroje a techniky používané v projektovém řízení. ICB samotné není vázáno na použití konkrétního nástroje či techniky, existuje však řada ověřených a doporučených postupů. Přehled základních používaných technik je uveden v příloze č. 1 normy [1], pro ilustraci uveďme například známé techniky Logické rámcové matice, SMART klasifikace cílů projektu a SWOT analýzu pro technické kompetence; principy asertivity, využití zpětné vazby nebo vedení týmů pro behaviorální kompetence.

Následující podkapitoly souhrnně popisují jednotlivé kategorie elementů, všechny uvedené informace vychází přímo ze standardu ICB [1].

2.2.1.1 Technické kompetence

První a největší skupinu tvoří technické kompetence, jejichž obsahem jsou profesionální záležitosti projektového řízení. Jedná se o zásadní elementy kompetencí projektového řízení a spadají sem ty oblasti projektového řízení, které se označují též jako tzv. „pevné elementy“.

ICB uvádí celkem dvacet technických kompetencí: Úspěšnost řízení projektu, Zainteresované strany, Požadavky a cíle projektu, Rizika a příležitosti, Kvalita, Organizace projektu, Týmová práce, Řešení problémů, Struktury v projektu, Rozsah a dodávané výstupy projektu, Čas a fáze projektu, Zdroje, Náklady a financování, Obstarávání a smluvní vztahy, Změny; Kontrola, řízení a podávání zpráv; Informace a dokumentace, Komunikace, Zahájení a Ukončení.

Porozumění a schopnost aplikace technických kompetencí tvoří základ profesní způsobilosti projektového manažera. Elementy se využívají nejen při iniciování, zahájení či ukončení projektu, ale též pro řízení samotné realizace. Pořadí elementů uvedené není nijak závazné, aplikace elementů i jejich důležitost je závislá na konkrétním projektu.

2.2.1.2 Behaviorální kompetence

Druhá skupina kompetencí zahrnuje elementy, které se týkají personálního projektového řízení. Zahrnuje práci s osobními vztahy mezi jednotlivci i skupinami řízenými v rámci projektů, programů a portfolií. Z pohledu projektového manažera sem tedy náleží jeho osobní přístup a dovednosti. Důležitost behaviorálních kompetencí je silně závislá na aktuální situaci v projektu.

Standard ICB zahrnuje celkem patnáct behaviorálních elementů relevantních pro projektové řízení a kontext. Jedná se o tyto kompetence: Vůdcovství, Zainteresovanost a motivace, Sebekontrola, Asertivita, Uvolnění, Otevřenost, Kreativita, Orientace na výsledky, Výkonnost, Diskuze, Vyjednávání, Konflikty a krize, Spolehlivost, Porozumění hodnotám a Etika.

2.2.1.3 Kontextové kompetence

Do poslední skupiny jsou zařazeny elementy kompetencí kontextových, tedy těch, které souvisí s interakcemi mezi projektovým týmem, kontextem projektu a trvalé organizace. U projektového manažera se jedná např. o kompetence využívané při řízení organizací s liniovým řízením nebo jeho schopnost fungovat v organizaci zaměřené na projekt. Elementy kontextových kompetencí tedy popisují koncepce projektu, programu a/nebo portfolia a propojení těchto koncepcí s organizací nebo organizacemi, které se projektu účastní.

Prvních pět elementů – Orientace na projekt, Orientace na program, Orientace na portfolio; Realizace projektu, programu a portfolia a Trvalá organizace – se věnuje podpoře managementu projektu, programu a portfolia v organizaci. Zbývajících šest je potom zaměřeno na podpůrné funkce v trvalé organizaci a minimální znalosti o těchto funkcích potřebné pro projektové týmy: Byznys; Systémy, produkty, technologie; Personální management; Zdraví, bezpečnost, ochrana života a životního prostředí; Finance a Právo.

2.2.2 PMBOK

Project Management Body of Knowledge (PMBOK) je americkým standardem projektového řízení vydávaným institutem Project Management Institute (PMI). U nás jej využívají především firmy vlastněné americkými společnostmi [3]. Také pro tento standard lze získat certifikaci, více viz webové stránky České komory PMI [11].

Slovy samotných autorů je PMBOK „standardem pro řízení většiny projektů po většinu času napříč různými druhy průmyslu“ a „popisuje procesy, nástroje a techniky projektového řízení používané pro řízení projektu vstříc úspěšným výstupům“ (volně přeloženo z [4]). Aktuálně je k dispozici páté vydání normy, které vyšlo v roce 2013 [6], z něj také primárně vychází informace v této kapitole.

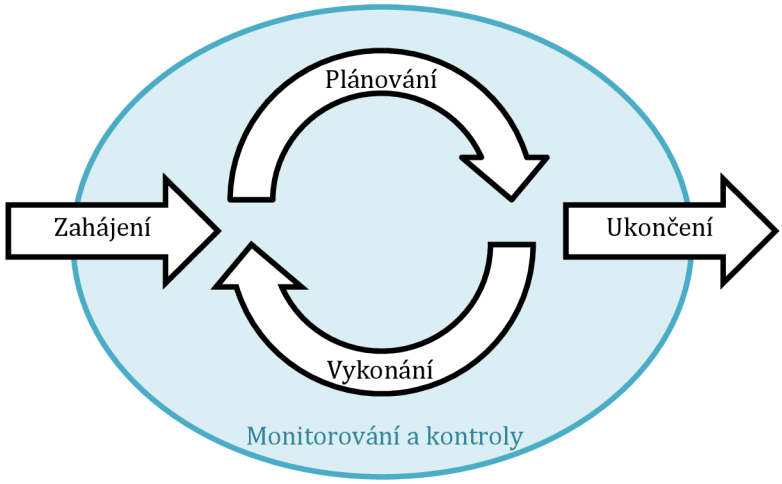
Analogicky k ICB také PMBOK zahrnuje do projektového řízení tři úrovně – projekt, program a portfolio [6]. Projekt tedy vnímá a řídí včetně jeho kontextu. Na rozdíl od ICB však PMBOK obsahuje konkrétní postupy pro řízení, ne jen popis potřebných znalostí [9].

Management projektů z pohledu PMBOK typicky zahrnuje [6]:

- identifikaci požadavků,
- cílení na potřeby zainteresovaných stran,
- efektivní komunikaci se zainteresovanými stranami,
- řízení zainteresovaných stran s ohledem na požadavky a výstupy projektu,
- balancování projektových omezení.

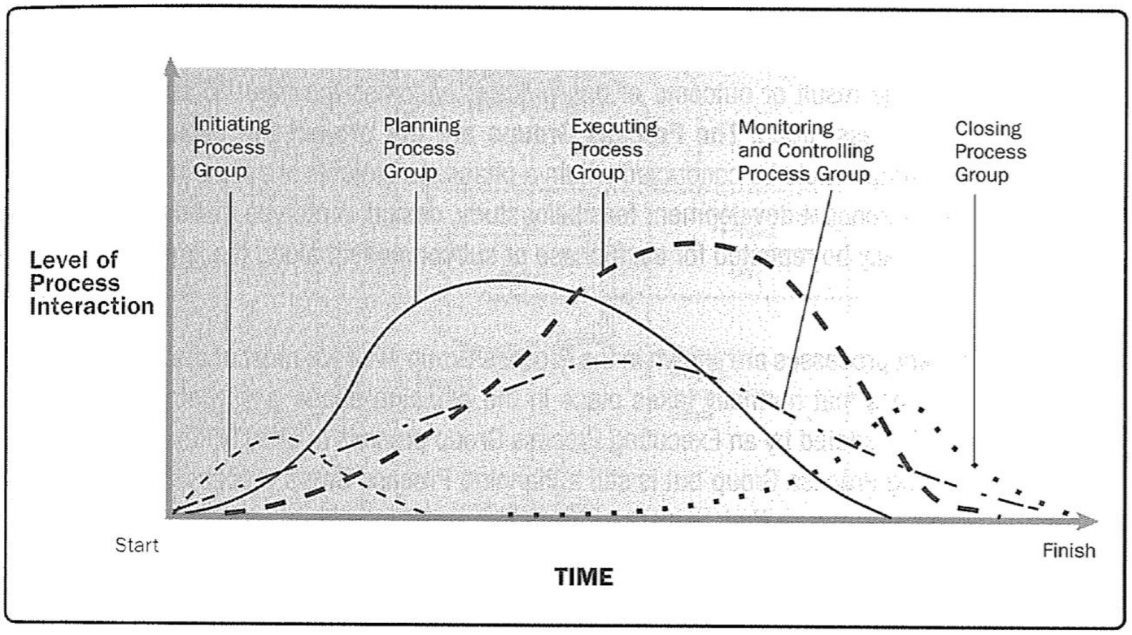
Zmiňovaná omezení potom zahrnují rozsah, kvalitu, čas, rozpočet, zdroje a rizika. Úroveň zaměření se na určité omezení závisí na charakteristikách a stavu konkrétního projektu.

PMBOK je procesně orientovaným standardem, celé řízení projektů popisuje pomocí čtyřiceti sedmi procesů seskupených do pěti logických celků: Zahájení, Plánování, Vykonání (Realizace), Monitorování a kontroly a Ukončení (Uzavření) [6].



Obrázek 3 Skupiny procesů podle PMBOK [převzato z [9]]

Úroveň aktivity procesů z jednotlivých skupin se mění v závislosti na pokroku v projektu, vzájemně se však prolínají a nelze je striktně oddělit podle fáze projektu. Jejich vzájemné vztahy ilustruje Obrázek 3, analogické vazby pak lze pozorovat i v rámci jednotlivých iterací v projektu (každá iterace může být chápana jako malý projekt). Aktivita procesů z jednotlivých celků vzhledem k fázi projektu zachycuje Obrázek 4. Detailnější popisy jednotlivých celků obsahují podkapitoly následující níže.



Obrázek 4 Aktivity skupin procesů dle fáze projektu [převzato z [6]]

Kromě rozdělení do logických fází jsou procesy seskupeny také do znalostních oblastí, kterých je v nové verzi celkem deset [6]. PMBOK obsahuje ke každé z nich detailní popis a konkrétní kroky její realizace.

Znalostní oblasti lze rozdělit do tří skupin – základní funkce, podpůrné funkce a řízení integrace (viz Obrázek 5). Toto rozdělení bylo vytvořeno pro starší verzi standardu [12], původní oblast komunikace byla proto v souladu s pátou verzí standardu a pro účely této práce nahrazena dvěma oblastmi – komunikací a zainteresovanými stranami. Umístění do podpůrných funkcí zůstalo zachováno.

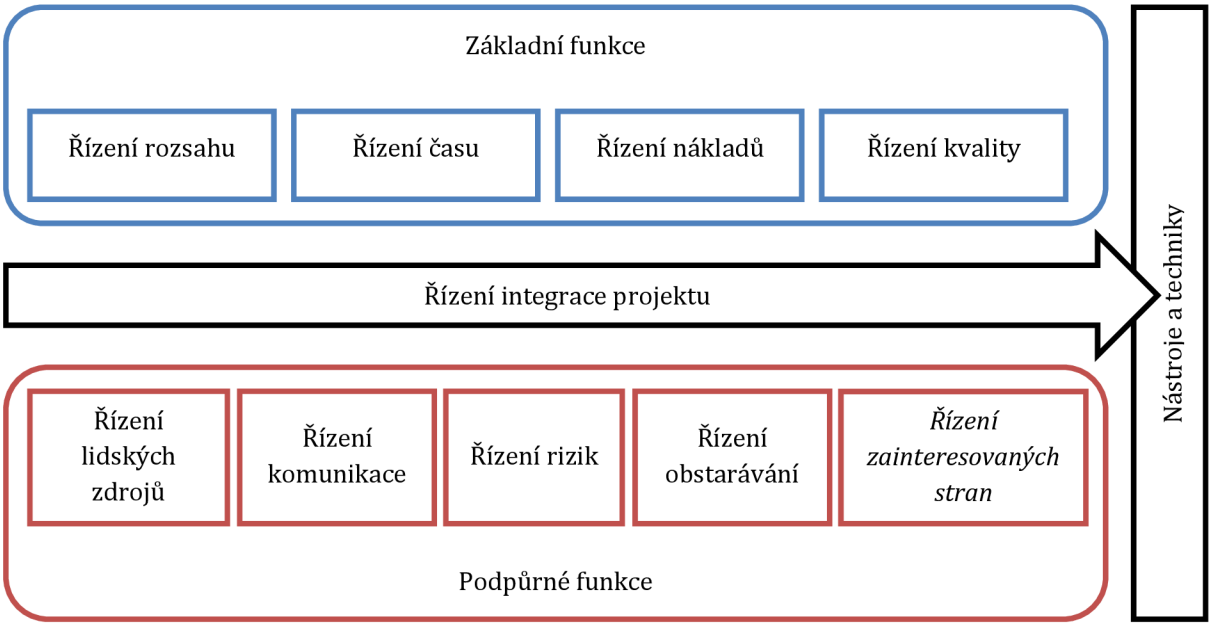
Základní funkce vedou k naplnění cílů samotného projektu [9] a zahrnují:

- řízení rozsahu (RRo) – definování a řízení prací nezbytných k úspěšnému dokončení projektu,
- řízení času (RC) – odhady doby prací, návrh časového plánu a zajištění včasného dokončení,
- řízení nákladů (RN) – příprava a řízení rozpočtu a
- řízení kvality (RKv) – zajištění, že projekt naplní stanovené a předpokládané potřeby.

Podpůrné funkce definují procesy, jejichž prostřednictvím dosahujeme cílů [9] a spadají sem znalostní oblasti:

- řízení lidských zdrojů (RLZ) – efektivní nasazení na projektu,
- řízení komunikace (RKo) – generování, shromažďování, distribuce a ukládání informací,
- řízení rizik (RRi) – identifikace, analýza a reakce na rizika,
- řízení obstarávání (RO) – zajištění potřebného zboží a služeb třetích stran,
- řízení zainteresovaných stran[2] (RZS) – řízení vlivu, komunikace a zájmů stran dotčených projektem a ovlivňujícím projekt.

Poslední částí je jediná oblast a to **řízení integrace** (RI) projektu, která je průřezovou oblastí a koordinuje ostatní oblasti během celého životního cyklu. Jejím účelem je zajištění, že vše potřebné proběhne ve správný čas [9].



Obrázek 5 Oblasti znalostí PMBOK [převzato z [12], upraveno pro soulad s 5. verzí standardu]

2.2.2.1 Procesy zahájení

Procesy inicializace zajišťují rozběhnutí nového projektu či fáze – definování rozsahu, zdrojů, identifikaci zainteresovaných stran, rozdělení do fází, schválení apod. Řadíme sem dva procesy: Vytvoření zadávací listiny projektu (RI)[3] a Identifikace zainteresovaných stran (RZS) [6]. Zahajovací procesy probíhají na začátku každé projektové fáze, ne jen na začátku celého projektu [12].

---

[2] Přidáno autorkou této práce pro zachování souladu s aktuální verzí standardu.
[3] Zkratka uvedená v závorce za názvem procesu označuje, do které znalostní oblasti proces spadá. Označení je v souladu s označením uvedeným v přehledu oblastí výše.

2.2.2.2 Procesy plánování

Nejvíce procesů spadá právě do skupiny plánování. Jejich cílem je určení celkového rozsahu a výstupů, včetně stanovení potřebných aktivit [6]. Plánování navazuje na výstupy inicializace, především na identifikaci zainteresovaných stran. Jeho vlastní výstupy pak musí pokrýt všechny znalostní oblasti projektu [12].

Celkem plánování zahrnuje dvacet čtyři dílčích procesů: Vytvoření plánu řízení projektu (RI), Plánování řízení rozsahu (RRo), Sběr požadavků (RRo), Určení rozsahu (RRo), Vytvoření WBS (RRo), Plánování řízení času (RC), Definování aktivit (RC), Seřazení aktivit (RC), Odhad zdrojů pro aktivity (RC), Odhad trvání aktivit (RC), Vytvoření harmonogramu (RC), Plánování řízení nákladů (RN), Odhad nákladů (RN), Určení rozpočtu (RN), Plánování řízení kvality (RKv), Plánování řízení lidských zdrojů (RLZ), Plánování řízení komunikace (RKo), Plánování řízení rizik (RRi), Identifikace rizik (RRi), Provedení kvalitativní analýzy rizik (RRi), Provedení kvantitativní analýzy rizik (RRi), Plánování reakcí na rizika (RRi), Plánování řízení obstarávání (RO) a Plánování řízení zainteresovaných stran (RZS).

2.2.2.3 Procesy vykonání

Procesy realizace zahrnují samotnou práci na projektu a skrze ně jsou naplňovány cíle projektu. Patří sem koordinace zdrojů, zainteresovaných stran, integrování a aktivity stanovené projektovým plánem. V průběhu realizace může docházet k aktualizacím nebo změnám, toto je pro případ potřeby ošetřeno procesy změn, rizik a příslušných reakcí [6].

Patří sem těchto osm procesů: Směřování a správa realizace projektu (RI), Zajištění kvality (RKv), Získání projektového týmu (RLZ), Rozvoj projektového týmu (RLZ), Řízení projektového týmu (RLZ), Řízení komunikace (RKo), Řízení zapojení zainteresovaných stran (RZS) a Vedení obstarávání (RO).

2.2.2.4 Procesy monitorování a kontrol

Účelem této skupiny procesů je sledování, revidování a regulace průběhu a výkonu projektu, případně identifikace a inicializace potřebných změn. Průběžné monitorování umožňuje efektivní zachycení odchylek od plánu a jejich ošetření. Kromě toho zahrnuje tato skupina řízení změn, korelaci aktivit s plánem a kontrolu projektu jako celku. U více fázových projektu zajišťuje také koordinaci fází.

Celkem do této skupiny spadá jedenáct procesů: Monitorování a kontrola práce na projektu (RI), Kontrolování integrovaných změn (RI), Validace rozsahu (RRo), Kontrola rozsahu (RRo), Kontrola časového plánu (RC), Kontrola nákladů (RN), Kontrola kvality (RKv), Kontrola komunikace (RKo), Kontrola rizik (RRi), Kontrola obstarávání (RO) a Kontrola zapojení zainteresovaných stran (RZS).

2.2.2.5 Procesy ukončení

Poslední skupina projektů má za cíl finalizaci aktivit, potvrzení korektního ukončení projektu či fáze a formální označení projektu/fáze jako kompletní [6]. Součástí uzavírání může být mj. akceptace, revize projektu apod. Typicky sem spadají také administrativní činnosti jako archivace dat, ukončení smluvních závazků atd. [12]. Procesy náležící k této skupině jsou dva – Uzavření projektu/fáze (RI) a Uzavření obstarávání (RO).

2.2.3 PRINCE2

Metodika řízení projektů PRINCE2 (PRojects IN Controlled Environments) pochází z Velké Británie a stala se oblíbenou i v řadě dalších evropských zemí. U nás bývá využívána dceřinými společnostmi britských firem [3]. Obecně je považována za robustní a promyšlený způsob řízení projektů všech velikostí. Původně byla navržena pro projekty v IT, aby zajistila jejich kvalitu, dnes se však využívá v celé řadě oborů [3]. Garantem a vlastníkem metodiky je britská vládní organizace Office of Government Commerce (OGC), certifikát k metodice lze získat u celé řady akreditovaných společností (pro více informací viz webové stránky APMG International [13]).

PRINCE2 je důsledně procesně orientovaná [3] a poskytuje komplexní aparát navazujících komponent, kontrolních mechanismů i konkrétních projektových dokumentů [5]. Jejím základem jsou zavedené a ověřené praktiky projektového managementu v IT (best practices) [13].

Základním schématem projektového řízení je cyklus Plan – Delegate – Monitor – Control (viz Obrázek 6). Smyslem řízení je potom kontrola všech šesti aspektů úspěšnosti projektu – nákladů, času, kvality, rozsahu, rizik a přínosů [14]. Řízení je realizováno skrze osm hlavních procesů a řady pod-procesů. Ty jsou sdruženy šesti komponent, tedy jakýchsi oblastí pokrývajících jednotlivé části projektového řízení. Další součástí PRINCE2 je popis technik využívaných během těchto procesů [15].

Obrázek 6 Projektový management podle PRINCE2 [převzato z [14]]

2.2.3.1 Komponenty

Jak bylo zmíněno výše, PRINCE2 je rozdělen do komponent, které účelně sdružují procesy dotýkající se určité oblasti projektového řízení. Jedná se o tyto komponenty: Organizační řízení, Plánování, Nástroje kontroly, Fáze, Řízení rizik, Kvalita v projektovém prostředí, Řízení konfigurace a Řízení změn.

2.2.3.2 Techniky

Manuál k PRINCE2 dělí techniky do tří základních skupin [15]:

- Produktové plánování – plánování podle druhu produktu,
- Kontrola kvality a
- Řízení změn – včetně jejich identifikace.

2.2.3.3 Procesy

PRINCE2 zahrnuje osm procesů – Začátek projektu, Inicializace projektu, Plánování, Řízení projektu, Kontrola, Realizace, Vymezení projektu a Uzavření projektu. Každý z nich obsahuje ještě několik dalších pod-procesů, celkově se tak dostaneme na 45 procesů k realizaci. Popis každého z procesů obsahuje jeho princip, kontext, samotnou definici, škálovatelnost, odpovědnost, kritéria, další potřebné informace a také tipy a triky z praxe [15].

Na základě výše uvedených popisů metodik PMBOK a PRINCE2 si lze povšimnout určité podobnosti mezi oběma standardy. V obou případech tvoří jádro řízení procesy a ty jsou dále seskupovány do určitých logických celků, vzájemně propojovány vazbami a detailně popsány tak, aby bylo možné je přímo provádět. PRINCE2 je vytvořen přímo pro IT prostředí, zatímco PMBOK je orientován více obecně, ovšem oba standardy si v zásadě neodporují a v odborné literatuře lze nalézt i jejich srovnání a vzájemné převody [15].

Prvkem společným pro všechny tři uvedené standardy je uvedení technik využitelných při řízení, např. produktových rozpadů (WBS), SWOT analýz, Ganttova diagramu, atd.

2.3 Řízení projektů ve společnosti

Tato kapitola se blíže věnuje danému postupu řízení projektů, konkrétně postupu aplikovanému v rámci reálné softwarové společnosti, pro kterou je implementován projekt tvořený v rámci této práce.

V rámci společnosti se na vývoj software na zakázku nahlíží jako na službu a její dodání zajišťuje proces nazvaný Proces dodání projektu (Project Delivery Process, PDP). Tento je navázán na systém řízení certifikovaný podle norem ISO 9001, ISO 20000-1a ISO 27001 a jeho

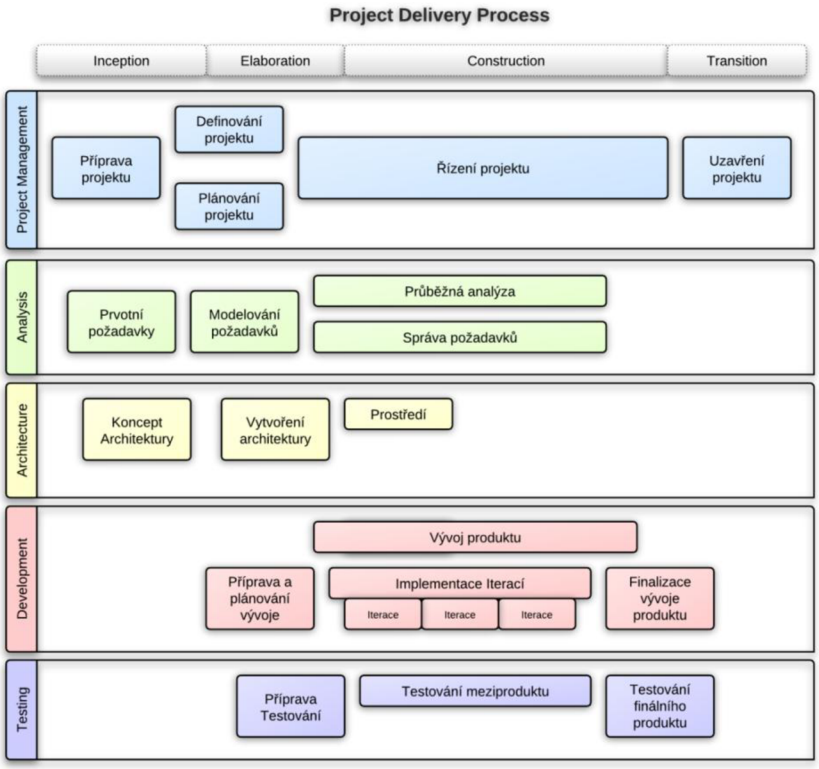
dodržování je vyžadováno. Podrobnou dokumentaci procesu včetně všech postupů a pod-procesů obsahuje interní wiki společnosti [16], z ní také vychází text celé následující kapitoly.

2.3.1 Procesní oblasti

Proces dodání projektu (Project Delivery Process, PDP) [16] obsahuje pět hlavních procesních oblastí, které pokrývají jednotlivé části řízení a vývoje SW. Jedná se o následující oblasti:

- Projektový management
- Analýza
- Architektura a Design
- Vývoj
- Testování

Jednotlivé oblasti se v průběhu projektu vzájemně prolínají a jsou provázány návazností svých vstupů a výstupů. Situaci ilustruje Obrázek 7.



Obrázek 7 Project Delivery Process [převzato z [16]]

Pro tuto práci je středem zájmu především první oblast, protože však cílem práce je dodání kompletního softwarového produktu, ani ostatní části procesu nelze ignorovat. Následující podkapitoly se detailněji věnují jednotlivým oblastem, zvláštní zřetel je pak kladen na první oblast. Veškeré zde uvedené informace stručně shrnují relevantní obsah interní wiki společnosti [16], pro kompletní verzi popisu PDP obsahuje Project Delivery Process.

2.3.1.1 Projektový management

Oblast projektového managementu vychází z mezinárodního standardu IPMA [3], čímž je dosaženo pokrytí všech základních oblastí řízení. Aplikaci standardu zajišťují projektoví manažeři s IPMA certifikacemi.

Aktivity v rámci managementu směřují k zajištění úspěšného splnění cíle projektu a jsou rozděleny do pěti oblastí:

- Příprava projektu – probíhá před zahájením, vymezuje se termín, rozsah a cena
- Definování projektu – domluva na detailech projektu, jeho průběhu a ukončení
- Plánování projektu – vytvoření detailního harmonogramu a dalších plánů
- Řízení projektu – stěžejní část, obsahuje kroky k zajištění splnění cíle projektu
- Uzavírání projektu – předání a vyhodnocení výsledků projektu

2.3.1.1.1 Příprava projektu

Příprava projektu probíhá v rámci před-projektové fáze a končí podepsáním smlouvy zahajující realizaci nebo případným ukončením projektu.

Základní otázky o potencionálním projektu zodpovídá **předprojektová rešerše**, a na jejím základě se také rozhoduje, zda projekt bude realizován. Pokud ano, stanoví se osoba zodpovědná za celou obchodní příležitost – manažer nabídky (bid manager, BM) – a navazují další aktivity příprav.

Důležitým krokem přípravy je získání **prvotních požadavků** zákazníka, a to jak funkčních, tak nefunkčních. Za tímto účelem jsou organizovány schůzky se zákazníkem a pořizovány zápisy. Získané požadavky jsou prioritizovány podle metody MuSCoW (Must, Should, Could a Won't Have požadavky, více viz [17]).

Na základě vytvořených podkladů vzniká **návrh řešení**. Jeho cílem je pokrytí zákaznických požadavků pomocí minimální konkurenceschopné varianty řešení, nejlépe se znovu-využitím existujících komponent. Součástí návrhu jsou také předpoklady (upřesnění neúplných informací, technických i netechnických) a uvedení případných nedodržených požadavků. Všechny výstupy a aktivity, které je nutné provést pro zhotovení výsledného produktu, jsou zachyceny do stromové struktury v produktovém rozpadu (**Work Breakdown Structure**, WBS). WBS tvořená během přípravy nemusí být detailní, ovšem napomáhá vymezení rozsahu projektu a je základem pro další plánování a odhadování.

Dalším krokem přípravy je **identifikace rizik**, tj. externích událostí, které mají určitou pravděpodobnost výskytu a daný pozitivní či negativní dopad. Protože kvantitativní posouzení je v této fázi projektu problematické, nahrazuje se kvalitativním (např. posouzení na škále malá, střední, vysoká pravděpodobnost).

S využitím návrhu řešení a prvotní WBS se provádí **prvotní odhad pracnosti**. Ten musí zohledňovat projektové role, aby bylo možné stanovit kritické cesty a náklady na projekt. Z odhadu pracnosti potom vychází **prvotní harmonogram** projektu, který slouží především pro orientaci v projektu a umožňuje jeho řízení. Jeho součástmi jsou základní fáze, milníky, body součinnosti a časový sled aktivit v projektu. Poslední chybějící položku trojimperativu – náklady a výnosy – zachycuje **kalkulace projektu**. Na základě tohoto rozpočtu je projekt hrazen, proto musí být výsledná kalkulace schválena odpovědnými osobami.

Informace o rozsahu, obsahu a předpokladech shrnuje **Logický rámec** (Logical frame, LF) projektu. Současně pomáhá definovat měřitelné výstupy a cíle a případně je přehodnotit. Opět není potřeba zacházet do detailů, ideálně by se měl dodržet formát stránky A4.

Všechny vytvořené výstupy prochází na závěr revizí a na jejím základě se rozhodne, zda bude podána nabídka. Pokud ne, příprava projektu se ukončí, pokud ano, navazují další více administrativní činnosti. První z nich je **příprava nabídky**, kdy se předchozí výstupy kompletují a doplňují, aby vznikl ucelený dokument splňující požadavky zákazníka na nabídku a zároveň orientovaný na zákazníkův byznys. Součástí nabídky bývá i **návrh smlouvy** vytvořený na míru konkrétnímu projektu dle aktuálních šablon společnosti a ideálně konzultovaný s právním zástupcem.

Pokud se podaří obchodní příležitost získat, končí příprava projektu podepsáním smlouvy. Tomu předchází diskuze, upřesňování detailů a případné úpravy smlouvy.

2.3.1.1.2 Definování projektu

Definování projektu začíná po schválení projektu a ukončení příprav, jedná se o kroky nutné pro úspěšné řízení. Obrazně řečeno se v této fázi definují pravidla hry.

Prvním krokem je **sestavení projektového týmu** nebo alespoň jeho jádra, které je potom doplňováno podle potřeb. Základní role v projektovém týmu jsou projektový manažer, vedoucí vývojář, analytik, architekt a tester. Součástí týmu jsou i další lidé na straně zákazníka – uživatelé, garanti požadavků, projektový manažer, atd. Kromě projektového (realizačního) týmu se vytváří ještě další dočasné organizační struktury: Řídící výbor projektu (rozhoduje o projektu) a Statutární orgán projektu (řeší výjimečné situace). Komunikaci mezi jednotlivými úrovněmi specifikuje **komunikační matice**.

Dále je potřeba připravit systémy pro realizaci projektu – vytvořit a nastavit **projektový kontejner**. Ten zahrnuje wiki stránky (znalostní báze) v Confluence[4], tiketovací systém JIRA pro evidenci úkolů, DMS pro ukládání dokumentace, systém pro verzování zdrojových kódů

[4] Jedná se o název konkrétního produktu využívaného ve společnosti, z tohoto důvodu není vhodný překlad a v práci zůstala zachována anglická verze.

(např. Subversion4, GIT4), SkillManager4 pro znalostní profily zaměstnanců a případně další systémy vyžadované projektem.

Jedním ze stěžejních faktorů úspěchu projektu jsou **zainteresované strany**, proto je potřeba tyto identifikovat, stanovit jejich požadavky a klíčové výstupy. Na základě těchto informací se potom volí vhodná komunikační strategie a případně přizpůsobení nabídky tak, aby více zaujala.

Hlavním dokumentem celého projektu je **Identifikační listina**. Ta stanovuje jeho hranice, způsob realizace i položky nutné k úspěšné realizaci. Typicky zahrnuje výstupy fází přípravy a definice a měla by být součástí smlouvy o dílo. Vzhledem k důležitosti tohoto dokumentu musí být oboustranně schválen a archivován.

Projekt je interně zahájen úvodní schůzkou (**kick-off meeting**), při které se členové projektového týmu seznámí s materiály k projektu, jeho cílem, zodpovědnostmi osob atd. tak, aby každý člen získal informace potřebné pro svou práci a samostatné rozhodování. Z pohledu zákazníka je startem **úvodní schůzka** klíčových osob, během níž se na základě Identifikační listiny shrne aktuální stav věcí a nastaví fungování projektu až do jeho předání a akceptace.

Samotná akceptace bude probíhat na konci projektu či iterace, a to na základě **akceptačních kritérií** stanovených během definice projektu pro každý z výstupů. Akceptací všech výstupů je potom dosaženo splnění cíle projektu. Kromě kritérií samotných se stanovuje i proces akceptace – způsob, jakým dojde k převzetí díla.

Nedílnou součástí projektu je ohodnocení výkonu týmu i jednotlivců. K tomu slouží udělování prémií, je tedy nutné stanovit si **rozpočet pro prémie** a kritéria úspěšnosti pro jejich přidělování.

2.3.1.1.3 Plánování projektu

Během předchozích kroků byly vytvořeny prvotní verze řady podkladů, které slouží pro řízení projektu. Během plánování projektu se tyto návrhy dále rozpracovávají a zpřesňují.

Jedním z upravovaných dokumentů je **WBS**, během plánování se prvotní strom výstupů a aktivit rozšiřuje o detailnější prvky. Na základě nové WBS je potom upřesňován také prvotní **odhad pracnosti**, a to až na úroveň konkrétního počtu člověko-dní a tedy i celkových interních nákladů projektu. Pracnost lze odhadovat dvěma způsoby: shora-dolů nebo zdola-nahoru. Ve společnosti se používá spíše druhá uvedená a to v následujícím postupu:

- Odhadne se implementační část pomocí metody PERT
    - Jednotlivé položky se ohodnotí optimisticky O, realisticky R a pesimisticky P
    - Vypočte se PERT hodnota každé položky:

$$H_p = (O + 4 * R + P)/\ 6$$

    - Hodnoty se sečtou

- Doplní se odhady dalších fází s ohledem na konkrétní projekt, jeho rozsah a rizika
    - Projektové řízení je typicky 30% implementace
    - Analýza je cca 20% implementace
    - Testování odpovídá cca 20% implementace
- Přidá se rezerva ve výši 30% celého projektu

Pro úspěšné řízení projektu je potřeba vytvořit **plán řízení projektu.** Ten stanovuje časový sled a vzájemné závislosti aktivit a typicky zahrnuje řadu dílčích plánů, které detailněji řeší konkrétní oblast řízení:

- Projektový harmonogram, milníky – včetně případných částečných úhrad
- Plán součinnosti – pro zákazníka, případně třetí strany
- Plán využití lidských zdrojů – kdo a kdy konkrétně bude na projektu pracovat
- Plán školení týmu – pokud je nutné získat nové kompetence
- Podrobné **iterační plány** – pro každou z iterací, obsahují počet člověko-dní, implementované případy užití a konkrétní osoby, které implementaci provedou
- Testovací plán – viz část 2.3.1.5 Testování
- Plán uživatelského testování (UAT) – termín akceptačního testování, lhůty pro změny
- Plán nasazení – kdy se bude projekt nasazovat na jaké prostředí (test, produkce, …)
- Plán školení zákazníka – zaškolení zákazníka, případně třetích stran

Při sestavování plánu se vychází z Prvotního harmonogramu, postupně se přidávají základní milníky a fáze. Ty se následně rozpracovávají do větších detailů, vytváří se dílčí plány atd.

Po dokončení detailní dokumentace a před zahájením implementace se musí finální rozsah **schválit** osobami s odpovídajícími pravomocemi. Ten se porovná s původní nabídkou, a pokud se významně liší, zahájí se změnové řízení ohledně dalšího postupu v projektu.

2.3.1.1.4 Řízení projektu

Projekt je nutné řídit průběžně po celou dobu jeho existence. V této kapitole ovšem chápeme řízení jako část managementu projektu mezi fázemi definování a ukončení.

Základní fázovou jednotkou projektu je **iterace**. Tato se řídí podle příslušného dílčího plánu iterace (tzv. mikroplánu), který je součástí celkového plánu a před zahájením iterace se rozpracovává do detailů. V případě nutnosti může rozpracování vyústit i ve změnu celkového plánu nebo změnové řízení. Výstupem iterace je meziprodukt projektu, ten se osobně prezentuje zákazníkovi a dle zpětné vazby se dále upravuje. Na závěr je meziprodukt buď schválen (a případně proplacen) nebo se zahájí změnové řízení.

Zmiňované **řízení změn** patří mezi nejdůležitější faktory úspěchu, bez něj se projekt stává nekontrolovatelným. Změnový proces musí být jasný a přehledný pro všechny zúčastněné strany. Při vzniku změny se provede kategorizace a následné posouzení orgánem

kompetentním k hodnocení změn dané kategorie. Změny obecně mají dopad na projektový trojimperativ, proto je nutné je vždy komunikovat se všemi zainteresovanými stranami.

Během realizace projektu vzniká řada **problémů** a ty je nutno evidovat a řešit. Jako vhodný nástroj může sloužit například sada tiketů v JIRA nebo seznam na wiki stránce projektu. Každý otevřený problém má stanoveného zodpovědného řešitele a termín pro uzavření takový, aby nedošlo k narušení plánu projektu.

Otevřené problémy mohou přejít až do rizik, proto je nutné je sledovat, upozornit na možné problémy a případně včas eskalovat výše. Také **rizika** samotná musí být řízena a to na základě pravidel stanovených v dokumentu Řízení rizik projektu. Zmíněný dokument je potřeba průběžně aktualizovat podle aktuálního stavu rizik. V průběhu projektu postupně získáváme detailnější informace k rizikům, díky tomu je můžeme kvantitativně analyzovat (tj. určit jejich finanční dopad):

$$financni\_dopad = pravdepodobnost\_vyskytu * dopad$$

Dopad vyjádřený ve finančním objemu je mnohem lépe představitelný pro zákazníka.

Další důležitou položkou jsou **náklady**. Projektový manažer je průběžně sleduje a posuzuje, zda aktuální rozpočet odpovídá plánu. Aktuální rozpočet se vypočte následovně:

$$R_a = MD * IV + N_d$$

Kde MD je počet aktuálně odpracovaných člověko-dní, IV je interní náklad na člověko-den a N_d jsou další náklady (externí služby, licence, atd.).

O aktuálním stavu projektu je potřeba informovat také řídící orgány projektu. K tomuto účelu slouží **reporty** obsahující fázi projektu, pozici v plánu, odchylky od plánu a případně další informace relevantní pro vedení.

Závěrečným a zároveň kritickým bodem řízení je **předání a akceptace projektu**. Úspěšné předání je základem pro proplacení celého projektu, případně u větších projektů jeho dílčí části. Akceptace probíhá na základě dříve stanovených akceptačních kritérií, ta zase často navazují na výsledky uživatelského akceptačního testování (User Acceptance Testing, UAT). Průběh akceptace z pohledu dodavatele bývá následující:

- Předání k akceptaci – předání díla a protokolu (včetně lhůty pro odpověď)
- Obdržení akceptačního protokolu – včetně seznamu vad proti specifikaci
- Odhad zdrojů pro další úpravy díla dle zaslaných výhrad zákazníka
- Schválení výhrad – určení, co se bude upravovat, návrh změnových požadavků
- Obdržení finálního podepsaného schváleného akceptačního protokolu
- Zaevidování akceptačního protokolu

2.3.1.1.5 Uzavírání projektu

Poslední fází managementu je uzavírání projektu. Za uzavření projektu je zodpovědný projektový manažer, který provádí všechny potřebné kroky, mj. aktualizace popisů souvisejících s projektem, zpracování případové studie a získání reference, vytvoření zpětné vazby a rozdělení prémií, kontrola dokumentace a její archivace nebo vyplnění reportu pro vedení.

V rámci uzavírání probíhají také případná **školení** zákazníka, díky kterým se zjednoduší po-implementační podpora (proškolení uživatelé reportují menší množství incidentů, navíc tyto dovedou lépe popsat).

Dočasná organizační struktura projektu se ruší, členové týmu se vracejí do svých stálých organizačních jednotek a aktualizují se plány zdrojů a nastavení přístupů. Pokud na realizaci navazuje záruka či podpora díla, musí se projekt **předat do Delivery Unit** (DU). To zahrnuje splnění podmínek pro předání, informování zástupce DU, předání komunikace ohledně servisní smlouvy a znalostí.

Důležitým krokem je finální neformální **vyhodnocení projektu**. Projektový manažer zhodnotí úspěšnost projektu, finanční aspekty, přínosy v know-how a v neposlední řadě také individuální přínosy jednotlivých členů.

2.3.1.2 Analýza

Cílem analýzy je pochopení toho, co a také proč má být implementováno. Zákaznické požadavky jsou v rámci této oblasti modelovány a výsledná specifikace je pak vstupem pro oblasti architektury a designu a vývoje.

Analýza je stěžejní činností na začátku, v menší míře probíhá po celou dobu běhu projektu. Průběžně řízen a kontrolován je rozsah projektu, kompletní návrh řešení je pak za běhu upřesňován a doplňován podle reálného stavu.

2.3.1.2.1 Zjištění požadavků

Cílem prvního kroku je sestavení seznamu požadavků zákazníka. Mohou nastat tři varianty:

a) Separátní analýza – analýza je odděleně placenou částí projektu, na základě jejích výstupů zákazník rozhodne o (ne)realizaci projektu.
b) Předem zpracované požadavky – zákazník má požadavky zpracované, provádí se jen jejich validace a odhad pracnosti.
c) Analýza součástí projektu – požadavky se analyzují až po podpisu smlouvy. Hrozí zde riziko nesouladu nabídnutého řešení a skutečných požadavků, částečně jej lze eliminovat stanovením předpokladů již v nabídce.

V závislosti na zvolené variantě jsou prováděny vybrané kroky z následujících:

- **Sběr požadavků**
  - Požadavky zákazníka se získávají s využitím standardních analytických postupů (interview, dotazník, analýza procesů, stávajícího SW, …).
- **Revize a schválení požadavků**
  - Získané požadavky se revidují, validují a oboustranně schvalují
- **Potvrzení odhadů pracnosti**
  - Na základě informací získaných z požadavků se zpřesní odhad pracnosti

Je vhodné pro každý požadavek stanovit tzv. garanta, tedy osobu na straně zákazníka, která formulovala daný požadavek a má zodpovědnost za jeho formu ve výsledném produktu.

2.3.1.2.2 Návrh systému

Požadavky získané v předchozím kroku se dále detailně analyzují na **analytických workshopech** a dalších schůzkách se zákazníkem. Na základě odsouhlasených zápisů z těchto schůzí se vytváří systémová specifikace. Je vhodné organizovat více častějších a kratších schůzek zaměřených na konkrétní problém a také zákazníka předem informovat o agendě. Pro minimalizaci rizika nedorozumění se zákazníkem se vytváří **slovník pojmů.** Jedná se o dokument obsahující abecedně řazené pojmy a zkratky, který je nutno průběžně aktualizovat.

Systémová specifikace může mít různé podoby, typicky je stanovena dle zvyklostí zákazníka. Nejčastější formou je popis chování a ovládání systému formou případů užití, některé druhy projektů mohou vyžadovat tvorbu dalších dodatečných diagramů.

Cílem celého návrhu řešení je vytvoření srozumitelné, kompletní a podrobné **specifikace požadavků**, která definuje, jak jsou v systému naplňovány požadavky zákazníka. Tento návrh musí popisovat, jaké komponenty a jak se využívají. Návrh řešení by měl primárně vycházet z existujících komponent a znalostí, snahou je navrhnout minimalistické řešení s orientací na přidanou hodnotu pro zákazníka. Pokud je vyžadováno sjednocení grafického uživatelského rozhraní (Graphical User Interface, GUI) s jinými systémy, musí být součástí specifikace pravidla pro tvorbu GUI.

Vytvořený dokument Specifikace prochází revizí PM a musí být schválen zákazníkem. Podle této detailní specifikace systému se také ověří a případně **zpřesní odhady pracnosti** tak, aby podle nich již bylo možné projekt závazně realizovat.

2.3.1.2.3 Průběžné doplňování návrhu

Zákazník má možnost průběžně sledovat vyvíjený produkt, připomínkovat jej a upřesňovat požadavky na produkt. Připomínky jsou vyhodnocovány a následně zapracovány do specifikace, zamítnuty nebo je zahájeno změnové řízení.

2.3.1.2.4 Správa požadavků na změnu

Při požadavku na změnu se provádí analýza jejího dopadu, výsledky se potom použijí pro změnové řízení. V případě schválení se prováděné změny se musí **zapracovat do specifikace** a nová verze musí být **schválena**. Všechny změnové požadavky se formálně evidují, aby byly zpětně dohledatelné.

2.3.1.3 Architektura a design

Procesní oblast architektury se zabývá technickou stránkou projektu. Snahou je navrhnout architekturu a komponenty systému tak, aby systém byl:

- rozšiřitelný, škálovatelný, robustní – splňující nefunkční požadavky,
- integrovatelný – snadno zabudovatelný do existující infrastruktury,
- udržovatelný – s pomocí standardních nástrojů a procesů a
- kvalitní – se zajištěním a kontrolou kvality produktu.

Z uvedených bodů je zřejmé, že architektura má silnou návaznost na procesní oblast vývoje.

Základní rysy architektury systému vznikají v počátečních fázích projektu, při odhadech složitosti řešení. V průběhu projektu se pak na základě analýzy požadavků zpřesňuje. Z pohledu architektury jsou stěžejní zejména tyto požadavky:

- funkční – integrace produktů třetích stran,
- nefunkční – počet uživatelů, dostupnost, bezpečnost, vyžadován konkrétní software.

2.3.1.3.1 Příprava konceptu architektury

Prvotní návrh architektury probíhá v raném stádiu projektu, často s omezeným množstvím podkladů. Vytváří se proto pouze základní architektura jako podklad pro komunikaci, odhad nároků a nacenění řešení. Stěžejním bodem je v tomto kroku identifikace fyzických a důležitých logických prvků a určení rozhraní s produkty třetích stran.

2.3.1.3.2 Vytvoření architektury

Architektura vychází z návrhu architektury s využitím dříve vytvořených komponent a ověřených konceptů. Pokud je potřeba ověřit nový koncept, vznikají **prototypy** - minimalistická řešení nepokrývající plnou funkcionalitu, ale pouze část potřebnou pro ověření. Součástí prototypu je dokumentace popisující cíl, postup a dosažený výsledek prototypu.

Pro vývoj produktu je zapotřebí technická infrastruktura, musí se **připravit prostředí**:

- Lokální – pro vývoj aplikace; pokud se připojuje k externím systémům je potřeba zajistit vývojové verze i u těchto systémů
- Vývojové – pro ověření aktuálně přidávané funkčnosti
- Testovací – pro otestování funkčnosti aplikace
- Produkční – pro nasazení u zákazníka

Všechna výše uvedená prostředí obsahují operační systém, SW a HW dle potřeb projektu a dokumentaci. Dále příprava prostředí zahrnuje konfiguraci nástrojů pro vývoj, viz projektový kontejner v kapitole 2.3.1.1.2 Definování projektu.

Aplikace je tvořena jednotlivými funkčními celky – **komponentami**. Ty jsou navrhovány během vývoje dané funkcionality, formálnost a důkladnost návrhu závisí na složitosti komponenty, její obecnosti a využití ostatními částmi systému.

Aktuální stav všech oblastí architektury, a to včetně případných změn vzniklých během vývoje, je obsahem **dokumentu architektury**. Reálně vytvořenou aplikaci z pohledu běžného provozu a administrativy pak popisuje **dokument skutečného provedení**.

V průběhu celého procesu jsou **identifikovány znovupoužitelné komponenty**, z hlediska architektury se může jednat například o skripty, popisy nástrojů, best practices, řešení problémů, výsledky prototypů atd.

2.3.1.4 Vývoj

Cílem oblasti vývoje je samotné vytvoření produktu podle specifikace vzniklé při analýze. Vývoj je rozdělen do iterací a postupně vznikající produkt je průběžně testován, automatizovaně sestavován a také reprezentován zákazníkovi. Pro neustálé zlepšování tohoto procesu je využíváno zpětné vazby členů týmu, revize a statická analýza kódu zase napomáhají vylepšení kvality samotného kódu.

2.3.1.4.1 Příprava a plánování vývoje

Prvním, zcela zásadním krokem vývoje je **představení cílů** projektu včetně důvodů, proč se daný projekt spouští, jeho rozpočtu, rámcových případů užití a hlavních účelů systému. Konkrétní případy užití již pak studuje každý člen týmu samostatně.

Následně dochází k rozdělení projektových rolí a příslušných zodpovědností, minimálně Vedoucího týmu (Team Leader), Vedoucího vývojáře (Developer Lead) a Manažera vývoje (Build manager).

Aby vůbec bylo možné vyvíjet, je potřeba **připravit vývojová prostředí.** Kromě prostředí uvedených v kapitole 2.3.1.3 Architektura a design se nachystá i prostředí pro sestavení (build prostředí, ve společnosti se používá systém Hudson). To slouží pro automatizované sestavování artefaktů projektu do produkčního balíčku.

Vývoj se řídí pravidly uvedenými v obecném **vývojářském manuálu**, případně ve vlastních dodatečných manuálech. To napomáhá zvýšení konzistence kódu, usnadňuje orientaci v cizích částech programu a také předchází implementačním konfliktům.

2.3.1.4.2 Vývoj produktu

Vývoj produktu probíhá ve více iteracích, uvedené kroky se proto cyklicky opakují pro každou z těchto iterací.

Iteraci zahajuje **úvodní schůzka**, na které jsou představeny případy užití určená k implementaci v této iteraci. Pro jednotlivé funkční celky jsou odhalována rizika a vytvořen odhad pracnosti. Výsledkem setkání je pak stanovený a komunikovaný rozsah iterace včetně jejího trvání.

Na základě odhadů pracnosti se vytvoří **harmonogram** pro danou iteraci včetně termínů pro klíčové události a rozdělí se úkoly na konkrétní vývojáře. Schválený harmonogram je uveřejněn na místě dostupném všem členům týmu (např. interní wiki stránky).

Vedoucí týmu přiděluje úkoly (tikety) vývojářům podle a na každodenních schůzkách týmu kontroluje jejich plnění. Vývojář při **implementaci** vychází z požadavků projektu a dodržuje dohodnutou architekturu. Současně také pro vylepšení kvality kódu vytváří **testy JUnit Frameworku** ke všem netriviálním funkcionalitám. Pro zajištění celkové kvality a udržovatelnosti aplikace, rozšíření znalostí členů týmů a usnadnění rozšiřování i rozsáhlého kódu slouží **revize kódu.** Během těchto vývojáři revidují přidělené části kódu, následně se provede společné vyhodnocení a případně se implementují nápravné kroky.

Z aktuálních verzí zdrojových kódů se průběžně **sestavují částečné produkty**. Primárně se využívá automatického sestavení podle příslušné konfigurace, případně pokud není možné automatické sestavení, vytvoří se popis pro ruční sestavení vývojářem. Tyto částečné produkty jsou následně testovány, což umožní rychlejší nalezení a opravy případných chyb.

Poslední částečný produkt iterace reprezentuje **iterační přírůstek.** Tento je nasazen na testovací prostředí pro ověření správnosti směru vývoje a kontrolu dodržení harmonogramu. Doporučuje se současně s nasazením také zamrazit kód (*code freeze*) a vytvořit novou větev ve verzovacím systému vývoje.

Vzniklý iterační přírůstek nutně vede také k **aktualizaci dokumentace**, zejména pak instalačních instrukcí pro zákazníka, dokumentu architektury, technické dokumentace a popisů komponent a instalační instrukce pro vývojáře.

Na závěr iterace probíhá seznámení týmu s výsledky a vyhodnocení dosažených cílů. Připomínky k výslednému produktu se zaznamenávají jako úkoly pro další iterace. Taktéž se zaznamenává zpětná vazba týmu k průběhu iterace a její vyhodnocení.

2.3.1.4.3 Finalizace vývoje produktu

Z aktuálních verzí zdrojových kódů je sestaven balíček *final package,* který se **nasadí k zákazníkovi** jako finální akceptovaný produkt. Zároveň se provede **revize dokumentace**, zda odpovídá skutečnému řešení a případně se zaktualizuje. Podle šablon společnosti (případně zákazníka) se také vytváří **uživatelská příručka**. Ta slouží koncovým uživatelům jako manuál

pro práci s aplikací a typicky obsahuje shrnutí, účel aplikace, jaké funkce nabízí a popis všech jejích částí.

Než dojde k ukončení projektu a **předání produktu podpoře** na oddělení Delivery, je do týmu vývojářů přidán člen Delivery Unit (DU), aby se seznámil s projektem a získal představu o tom, co bude přebírat. Tento člen se může podílet na vývoji, účastnit revizí kódu nebo prezentace produktu, apod.

Po ukončení se ještě identifikují **komponenty znovupoužitelné** pro další projekty, u vývoje jde například generické třídy, jsp komponenty, javascriptové komponenty a další.

2.3.1.5 Testování

Cílem testování je zjištění úrovně kvality produktu a její zvyšování. Vždy se jedná o kreativní činnost přizpůsobenou konkrétnímu projektu a jeho kontextu. Z pohledu zajištění kvality (quality assurance) se jedná jen o jednu ze složek vedoucí k celkovému ověření a zajištění kvality.

2.3.1.5.1 Příprava Testování

Před samotným testováním se musí tester **seznámit s požadavky projektu** a jeho kontextem. Studiem dostupných zdrojů zjišťuje funkční i nefunkční požadavky relevantní pro testování a současně ověřuje, zda je dokumentace srozumitelná a produkt dobře testovatelný.

Na základě zjištěných požadavků se definuje **testovací strategie**, tedy jak bude testování probíhat. Ta se dále detailně rozpracuje a vzniká **testovací plán,** který již obsahuje detailní popisy kdo, kdy, co, s jakým cílem a jak bude testovat, případně co naopak testovat nebude. Podoba plánu se přizpůsobuje se potřebám projektu a je nutné jej průběžně aktualizovat.

2.3.1.5.2 Testování meziproduktu

Prvním krokem testování je **příprava testovacích scénářů.** Testování funkčních požadavků obvykle vychází z případů užití, způsob ověření nefunkčních požadavků závisí na uvážení testera. Známý formát scénáře – sled kroků s uvedeným vstupem a očekávaným výstupem – bývá drahý a náročný na údržbu, navíc pro projekty s jedním testerem obvykle nepotřebný. Z těchto důvodů jsou preferovány jednodušší formáty. Ke scénářům je vhodné přidávat testovací data a případně postupy vedoucí k chybám.

Během iterace tester kontroluje všechny implementované části na základě testovacích scénářů, dokumentace, požadavků na funkčnost, použitelnosti, atd. Součástí jsou i destruktivní testy, tedy snaha danou funkcionalitu zničit, nabourat atd.

Během testování může tester narazit na jeden za dvou základních typů nedostatků:

- **Defekt (bug),** kdy implementovaná funkcionalita neodpovídá specifikaci,
- **Problém s použitelností**, kdy má návrh na její vylepšení (improvement).

V obou případech zakládá nový tiket příslušného typu s popisem. V případě defektů zkouší tester po nasazení opravy znovu reprodukovat nahlášený bug, pokud se mu to podaří, znovu jej pošle k opravě, jinak uzavírá příslušný tiket. U kontroly implementovaného návrhu ke zlepšení použitelnosti tester kontroluje funkčnost a použitelnost úpravy.

Během celého cyklu testování probíhá také **regresní testování**, tedy ověření, zda úpravou kódu nedošlo k zavlečení nové chyby do dříve testované části aplikace. Na základě požadavku zákazníka může být provedeno také **výkonnostní testování**, ověřující např. odezvu, počet možných paralelně přihlášených uživatelů apod. Nepovinnou položkou jsou **automatizované testy** s využitím systému Selenium.

2.3.1.5.3 Testování finálního produktu

Na **konci iterace** se znovu ověřuje veškerá funkcionalita implementovaná v příslušné iteraci, především pozitivními testy a ověřením typických scénářů. Pokud následuje testování u zákazníka nebo uvedení do produkce, testuje se i funkcionalita z předchozích iterací.

Z chyb nalezených v tomto kroku opravují pouze ty velmi závažné (tzv. showstoppers), aby se omezilo riziko regresního zanesení chyby. Ostatní chyby se zanesou do seznamu známých chyb.

Pokud to zákazník požaduje, může být součástí testování vytvoření **uživatelské příručky**, zrealizování **školení** pro uživatele systému nebo **vytvoření UAT scénářů** pro ověření, produkt splňuje akceptační kritéria. Forma a rozsah všech tří položek závisí na konkrétním projektu a požadavcích zákazníka.

2.3.1.5.4 Uzavření testování

Závěrem testování je nutné **vyhodnotit jeho průběh**, provést aktualizaci interních dokumentů o poznatky vylepšující testování v budoucnu a vyhodnotit metriky. Artefakty a znalosti získané během testování se předávají společně s projektem do oddělení zajišťující post-implementační podporu.

2.3.2 Životní cyklus projektu

Z časového hlediska je proces rozdělen do čtyř významných fází: Inception, Elaboration, Construction a Transition. Toto členění vychází z metodiky RUP. Standard IPMA naproti tomu definuje pouze tři fáze – předprojektovou, projektovou a po-projektovou. Toto třífázové členění není v rozporu s verzí RUP, jedná se pouze o odlišné seskupení aktivit.

Úvodní fáze – **Inception** – slouží především k seznámení se s projektem, určení jeho obsahu a rozsahu a stanovení jeho cílů. V tomto kroku se definují základní parametry projektu, jako jsou projektová pravidla a metody, postupy kontroly kvality, dokumentační pravidla,

harmonogram, akceptační kritéria, projektový tým a komunikační plán. Současně jsou identifikovány rizikové faktory a zvoleny postupy změnového managementu.

Druhá fáze – **Elaboration** – detailněji rozpracovává požadavky zákazníka získané v předchozím kroku, probíhá podrobná analýza projektu, návrh systému včetně jeho architektury, dokončují se inicializační práce a je vytvořen cílový koncept díla. Prvotní artefakty jako je plán, harmonogram a odhad pracnosti, se dále zpřesňují, vytvářejí se prototypy pro ověření funkčnosti konceptu a snížení technologických rizik projektu.

Cílem této fáze je přesněji definovat, jak bude tvořen výstup projektu. Na konci by měl být stanoven konkrétní postup prací, který umožní začít se samotnou implementací.

Fáze **Construction** přímo realizuje požadovaný produkt a to na základě předchozí analýzy a návrhu. Fáze bývá rozdělena do více vývojových etap (iterací), výstupem každé z nich je meziprodukt prezentovaný zákazníkovi. Po několika iteracích vzniká beta produkt, který je detailně interně testován a následně předán k uživatelskému akceptačnímu testování (UAT).

Poslední fáze **Transition** zahrnuje předání a uvedení produktu do provozu. Na základě UAT se dolaďuje funkcionalita, optimalizuje se výkon a testuje bezpečnost. Zároveň se dokončuje se veškerá dokumentace. Na závěr je dílo akceptováno a finálně předá.

Pro úspěšné naplnění cíle projektu je v jeho průběhu vyžadována součinnost ze strany zákazníka ve formě vyhrazení času a dalších zdrojů. Její rozsah závisí na povaze projektu a možnostech zákazníka.

2.4 Řízení malých projektů

Postupy řízení projektů se většinou nezabývají velikostí samotného projektu, bývají aplikovatelné univerzálně. U velmi malých projektů se ovšem může stát, že režie spojená s řízením převýší ostatní práci na projektu, a zbytečně jej tak prodraží či prodlouží [5]. Protože cílem práce je právě projekt malého rozsahu, zaměříme se v této kapitole na některá specifika a možná zjednodušení řízení u malých projektů.

2.4.1 Specifika malých projektů

Posouzení, zda je konkrétní projekt malý nebo naopak, může být značně individuální. Záleží pak především na zkušenostech organizace a porovnání s ostatními projekty v ní. Obecně lze říci, že ve srovnání s běžnými projekty malé projekty bývají jednodušší, kratší a spotřebovávají méně zdrojů, (finančních i personálních) a často mívají nízkou prioritu. Také pravděpodobnost, že se dostanou do potíží, je menší. Na druhou stranu i malé projekty je potřeba řídit a sleduje se u nich dosahování trojimperativu [18].

Uvedené charakteristiky malých projektů přináší řadu výhod, ale také řadu problémů. Přínosem **menšího týmu** může být například lepší vzájemná znalost jeho členů, možnost neformální komunikace, lepší atmosféra apod. Na druhou stranu potřební odborníci se často nevěnují naplno pouze jednomu projektu, ale podílejí se částečnými úvazky na více projektech. Vznikají tak prostoje, kdy se odborník musí znovu „zasvětit" do daného problému [18].

Kratší doba trvání může zlepšit odhady trvání, ovšem neumožňuje zahrnutí větších rezerv a i malé zpoždění může mít velké následky [18], poměrově k celkové době trvání je totiž významnější. Také režie spojená s rozběhnutím projektu zabírá poměrově větší část celkového času a je tedy na schopnostech projektového manažera, aby zajistil dodržování plánu.

U menšího projektu se dá očekávat také **menší rozpočet.** Ten obsahuje méně položek, lépe se tak sestavuje. Problém ale nastává u rezerv, malý objem financí neumožňuje zahrnout velké rezervy a v případě potřeby překročení původního rozpočtu není manévrovací prostor dostatečný [18]. Současně narůstá poměr nákladů vynaložených na řízení projektu, náklad na člověko-hodinu projektového manažera je typicky vyšší než u běžných pracovních pozic a pracnost řízení projektu může snadno překročit běžně kalkulovaných 20% z celkové pracnosti [16].

Poslední uvedené specifikum je **priorita** projektu. Ta bývá u malých projektů často nízká, což sice může snižovat případné dopady rizik, ovšem zase nezřídka dochází k situacím, kdy projekt nedostane potřebné zdroje, protože jsou alokovány pro jiný, významnější projekt. Takové situace vedou ke stavu, kdy míra priority koresponduje s mírou úspěšnosti dokončení. Malá priorita také mnohem méně motivuje pracovníky a to jen přispívá k celkovým problémům malých projektů [18].

Jedním z velkých problémů malých projektů je jejich tendence k narůstání [18]. Řada věcí se zpočátku jeví jako velmi jednoduchá, postupem času se však nabalují další požadavky a ve výsledku projekt hrubě přesáhne svůj plán. Problémy související se specifiky malých projektů uvedené výše se pak stávají výrazně palčivější.

2.4.2 Možnosti řízení

Základy řízení malých projektů se principiálně neliší od běžných projektů, ovšem při jejich aplikaci je potřeba zohlednit potřeby a především možnosti malého projektu. Odborná literatura se v zásadě shoduje na faktu, že běžné postupy je potřeba zjednodušit, odpadá potřeba některých výstupů apod.

Zjednodušené řízení zahrnuje celou řadu činností, například se může jednat o následující posloupnost kroků [19]:

1. Stanovení cílů, hypotéz
2. Seznam všech činností

3. Přirozená seskupení
4. Časový rozvrh
5. Rozpočet finanční
6. Funkční schéma
7. Lidé
8. Umístění (zodpovědnost)
9. Organizační schéma
10. Delegování
11. Realizace a kontrola

Uvedený seznam zahrnuje základní kroky dostačující pro malý projekt, ovšem ve velmi stručné podobě. Jednotlivé body je proto vhodné rozvést, minimálně poslední bod vyžaduje větší doplnění.

Jednou z možností řízení realizace je využití formuláře [18]. Ten svým způsobem nahrazuje jednotlivé druhy plánů a pomáhá manažerovi udržovat přehled o základních vlastnostech a stavu projektu. Zapisují se zde jednotlivé položky projektu, jejich časové plnění a náklady, případně další položky dle potřeby. Tento přístup je velmi jednoduchý, nevyžaduje speciální programové vybavení (postačí i tabulkový procesor) a pro velmi malé projekty může dostačovat. Na druhou stranu formulář zachycuje jen fázi řízení v průběhu projektu, nepokrývá oblasti rizik, zahájení projektu apod., je tedy potřeba jeho doplnění dalšími dokumenty. Kombinací s předchozím seznamem dostaneme jednoduchou a efektivní variantu řízení pro malé projekty.

Uvedený postup je jen jedním z možných kroků, lze z něj však vyvodit, které části jsou nevyhnutelné nezávisle na velikosti projektu – cíle a aktivity; čas; lidé a finance (obecně zdroje). Zjistíme tak, že se opět jedná o známý projektový trojimperativ, který musí být dosahován a řízen.

2.4.3 Použití standardů

Stručný přehled často používaných standardů obsahovala kapitola 2.2 Standardy projektového řízení**,** nyní se však zaměříme více na jejich použitelnost a vhodnost pro malé projekty. Aplikace standardu a tedy exaktní řízení snižuje rizika projektu [5], ovšem může velmi snadno narazit na některé z omezení malých projektů a ve výsledku se tak stát spíše nevhodným.

Standard **IPMA** neuvádí konkrétní procesy a praktiky, ale zaměřuje se na obecné dovednosti manažera [9]. Obecně tak můžeme říci, že jej lze aplikovat univerzálně bez ohledu na velikost projektu.

Na rozdíl od IPMA standard **PRINCE2** je orientován procesně a u malých projektů může nastat situace zmíněná v předchozím textu – příliš velký poměr řízení k ostatním částem

projektu. Pro použití je proto potřeba jej přizpůsobit. Doporučení takových úprav nalezneme přímo v literatuře *PRINCE2 pro řízení malých podniků* [5], kde je uveden koncept tzv. **škálovací matice**. Tato na základě splnění podmínek či vlastností projektu určuje, které elementy řízení jsou nutné a které doporučené. Kniha obsahuje popis neredukovatelného minima, tedy toho, co je základem pro řízení a je potřeba vždy. K ostatním elementům jsou zde zase uvedeny možnosti jejich zjednodušení – například nahrazení Ganttova diagramu kontrolním seznamem, snížení úrovně formálnosti záznamů rizik či použití kritérií kvality místo komplexního plánu kvality. PRINCE2 tedy lze využít i pro malé projekty a to v jeho zmenšené podobě.

Analogicky k PRINCE2 se standard **PMBOK** věnuje konkrétním procesům, které by mělo projektové řízení zahrnovat. Znovu zde tak narážíme na problém příliš rozsáhlého řízení a řešením může být opět úprava standardu – méně iterací, zjednodušení procesů případně jejich výstupů. Bohužel v případě PMBOK musí zjednodušení provést sám projektový manažer na základě svých vlastních posudků a zkušeností, žádná dostupná literatura se totiž tomuto tématu blíže nevěnuje. Dá se však předpokládat, že by šlo využít stejný přístup jako v případě PRINCE2, tedy návrh škálovací matice, která by posuzovala podstatnost jednotlivých částí na základě vlastností projektu.

Z výše uvedeného přehledu vyplývá, že použití procesních standardů je dobrým odrazovým můstkem při řízení malých projektů, vyžaduje však jejich uzpůsobení. Naproti tomu standard IPMA lze v podstatě využít v nezměněné formě.

3 Tiketovací systém JIRA

Jak bylo uvedeno výše, cílem projektu je vytvoření zásuvného modulu pro komerční tiketovací systém JIRA společnosti Atlassian. Tato kapitola shrnuje základní technický popis JIRA, potřebný pro pochopení její funkčnosti a navržení rozšíření.

Systém JIRA je jakýmsi sledovačem projektu (project tracker) pro jednodušší řízení a práci s projekty. Umožňuje zachytit a organizovat úkoly (issues) libovolného typu podle potřeb projektů, díky čemuž usnadňuje týmovou práci. Nabízí také řadu pokročilých způsobů práce s úkoly: jejich označování štítky, automatické zpracování pomocí workflow, plánování, sledování, pokročilé vyhledávání, generování reportů, automatická upozornění a řadu dalších funkcionalit [20].

3.1 Tikety

Stěžejní součástí JIRA jsou tikety – jakési virtuální lístky s popisem úkolu. Existuje mnoho typů tiketů, například chyba (bug), úkol (task), objednávka (order) či pod-úkol (sub-task). Volba typu závisí především na požadavcích uživatele, jaký typ reálného úkolu chce zaznamenat. Díky tomuto může JIRA sloužit jako systém pro hlášení chyb stejně dobře jako nástroj pro vyřizování objednávek zákazníka.

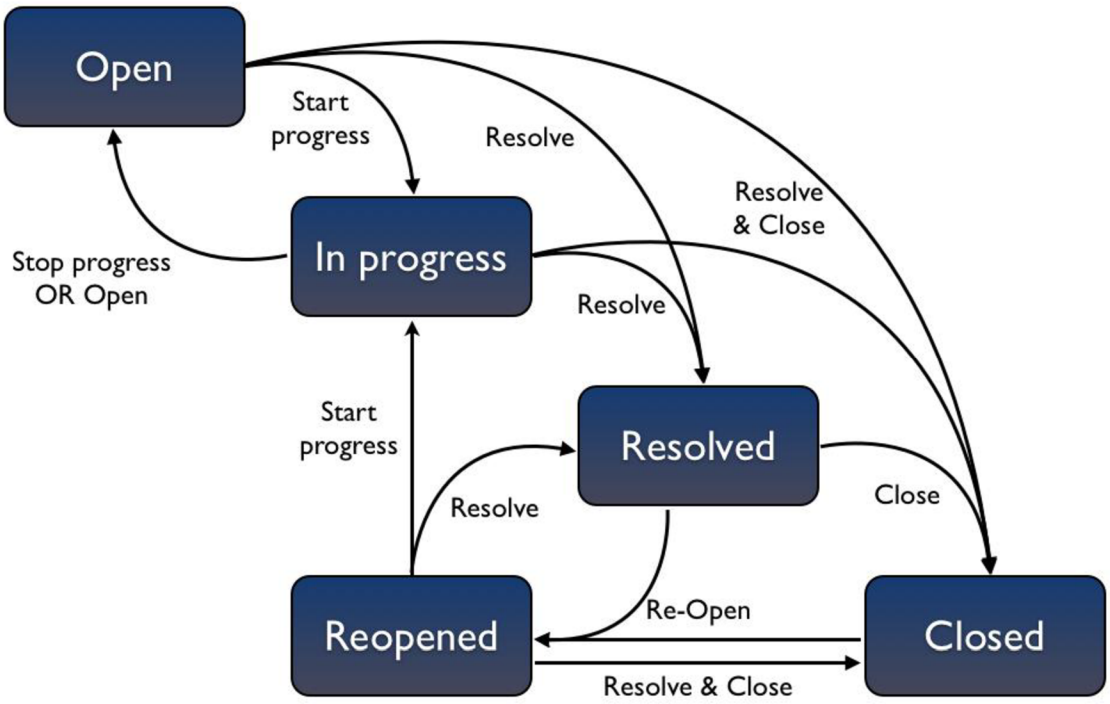
V závislosti na svém typu obsahuje tiket různá pole s datovým obsahem, například Projekt, Název, Popis, Prostředí, Priorita, Řešitel. Zadavatel atd. Obsah polí má určen datový typ a případná omezení. Taktéž je možné nastavit viditelnost polí v závislosti na přihlášeném uživateli a jeho oprávněních k práci s konkrétními tikety.

Jednotlivé tikety lze označovat štítky, díky tomu se lze v systému lépe orientovat a vyhledávat související úkoly. Štítky jsou společné pro všechny uživatele, nelze si nastavovat své soukromé.

3.1.1 Workflow

JIRA podporuje tvorbu workflow (WF), tiket tedy prochází různými stavy (status) přes dané přechody (transitions) [20]. Stavy a přechody výchozího workflow systému ilustruje Obrázek 8. Při změně stavu mohou probíhat automatické změny tiketu, ty se do systému přidávají jako funkce navázané na přechod mezi stavy.

Systém umožňuje úpravy existujících WF i tvorbu nových. Na konkrétní projekt a volitelně i na konkrétní typ tiketu v projektu je vždy před schéma (WF Scheme) navázáno jedno z dostupných WF.



Obrázek 8 Systémové workflow JIRA [převzato z [21]]

3.1.2 Přiřazování tiketů

Každý existující tiket obsahuje pole Řešitel (*Assignee)*, ve kterém bývá přiřazen konkrétní uživatel řešící problematiku popsanou v tiketu, případně je toto pole prázdné. Uživatel musí mít oprávnění přiřaditelný (*assignable)* pro projekt, ke kterému tiket patří, aby mu mohl být tiket přiřazen.

Přiřazení tiketu probíhá dvěma různými způsoby – automaticky nebo ručně. První automatické přiřazení se volá ve chvíli vytvoření tiketu, kdy systém vyplní pole řešitele přednastavenou hodnotou. Ve výchozím stavu je jako výchozí řešitel nastaven vedoucí projektu (role *team lead*). Pokud se v nastavení JIRA povolí existence nepřiřazených tiketů, může být výchozí hodnota řešitele prázdná (*unassigned*). Jiné varianty než tyto dvě výše uvedené základní instalace JIRA bez rozšíření nenabízí.

Dalším okamžikem, kdy se může automaticky měnit řešitel tiketu, je změna stavu tiketu pomocí funkce v rámci workflow. Při změně stavu lze automaticky měnit některý obsah tiketu, tedy i pole *assignee.* Výchozí workflow toto pole nemění, pokud jsou nějaké změny požadovány, musí se workflow ručně změnit nebo vytvořit nové a přidat post funkce aktualizující pole řešitele. Tiket může být při změně přiřazen jakémukoliv uživateli s oprávněním přiřaditelný.

Analogií funkce ve workflow je reakce na některou událost v tiketu pomocí programového naslouchače (listener), více viz kapitola 3.2 Události.

Druhou možností je přiřazení tiketu ručně. Tuto změnu mohou provádět všichni uživatelé, kteří mají v rámci daného projektu oprávnění přiřazovat tikety (*assign isuue),* typicky například vedoucí projektu. Tiket může být přiřazen některému z uživatelů s oprávněním *assignable,* opět v rámci daného projektu. To bývají typicky uživatelé v projektové roli vývojář. Tento způsob přiřazování je flexibilnější než automatická verze, vyžaduje však cenný čas pracovníků.

Na webové stránce Atlassian MarketPlace lze nalézt rozšiřující modul User Picker From Project Role - Issue Alternative Assignee [22]. Ten umožní přiřadit tiketu automaticky i jiné uživatele než jen vedoucího projektu. Výběr probíhá jako post funkce na základě hodnoty přidaného pole (*custom field*) nebo podle role – aktuálně přihlášený uživatel, zadavatel tiketu nebo vedoucí vývojář. Zásuvný modul je k dispozici zdarma a jeho nastavení sice rozšiřuje možnosti přiřazení tiketů, ovšem stále se jedná o nastavení konkrétní pevné hodnoty.

3.2 Události

Všechny produkty Atlassianu pracují s událostmi (events) [23], tedy se signály vytvářenými a automaticky rozesílanými při změnách v systému. Jednotlivé události lze zachytávat a reagovat na ně, díky tomu je možné synchronizovat různé části systému.

Reagovat na událost může naslouchač (listener), tedy taková komponenta systému, která provede svou registraci k odposlouchávání událostí. Součástí komponenty je pak programový kód, který odchytává událost, zpracovává ji a provádí požadovanou akci. Některé naslouchače jsou přímo součástí systému a zajišťují část jeho základních funkcí, ovšem pro rozšíření funkcí je možné přidat naslouchače vlastní.

Existuje řada druhů událostí, základní systémové jsou společné pro všechny produkty, řada dalších je pak vázána na konkrétní produkt a typ jeho obsahu (wiki stránky, tikety, akce uživatelů, …).

Jednou ze skupin událostí specifických pro JIRA jsou události tiketu. Jsou implementovány třídou API *IssueEvents* [21] a jedná se o události vytvoření, aktualizace, přidání nebo změny komentáře, přiřazení řešiteli, změny stavu ve workflow, přesun apod. Díky těmto událostem je možné automaticky reagovat na prakticky všechny změny v tiketech, nejen na přechody ve workflow.

Analogicky lze také reagovat na události jiných typů (např. vytvoření projektu, přihlášení uživatele, …), což funkcemi ve workflow nelze ošetřit vůbec.

3.3 Zásuvné moduly

Systém JIRA stejně jako ostatní aplikace společnosti Atlassian podporuje rozšiřování či změny funkcionality pomocí zásuvných modulů. Širokou nabídku komerčních i volně dostupných modulů je možné stáhnout z oficiálních stránek Atlassian MarketPlace [24]. Kromě těchto hotových rozšíření se nabízí možnost vytvořit si modul vlastní s téměř jakoukoliv funkcionalitou.

Každý zásuvný modul prochází životním cyklem svázaným s událostmi v systému [23]. Do tohoto cyklu zasahují následující události (stadia):

- Inicializace komponenty – při startu modulového systému, instalaci modulu nebo jeho povolení
- Destrukce komponenty – událost komplementární k předchozí
- Události modulového systému
    - *PluginEnabledEvent* a *PluginModuleEnabledEvent* – po inicializaci
    - *PluginDisabledEvent* a *PluginModuleDisabledEvent* – při destrukci
    - LifecycleAware#onStart() – po startu, pro veřejné části modulu
    - Úlohy aktualizace zásuvných modulů – při upgrade verze systému
- JIRA události – např. start celého systému, nejsou primárně určeny pro moduly

Na jednotlivé události lze v zásuvném modulu reagovat a zajistit tak chování konzistentní se systémem jako celkem.

Jednotlivé zásuvné moduly lze do systému přidávat, odebírat případně je aktivovat či deaktivovat. Všechny tyto činnosti může provádět pouze uživatel s rolí systémového administrátora případně uživatel s přidanými právy pro práci s moduly. Instalace modulů probíhá přímo přes správcovský panel v JIRA [25] a to pro moduly dostupné na webu Atlassian Plugin Exchange i pro vlastní moduly ve formátu JAR. Kromě automatické instalace je možné také ruční nahrání modulu, doporučen je ovšem první způsob [20].

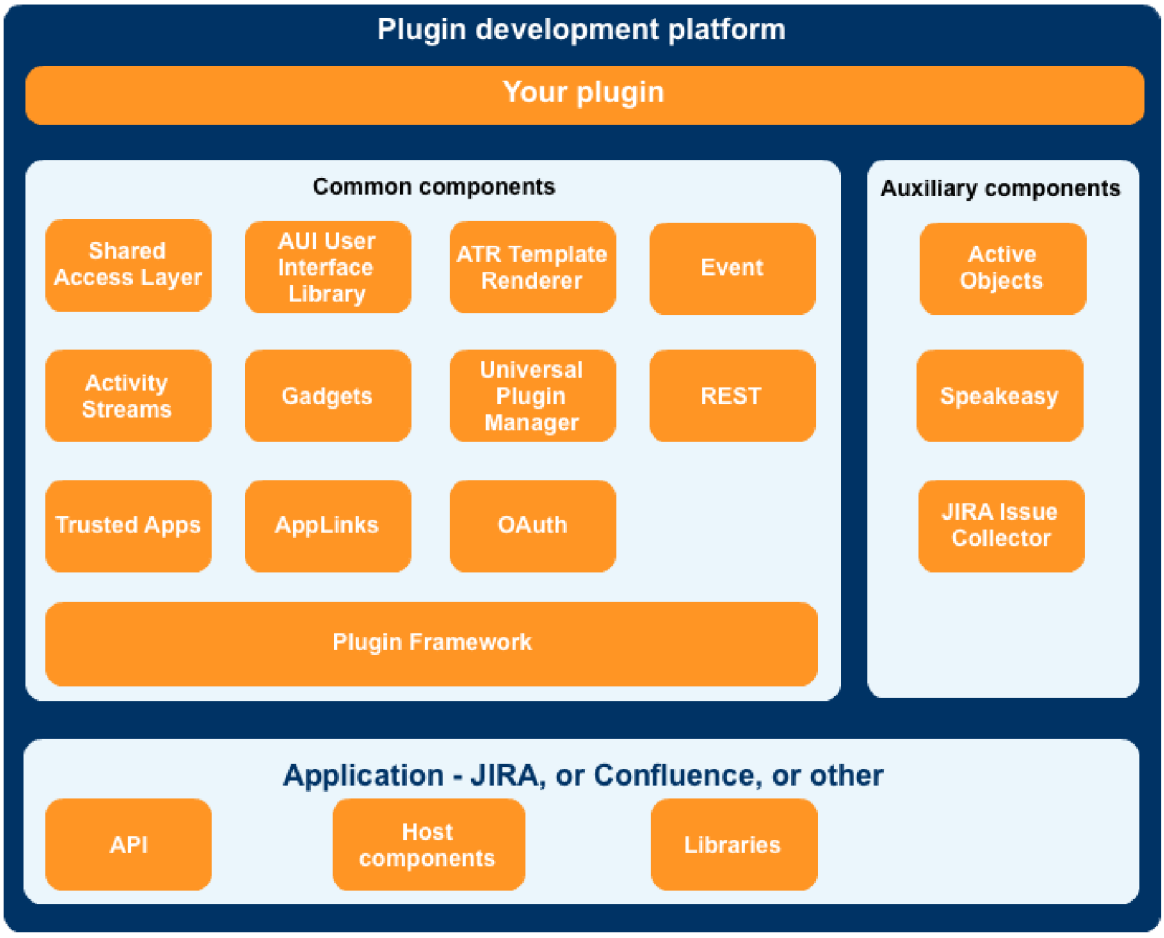
3.3.1 Vývoj modulů

Atlassian nabízí celou škálu komponent pro vývojáře, souhrnně označovanou jako Plugin Development Platform PDPl (přehled viz Obrázek 9) [23]. Vývojová platforma zpřístupňuje zásuvným modulům řadu vnitřních součástí a funkcí systému, díky čemuž lze vestavěnou funkcionalitu využívat ve vlastním kódu a dále ji rozšiřovat.

Základní společné a pro nás zajímavé komponenty PDPl jsou:

- Shared Access Layer (SAL) – API pro přístup k základním službám společným pro všechny produkty Atlassianu

- Atlassian User Interface (AUI) – JavaScript a CSS komponenty uživatelského rozhraní
- Atlassian Template Renderer (ATR) – API pro vykreslování textového kontextu
- Atlassian Event – knihovna pro práci s interními zprávami
- Activity Streams – API pro práci se streamy aktivit
- Universal Plugin Manager (UPM) – utilita pro rychlou práci se zásuvnými moduly
- Plugin Framework – Framework, ve kterém běží samotné zásuvné moduly
- Active Objects – modul pro ukládání dat



Obrázek 9 Atlassian Plugin Development Platform [převzato z [23]]

Kromě společných jsou obsaženy ještě aplikačně specifické součásti, v našem případě JIRA API, JIRA Host components a JIRA Libraries. Ty obsahují funkce specifické pro daný produkt, u JIRA je to například práce s tikety.

Pro samotné psaní zdrojových kódů je dostupný vývojový kit pro jazyk Java – Atlassian Plugin SDK, který obsahuje nástroje potřebné pro vývoj modulů. Samotné vývojové prostředí Javy je nutné nainstalovat zvlášť, stejně tak i některé potřebné knihovny. Návod lze nalézt na vývojářské wiki Atlassianu [23].

SDK obsahuje řadu utilit umožňujících snadnou tvorbu kostry modulů (příkazy atlas-`create-jira-plugin` a `atlas-create-jira-plugin-module`), překlad a testování ve vývojovém verzi JIRA (`atlas-run, atlas-compile, atlas-cli`, …), finální sestavení pro instalaci do JIRA (`atlas-package`) a řadu dalších užitečných nástrojů. Kromě toho zahrnuje aplikace třetích stran využívané výše uvedenými příkazy, např. Maven pro sestavování kódu (`atlas-mvn`).

Instalaci zásuvného modulu do vývojového systému lze provádět i za běhu, díky kombinaci UPM a utility FastDev. První zmíněná umožňuje dynamické vypínání/zapínání modulů, druhá potom provádí všechny potřebné kroky pro instalaci, od deaktivace staré verze přes překlad, nahrání nové verze až po její aktivaci. Toho lze s výhodou využít při vývoji, nahrání nové verze nevyžaduje restart celého systému.

3.3.2 Active Objects

Pro ukládání dat v zásuvných modulech se vždy využívá databáze, nad kterou je napojen celý systém JIRA. Pro vložení uživatelských dat lze využít tři koncepty [26]:

- Vložení nových entit hrubou silou
- Uložení v tabulce vlastností
- Uložení v samostatných objektech databáze v JIRA

První varianta je velmi nevhodná, jedná se o hrubý netransparentní zásah do databáze bez garance výsledků. Navíc v případě upgrade systému je nutno znovu provést tutéž úpravu a nahrát data, což je z hlediska dlouhodobého využití nepraktické a špatně udržovatelné.

Druhá verze spočívá v ručním přidání hodnot do existujícího objektu v databázi – propertyentry table [27]. Tento objekt je vytvářen automaticky systémem a je využíván právě pro ukládání nastavení. Tato metoda je vhledem ke svým možnostem vhodná spíše pro malé množství informací co nejjednoduššího typu. Obecně se jedná o zastaralý způsob a jeho použití není doporučováno.

Poslední možnost netrpí nedostatky uvedenými v předchozích odstavcích. Pro přístup do databáze využívá specializovanou komponentu rozhraní Active Objects.

Active Objects (AO) je doinstalovatelná komponenta systému JIRA [28], která umožňuje transparentní práci s databází. Jedná se o vrstvu, která abstrahuje příkazy a dotazy konkrétního typu databáze a umožňuje jejich volání přes obecné metody, shodné pro všechny podporované verze databází. Díky tomu jsou zásuvné moduly s AO databázově nezávislé.

Pro každý z modulů využívajících AO se vytváří samostatný datový prostor, sdílení dat mezi více moduly přes databázi tak není možné. Na druhou stranu tento přístup zvyšuje bezpečnost dat, není možné neoprávněně získat či přepsat cizí data. Taktéž oddělení prostorů předchází případným konfliktům při shodném pojmenování různých položek ve více modulech [27].

Pro použití AO v zásuvném modulu je nutné povolit příslušnou komponentu v nastavení a vytvořit v zásuvném modulu modul Active Objects. Následně už můžeme vytvářet rozhraní a implementační třídy pro práci s entitami a využívat metody nabízené AO. Podrobnější postup lze nalézt v dokumentaci AO [28].

Kromě základního volání metod pro vytvořené entity umožňují AO i nastavení některých běžných omezení datových hodnot, například "*not null*", "*unique*" apod. Jedná se však pouze o nastavení v databázi, před vložením není prováděna kontrola, zda data podmínce vyhovují. V případě chybného formátu či obsahu je vygenerována SQL výjimka. Je proto nutné s tímto faktem počítat a data ověřit programově, případně odchytávat a ošetřit vzniklé výjimky.

Další funkcionalitou AO je možnost vytváření vazeb mezi entitami, v rámci vlastních dat tak můžeme jednoduše vytvářet složitější struktury, stejně jako při přímé práci s databází.

Díky uvedeným vlastnostem a také přímé podpoře AO systémem JIRA je využívání AO vhodné i pro velká množství dat a je také variantou doporučovanou a preferovanou přímo výrobcem systému.

3.3.3 Nástroj Velocity

Nedílnou součástí zásuvných modulů bývá uživatelské rozhraní (user interface, UI). V JIRA se pro zobrazení obsahu používá open-source renderovací nástroj Velocity, který je projektem Apache Software Foundation [23], [29].

Renderování stránky pomocí Velocity Engine probíhá na straně serveru, na klienta se odesílá vygenerované HTML. Velocity má přístup ke všem *public* funkcím přiřazené vykreslovací třídy a díky tomu je možné zobrazovat dynamický obsah stránky – aktuální obsahy proměnných, výsledky funkcí, obsah přizpůsobený dle aktuálně přihlášeného uživatele atd.

Vstupem pro renderovací jádro je textový soubor formátu *vm*, který obsahuje napevno naprogramované části stránky a escape sekvence pro dynamicky měněný obsah. Tyto sekvence engine zpracuje a nahradí je požadovanými hodnotami daného kontextu.

Výstupem je webová stránka v HTML kódu, který může obsahovat další funkcionality a rozšíření běžně používané v tomto jazyce jako stylování pomocí CSS nebo interakce s uživatelem přes JavaScript.

Souběžně s Velocity je využívána komponenta vývojové platformy Atlassian Template Renderer. Díky tomu není třeba řešit vytvoření a udržování kontextu při komunikaci s klientem. Další komponenta – AUI – zase usnadňuje vytváření designu, umožňuje využití již existujících CSS stylů a JavaScript funkcí. Tvorba výstupních stránek tak zabere mnohem méně času a navíc stránky vzhledově ladí s ostatními částmi systému.

3.4 Jazykové mutace JIRA

Systém JIRA ve výchozím stavu používá jako jazyk angličtinu, na webových stránkách Atlassianu je pak k dispozici řada jazykových variant, včetně češtiny. Zobrazovací jazyk systému má jednu výchozí hodnotu, nastavenou systémovým administrátorem. Ten také může doinstalovat další jazyky a to stejným způsobem jako zásuvné moduly [25]. Každý uživatel si pak může nastavit vlastní jazykové preference a to výběrem z jazyků nainstalovaných správcem [20].

Systém překladů v závislosti na preferencích uživatele je možné využít i v zásuvných modulech. Umožňuje to systémová komponenta SAL (viz Obrázek 9), konkrétně třída `I18nResolver` [27]. Díky tomu je možné se všemi texty pracovat jako s proměnnými a jejich hodnotu načítat až ve chvíli, kdy se zobrazují. Která varianta hodnoty (tj. který jazyk) se načte, řeší opět transparentně tato třída, automaticky vybírá verzi podle nastavení aktuálně přihlášeného uživatele [20].

Překladový modul je možné použít přímo v kódu i v dalších komponentách – např. v šablonách Velocity. Taktéž je možné upravit stávající překlady, přidat další jazyky pro již existující moduly apod. Podrobné návody lze najít přímo na vývojářských webových stránkách Atlassianu [23].

4 Návrh realizace malých projektů

Jedním z cílů této práce bylo navržení takových úprav aktuálního procesu řízení a dodání projektů, aby bylo možné jej použít pro projekt realizovaný v rámci této práce a další projekty podobného typu v budoucnu. Popisu provedených přizpůsobení se blíže věnuje právě tato kapitola.

4.1 Řízení projektu

Projekt, který je obsahem této práce, je vzhledem k popisu uvedenému v kapitole 2.3 značně odlišný. Zaprvé se nejedná o standardní projekt dodávaný externímu zákazníkovi společnosti (třetí straně), ale o interní projekt v rámci společnosti, kdy odběratelem je přímo jedno z oddělení společnosti – ICT. Zadruhé rozsah implementace je znatelně menší než u běžných projektů ve společnosti.

Z informací uvedených výše vyplývá, že se z pohledu společnosti jedná o typický malý projekt. Běžně využívaný proces PDP je z tohoto důvodu zbytečně rozsáhlý a je vhodné jej upravit s ohledem na informace uvedené v kapitole 2.4 Řízení malých projektů. Postup pro řízení a realizaci malých projektů navržený autorkou práce popisují následující odstavce.

V rámci fáze **přípravy projektu** odpadají kroky přípravy nabídky a návrhu smlouvy, projekt je vyvíjen jako interní a žádná nabídka či smlouva se nepředkládá. Před-projektovou rešerši také není potřeba zpracovávat samostatně, lze ji nahradit schůzkou se zodpovědnou osobou (v tomto případě vedoucí IT oddělení) a prodiskutováním podkladů vytvořených v rámci přípravy. Na základě takové diskuze a posouzení výsledků pak může být rozhodnuto o realizaci projektu.

Výstupy naopak nezbytné i pro malý projekt jsou prvotní požadavky, včetně stanovení cílů. Podle nich následně vzniká návrh řešení, prvotní plány pracnosti a s tím související harmonogram. Firemní metodika uvádí použití logického rámce i produktového rozpadu, pro malé projekty však může postačovat pouze jeden z těchto výstupů.

Identifikace rizik také zůstává potřebným krokem řízení, vzhledem k nízké prioritě malých projektů však není tento krok kritický a výsledný dokument postačuje ve formě jednoduché tabulky rizik a jejich případných dopadů.

Kalkulace projektu je od ostatních kroků mírně odlišná, obecně by součástí projektu měla být. Ovšem projekt, který je obsahem této práce, je specifický ještě jedním prvkem – na jeho vývoji se podílí především studenti – stážisti. Náklady jsou proto pro společnost minimální ve srovnání s běžnými projekty, a pokud nedojde k překročení rozsahu stáže, tak v podstatě i fixní.

Cenová kalkulace proto není součástí tohoto projektu, přestože do obecné metodiky pro malé projekty by měla být zařazena.

Fáze **definování projektu** obsahuje nezbytné sestavení týmu a identifikaci zainteresovaných stran, což koresponduje s krokem 7 v seznamu činností řízení malého projektu. Vzhledem k velikosti týmu však není potřeba vytvářet komunikační matici a u interního projektu může komunikace zůstat zachována na neformální úrovni. Popis týmu nevyžaduje samostatný dokument, je součástí Identifikační listiny projektu.

Zainteresované strany jsou v podstatě omezeny jen na interní prostředí firmy a jejich identifikace i řízení je tak snadnější. Díky tomu lze projekt zahájit neformálně a bez speciální schůzky pro zákazníka.

Výstupy všech předchozích aktivit by měly být na závěr fáze shrnuty do Identifikační listiny projektu, navíc doplněné o akceptační kritéria. Tato může mít u interního projektu opět podobu wiki stránky, je ovšem velmi vhodné nechat si i tuto schválit zákazníkem.

Projektový kontejner koresponduje s požadavky standardního projektu, jedinou výjimkou je aplikace pro znalostní profily, která je pro stážisty nerelevantní. Analogicky rozpočet pro prémie nemá pro stážisty význam.

Rozdělení příprav na uvedené dvě fáze má své opodstatnění v případě, že jsou rozděleny schválením projektu zákazníkem a podepsáním smlouvy. Taková situace ale u interních projektů prakticky není, proto je vhodnější využít spíše rozdělení podle standardu PMBOK a uvedené činnosti zahrnout do jednoho logického celku – **Zahájení**.

Fáze **plánování projektu** je jednou ze stěžejních, mj. ji v různé formě uvádějí i všechny výše zmiňované standardy. Metodika společnosti zahrnuje velkou řadu dílčích plánů, pro malý však není potřeba zacházet tolik do detailů. Lze zde využít stejný postup jako při upravování metodiky PRINCE2 – nahrazení komplexních plánů seznamem kritérií, kontrolním seznamem apod. Celkový plán řízení projektu pak může být jeden větší dokument zahrnující i dílčí plány, bez rozdělení na řadu dokumentů s jednotlivými plány.

Samotné **řízení projektu** také může mít zjednodušenou podobu. Pro přehlednost je vhodné zachovat rozdělení na iterace, i když budou rozsahově menší. Pro realizaci je možné vyjít ze základního plánu (má-li vhodnou formu, např. Ganttův diagram) nebo využít jednodušší podobu formuláře zmíněnou v kapitole 2.4.2.

V souladu se standardem PMBOK je součástí řízení také řízení rizik a změn. Ty je nutné mít pod kontrolou u projektů všech velikostí, jinak je dosažení cíle téměř nereálné. U malých projektů si však můžeme vystačit s jednoduchými seznamy či formuláři, také projednání a schvalování je vzhledem k velikosti týmu a umístění zainteresovaných stran mnohem jednodušší. Finanční dopady změn a rizik jsou pro projekt, který je obsahem této práce, nerelevantní, viz důvody uvedené výše u kalkulace.

O stavu projektu informují zainteresované strany reporty. U malého projektu, který trvá jen krátce, je potřeba zvážit jejich četnost, jednotlivé kroky jsou totiž v čase více nahuštěny a příliš časté informování může být spíše na škodu věci. Na druhou stranu interní projekt umožňuje využití neformálního informování vedení, bez vytváření zvláštního dokumentu. Pro snadnější řízení a přehled je vhodné udržovat seznam případných již projednaných témat a dohodnutých výsledků.

Firemní metodika zahrnuje do řízení i předání a akceptaci, z logiky věci se však zdá vhodnější zařadit tyto kroky do fáze **ukončení**, stejně jako je to učiněno v PMBOK. Součástí závěrečné fáze bývá i školení, v případě projektu navázaného na tuto práci se jedná o školení administrátora systému JIRA. Archivace dat je zajištěna přímo systémy využívanými ve společnosti, není proto potřeba se jí speciálně věnovat.

Poslední kroky řízení – vyhodnocení projektu a ocenění pracovníků jsou nezbytné i u malých projektů. Dokonce mohou být ještě podstatnější než u velkých projektů, vezmeme-li v úvahu výše zmíněný fakt, že motivace pro práci na malých projektech je menší.

Navržený postup nesleduje precizně jeden z přístupů k řízení projektů, vychází primárně z procesu společnosti a upravuje jej s využitím poznatků z ostatních uvedených metodik řízení. Poněkud nejasné se může zdát napojení na standard IPMA. Ten je aplikovatelný obecně nezávisle na velikosti projetu a proces PDP společnosti jej v původní verzi také zohledňuje, proto je vhodné jej použít i u malých projektů. Forma standardu – kompetence – se však velmi špatně zahrnuje do popisu kroků, jeho vliv tak je patrný především přímo při realizaci – aplikací a rozvíjením manažerských dovedností.

4.2 Technická realizace projektu

Poslední fází zmiňovanou ve standardu PMBOK a neuvedenou v předchozí kapitole je samotná **realizace.** Ta z pohledu procesu společnosti zahrnuje oblasti analýzy, architektury a designu a především pak oblast vývoje a testování. Analogicky k části řízení je potřeba i tyto oblasti přizpůsobit potřebám malého interního projektu.

Některé aktivity korespondují s kroky uvedenými v kapitole 4.1, toto je dáno překrýváním jednotlivých oblastí znalostně i časově. Ovšem zatímco řízení projektu se věnovalo spíše otázce *co* musí být vytvořeno, realizační část se zaměřuje na konkrétnější postupy – tedy *jak* konkrétní výstup vytvořit.

Prvním krokem **analýzy** je zjištění požadavků, tento krok musí být zachován nezávisle na velikosti projektu. Navazující návrh systému je také nutnou položkou, je potřeba upřesnit požadavky, specifikovat systém i dokončit odhady pracnosti. Pro interní projekty může klesat důležitost slovníku pojmů, zákazník i dodavatel jsou spolupracovníky v jedné společnosti a

jednom oboru, riziko odlišného chápání pojmů je tak značně nižší než u externích zákazníků. Díky tomu lze slovník udržovat pouze v neformální podobě jako stránku na wiki společnosti.

Sledování a správa nových požadavků by také měla zůstat zachována, ovšem může mít neformálnější podobu – např. seznam na wiki stránce projektu s poznámkou, v jakém stádiu požadavek aktuálně je.

Jak vyplývá z předchozího textu, oblast analýzy zůstane obsahově prakticky zachována, sníží se pouze formálnost některých částí.

V souladu s procesem PDP oblast **architektury a designu** vychází z výstupů analýzy. Vzhledem k velikosti projektu je možné architekturu navrhnout přímo v podobě pro implementaci bez mezikroku prvotního návrhu. Vzhledem k tomu, že realizovaný projekt je obsahově nový pro společnost i pro autorku práce, je nutné věnovat čas ověření realizovatelnosti požadavků (například využitím prototypů) a zohlednit tento fakt při vytváření projektových plánů.

Rozdělení aplikace do komponent nemusí být u malých projektů potřeba, záleží však nejen na velikosti aplikace, nýbrž i na její logické rozdělitelnosti. I rozsahově malý program může být složen z částí, které jsou smysluplné i samostatně, a naopak, některý projekt nemá význam dělit, protože vzniklé části jsou samostatně nepoužitelné. S tímto faktem souvisí i identifikace znovupoužitelných komponent, která podléhá stejným omezením.

Dokumentace může u interních projektů mít neformálnější podobu, nelze ji však vynechat úplně. Jako vhodný prostředek mohou posloužit opět wiki stránky projektu.

Dílo dodávané v rámci malých projektů typicky zahrnuje pouze malý tým vývojářů či dokonce jediného vývojáře. Z pohledu **vývoje** v procesu společnosti se tento fakt však prakticky neprojeví, stále je potřeba seznámit tým s cíli projektu, nachystat potřebná prostředí či dodržovat úroveň kvality kódu.

Zjednodušení můžeme využít u uživatelské příručky a to díky vývoji pro interní potřeby společnosti. Obdobně jako v předchozích případech dokumentace může mít podobu pouhých stránek na wiki společnosti.

Komplikací v případě našeho konkrétního projektu je fakt, že vytvářené dílo je obsahově nové z pohledu společnosti i autorky, nelze proto tolik stavět na předchozích zkušenostech ani znovupoužitelných komponentách dostupných z jiných projektů.

Poslední oblastí realizace je **testování**, vzhledem k rozsahu malého projektu je možné testování provádět primárně v rámci vývoje a pouze finální produkt ověřovat přímo na testovacím prostředí. Rozsah celého produktu se totiž výrazně neliší od rozsahu výstupu jedné iterace velkých projektů.

Malý tým umožňuje využít k záznamu chyb jednoduchý dokument (tabulku), není potřeba vytvářet tikety. Důležité je zajistit verzování tohoto dokumentu a přístup k aktuální verzi pro

všechny členy týmu. U středního týmu by již mělo význam využití zavedeného způsobu – tiketovacího systému JIRA. Členové týmu by se nemuseli nic nového učit a navíc tento systém umožňuje zaznamenávání času stráveného na jednotlivých úkolech, je tedy ideální jako nástroj pro podporu řízení.

Automatizované testy se vzhledem k originalitě projektu také nepředpokládají, již existující testy použít nelze, jejich vytváření by zabralo velké množství zdrojů a potencionální další využitelnost je velmi nízká, návratnost takové časové investice by proto byla minimální.

Popis realizace uvedený v této kapitole reflektuje obecné potřeby malých projektů i případné konkrétní detaily projektu, na kterém se bude ověřovat jeho použitelnost. Popis je celkově jednodušší než původní proces řízení, vynechává aktivity, které u malých projektů nejsou stěžejní, a u řady dalších snižuje jejich detailnost.

Značné zjednodušení je také provedeno v oblasti dokumentace, malý interní projekt si může dovolit prakticky veškeré dokumenty uchovávat výhradně v elektronické podobě – jako stránky na wiki nebo uložené v DMS systému společnosti (pro dokumenty se specifickým formátem, pro které je textová forma wiki nedostatečná). Ty splňují potřeby verzování, zálohování i dostupnosti pro dané uživatele.

5 Projekt AutoAssign

Tato kapitola popisuje realizaci projektu AutoAssign. Jejím cílem byla vytvořit zásuvný modul pro systém JIRA a zároveň ověřit v praxi koncept řízení malých projektů navržený autorkou práce. Získané znalosti o sytému JIRA, teorie projektového řízení i navržený postupu řízení malých projektů ve společnosti jsou prakticky aplikovány v prostředí komerční společnosti.

Rozdělení do podkapitol odpovídá dělení projektu oblasti analýzy, architektury a designu, vývoje, testování a projektového řízení dále rozděleného podle časové fáze projektu. Poslední podkapitola shrnuje dosažené výsledky pohledu z pohledu této práce.

5.1 Analýza

Analýza hrála stěžejní roli především v úvodu projektu, kdy autorka řešila požadavky projektu, jeho cíle a způsob, jak jich dosáhnout. Zjišťování a zpracování požadavků bylo přímo součástí projektu, což sice není ve společnosti preferovaná varianta, ale vzhledem k rozsahu a implementaci projektu stážisty nebyl tento postup na škodu věci.

5.1.1 Prvotní požadavky

Prvotní požadavky na výsledný produkt byly stanoveny na schůzi s konzultantem společnosti, který v tomto případě vystupuje v roli zákazníka. Schůze se uskutečnila dne 26. 9. 2012 v sídle společnosti a přítomni byli konzultant společnosti a autorka této práce. Výstupem schůze je následující shrnutí požadavků:

Projekt AutoAssign

Cíl projektu: Zlepšit aktuální stav fungování helpdesku společnosti
Způsob dosažení: Automatizovat přiřazování tiketů na první úrovni

Definice projektu

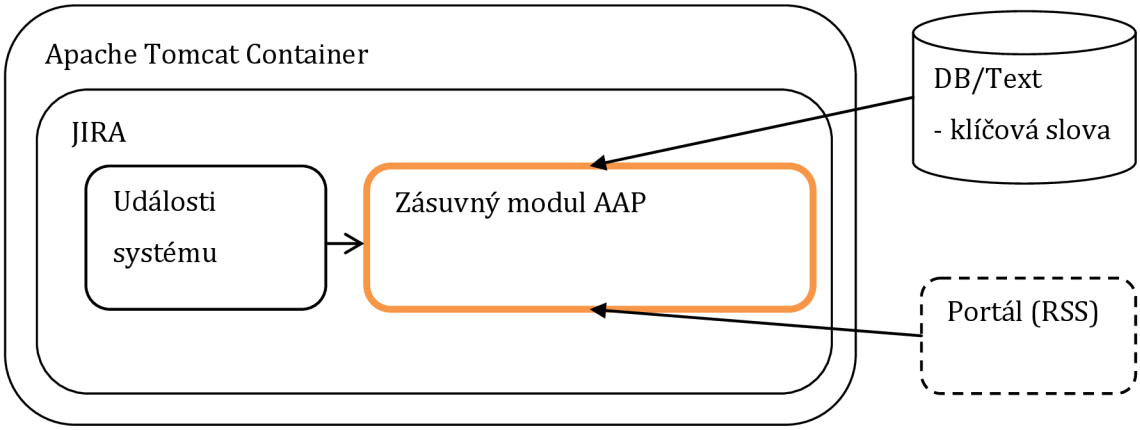
- Automatické řízení požadavků helpdesku
- Start projektu: 1. 10. 2012
- Konec projektu:
    - Zásuvný modul nasazen v produkčním prostředí
    - Předán k údržbě na support
    - Omezen trváním stáže autorky ve společnosti
- Zdroje
    - HW a SW prostředí poskytnuté společností
    - Lidé: konzultant společnosti, pracovníci ICT, autorka práce

Požadavky

- Nefunkční
  - Aktuální verze JIRA
  - Prerekvizity dle prostředí společnosti
  - Lokalizace česká a anglická
- Funkční
  - Automatické přiřazení tiketu řešiteli
    - Automatické štítky – Vyžadováno (Must Have, MH)
    - Automatické odmítnutí při nedostatku informací (MH)
    - Automatická zástupnost – nahrazení řešitele v případě jeho nedostupnosti (dovolené, služební cesta, školení, …) určeným zástupcem (Would nice to Have, WH)
    - Učení systému (WH)

Architektura

- Zásuvný modul navázaný na událost vytvoření tiketu v systému
- Nastavení uloženo v externím souboru nebo v databázi
- V případě implementace automatické zástupnosti se informace o dostupnosti zaměstnance přebírá z portálu společnosti (např. přes RSS)
- Modul součástí JIRA
- JIRA běží v prostředí Apache Tomcat
- Situaci ilustruje Obrázek 10 Základní návrh architektury



Obrázek 10 Základní návrh architektury

5.1.2 Návrh zásuvného modulu

Na základě požadavků zadaných zákazníkem, znalostí získaných při studiu a při přípravě teoretické části práce, byl proveden návrh zásuvného modulu. Návaznost na předchozí projekty ve společnosti a znovu-použití existujících komponent je v tomto případě irelevantní, autorka se proto snažila využívat alespoň existující funkce rozhraní systému JIRA.

5.1.2.1 Předpoklady

Zásuvný modul měl být vyvíjen pro systém JIRA 5.1 s využitím technologií Java SDK 1.6, Atlassian Plugin SDK 4.1 (dále jen kit) a dalších programů ve verzích obsažených v tomto kitu. V průběhu implementace však došlo ve společnosti k upgrade JIRA na verzi 5.2, čímž byl původní předpoklad porušen a projekt procházel změnovým řízením.

Zodpovědností zákazníka bylo zajistit přístup k vývojovému, testovacímu a produkčnímu prostředí po celou dobu práce na zásuvném modulu. Tento předpoklad byl zcela naplněn po celou dobu realizace, autorka práce měla přístup ke všem potřebným systémům a podkladům, včetně odborné literatury ve firemní knihovně.

Vytvořené dokumentace má formu wiki stránek v Confluence, zdrojové kódy se nahrávaly do systému GIT společnosti, v souladu s původními předpoklady projektu.

5.1.2.2 Přiřazování

Modul aktualizuje řešitele tiketu a štítky na základě popisu tiketu a nastavení modulu (rolí a klíčových slov, bližší popis viz dále). Pro spuštění přiřazení musí být splněny tyto požadavky:

- Tiket je aktuálně nepřiřazený (*unassigned*) nebo přiřazen neaktivnímu uživateli
- Tiket obsahuje pole popisu (konkrétně pole souhrn, prostředí, popis a štítky), mohou být i prázdná
- Vedoucí týmu (role *Team Lead*) projektu má oprávnění přiřazovat tikety (*assign issues*)
- Pro projekt existuje alespoň jedno nastavené klíčové slovo a alespoň jedna role
- Uživatelé nastavení v rolích jsou aktivní uživatelé a mají oprávnění přiřaditelný (*assignable*)
- Nastala jedna těchto událostí: vytvoření tiketu, aktualizace tiketu, přidání komentáře, změna komentáře, znovu-otevření tiketu

Při splnění podmínek modul vyhledá klíčová slova v popisu tiketu a přiřadí tiket uživateli s nejvyšším počtem nalezených slov. Jako autor změny je zaznamenán vedoucí týmu.

Pokud modul nenalezne žádné z klíčových slov, může tiket zanechat nepřiřazený nebo jej přiřadit výchozímu řešiteli (např. vedoucí týmu). Po diskuzi s konzultantem bylo stanoveno, že tiket zůstane nepřiřazen, aby však nezůstal přehlédnutý a neřešený, pošle modul automaticky zadavateli tiketu e-mail s žádostí o doplnění informací. Po doplnění se pokusí tiket znovu přiřadit. Tímto je naplněn další z požadavků a to automatické odmítnutí při nedostatku informací.

Uvedené přiřazování je velmi jednoduché, bylo by možné jej nahradit složitějším posuzováním textu v popisu tiketu, samo-učícím se systémem apod. Jednoduchá varianta byla zvolena na základě diskuze se zákazníkem, vyhovuje jeho požadavku na přehlednost a

dokonalou kontrolu nad tím, jak modul aktuálně tikety přiřazuje. Navíc se změna nastavení projeví vždy jasně daným způsobem, nenastane neočekávané chování a přiřazování.

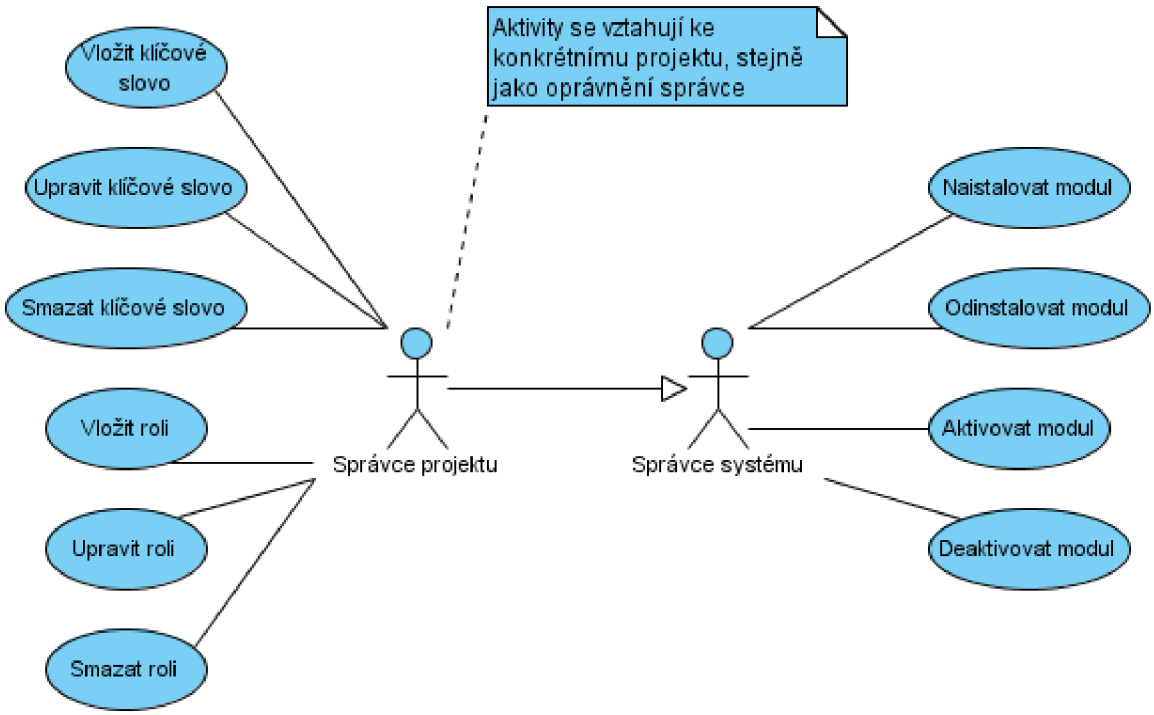
5.1.2.3 Práce s modulem

Zásuvný modul pracuje primárně automaticky na pozadí systému, bez interakce s uživatelem. Jediné místo, kde je zapojen uživatel, je nastavení modulu. Původně se předpokládalo nastavení pouze jako jednoduché vložení textu do pole formuláře webové stránky a možnost smazání dříve vložené hodnoty. V rámci druhého změnového řízení pak byla přidána také varianta editovat existující hodnoty.

Nastavení modulu bude specifické pro každý projekt v systému JIRA, hodnoty pro jeden projekt neovlivní nastavení pro jiný. Přístup k nastavení a možnost jej upravovat mají pouze uživatelé s oprávnění administrace projektu (role správce projektu, *Project Admin*) pro daný konkrétní projekt.

Na úrovni systémové administrace probíhá instalace, aktivace, deaktivace či úplné odinstalování modulu ze systému. Oprávnění k těmto akcím má pouze administrátor JIRA (role správce systému, *System Admin)* a jsou zajišťovány komponentou UPM.

Kompletní přehled možných uživatelských akcí přímo souvisejících s modulem ilustruje diagram případů užití, viz Obrázek 11. Běžný uživatel systému (člen týmu) zde není zahrnut, z jeho pohledu dojde po založení tiketu k automatické změně pole *řešitel* a to bez jeho aktivní interakce. Zadání tiketu samotné zase spadá pod funkcionality systému JIRA jako celku, nikoliv pod vyvíjený modul, proto taktéž není v diagramu uvedeno.



Obrázek 11 Případy užití modulu

5.1.2.4 Návaznost na další systémy

Původní základní návrh architektury (viz Obrázek 10) zahrnuje návaznost na dva externí systémy – databázi nebo textový soubor pro uložení nastavení a portál společnosti.

První z nich byl nahrazen transparentnějším řešením – uložení dat přímo v databázi systému JIRA a přístup k ní přes vrstvu Active Objects. Díky tomu je dosaženo požadované funkčnosti a zároveň je práce s databází korektně ošetřena vloženou přístupovou vrstvou. Snižuje se tak riziko chybného volání databáze a jejího případného poškození, navíc není potřeba řešit zvlášť instalace, přístupy a nastavení externí databáze.

Druhý uvedený systém – portál společnosti – měl být s modulem propojen kvůli požadavku automatické zástupnosti, využito mohlo být například rozhraní RSS. Po zvážení náročnosti implementace tohoto propojení a diskuzi autorky práce s konzultantem společnosti bylo rozhodnuto, že požadavek nebude v aktuální verzi modulu řešen. Vzhledem ke klasifikaci požadavku jako WH je toto rozhodnutí přípustné a případná realizace může být součástí budoucích rozšíření.

Kromě uvedených systémů se nepředpokládá žádná další explicitní spolupráce se systémy třetích stran.

5.1.3 Nastavení modulu

Kromě vytvoření modulu je součástí také jako iniciální nastavení. To zahrnuje vložení rolí a klíčových slov pro projektu Helpdesk, ostatní projekty v systému společnosti jsou v tuto chvíli nerelevantní.

Názvy rolí a jejich přiřazení konkrétním uživatelům vychází z aktuálně používaného seznamu rolí oddělení ICT, který je dostupný pro zaměstnance tohoto oddělení na stránkách wiki [16]. Další role mohou přidávat správci projektu podle potřeby.

Nastavení první části klíčových slov vychází také z wiki společnosti, role zde uvedené totiž jsou navázány na konkrétní produkty a služby. Jako klíčová slova se tak použily právě tyto názvy a jejich návaznost na role.

Druhá část klíčových slov využívá popisů již existujících tiketů Helpdesku. Autorka práce se několik měsíců věnovala přiřazování tiketů ručně, seznam klíčových slov proto doplnila na základě vlastních zkušeností podle často se opakujících výrazů, jednoznačných označení skutečností apod.

Vzniklé nastavení je pouze inicializační, předpokládá se jeho průběžné doplňování podle potřeb ICT oddělení. Pro udržení znalostí je vhodné aktuální popis nastavení ukládat také na wiki stránku projektu. Při pádu systému by sice nastavení mělo být obnoveno společně s celou databází, nutnost záloh se tím však neeliminuje.

5.2 Architektura a design

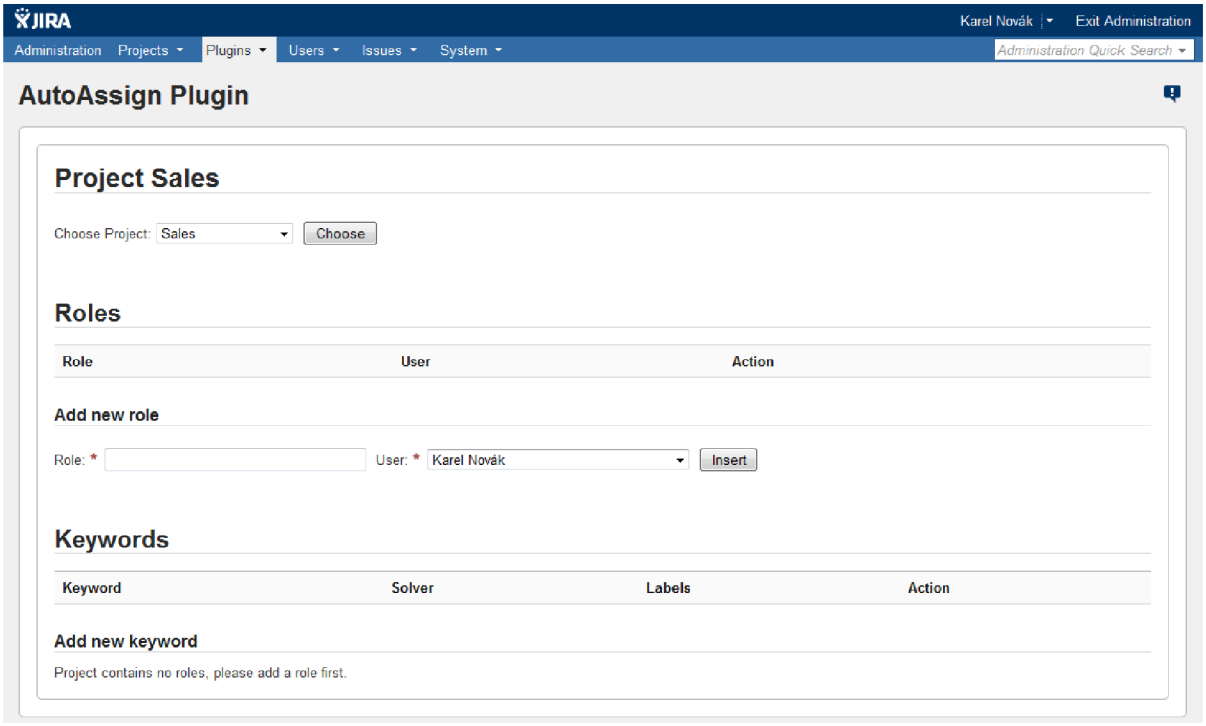
Tato kapitola se věnuje návrhu architektury a designu modulu. Stěžejním bodem obou návrhů je propojení se systémem. Architektura se zohlednit funkční možnosti systému a implementačního jazyka, design zase reflektuje jeho aktuální vzhled a rozložení.

5.2.1 Grafické uživatelské rozhraní

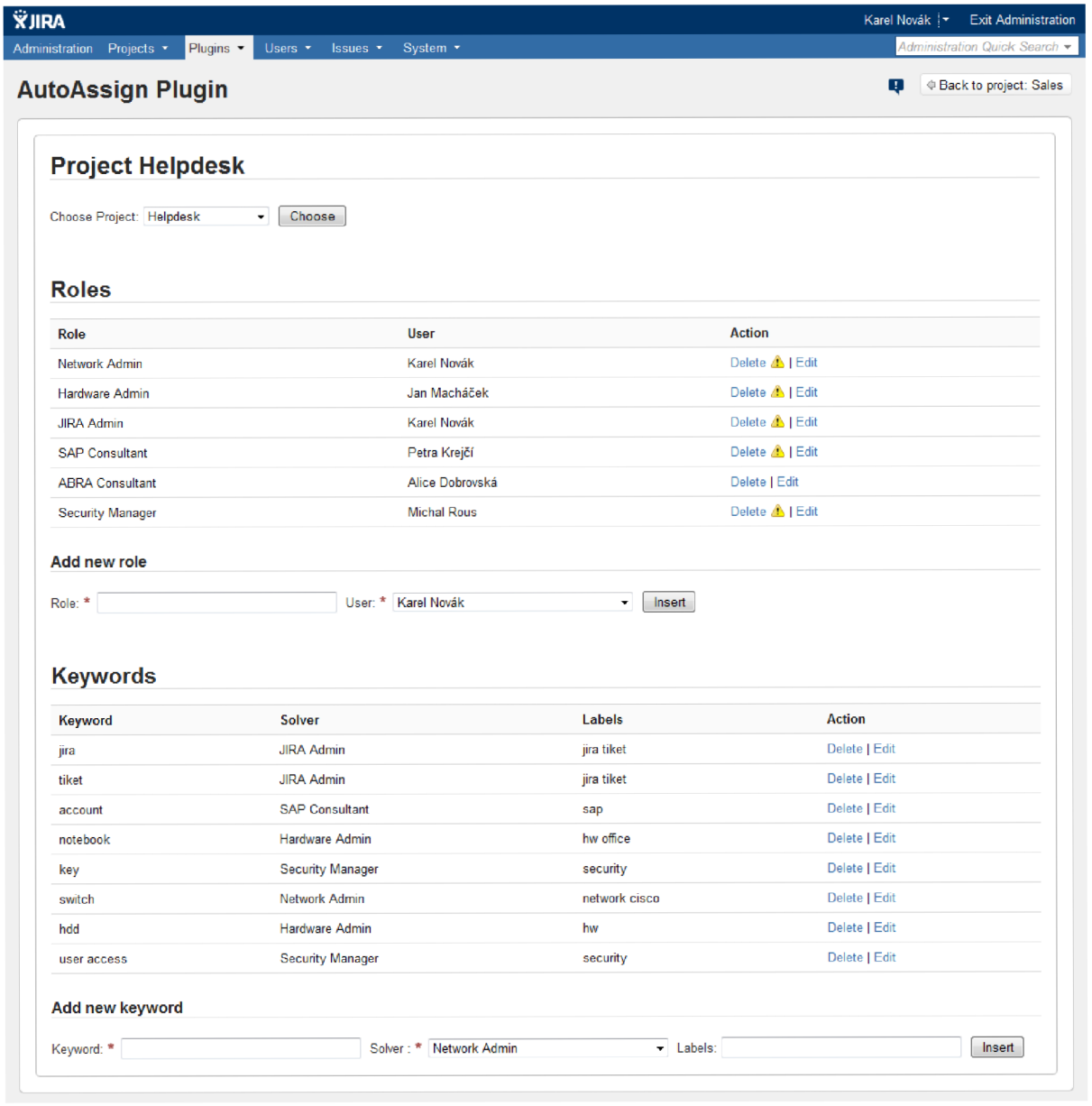
Uživatelské rozhraní (GUI) pro ovládání má velmi jednoduchou podobu, obsahuje seznamy uložených hodnot rolí a klíčových slov a formuláře pro vložení nových hodnot. Také umožňuje smazání již existujících položek a po zapracování změnového požadavku i jejich editaci. Pro tyto účely slouží jednoduché webové stránky s tabulkami a příslušnými formuláři.

Vzhledově je GUI sladěno s ostatními prvky systému, k tomuto účelu jsou využity existující kaskádové styly komponenty AUI. Stránka nastavení je zakomponována do existující struktury stránek a odkazů systému JIRA. Je přístupná pouze pro uživatele s dostatečnými oprávněními (*Project Admin, System Admin*) přes administraci jako součást hlavního menu Pluginy a také přes popis zásuvného modulu v seznamu všech modulů.

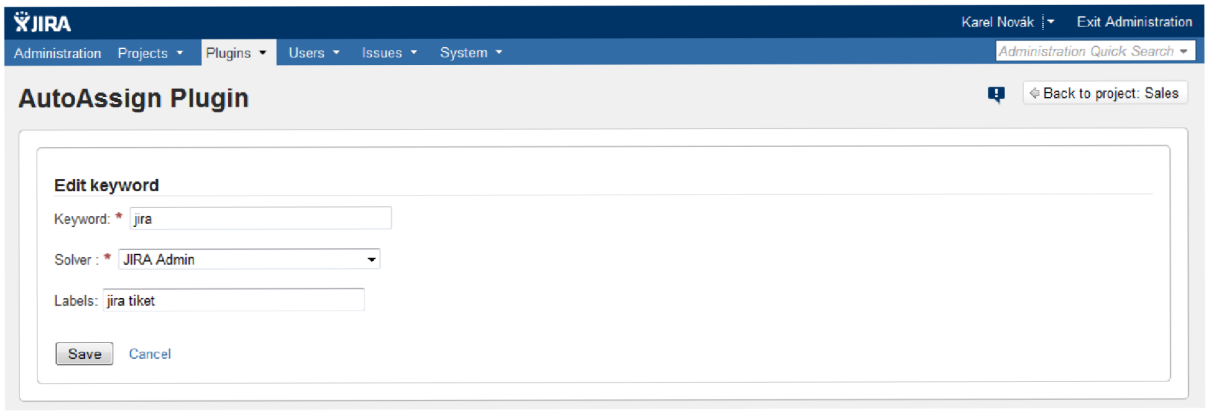
Výsledný design GUI ilustrují snímky obrazovek z testovacího prostředí, viz Obrázek 12 Výchozí prázdný formulář nastavení pro projekt, Obrázek 13 Formulář pro projekt s nastavením a Obrázek 14 Formulář pro editaci existujícího klíčového slova.



Obrázek 12 Výchozí prázdný formulář nastavení pro projekt



Obrázek 13 Formulář pro projekt s nastavením

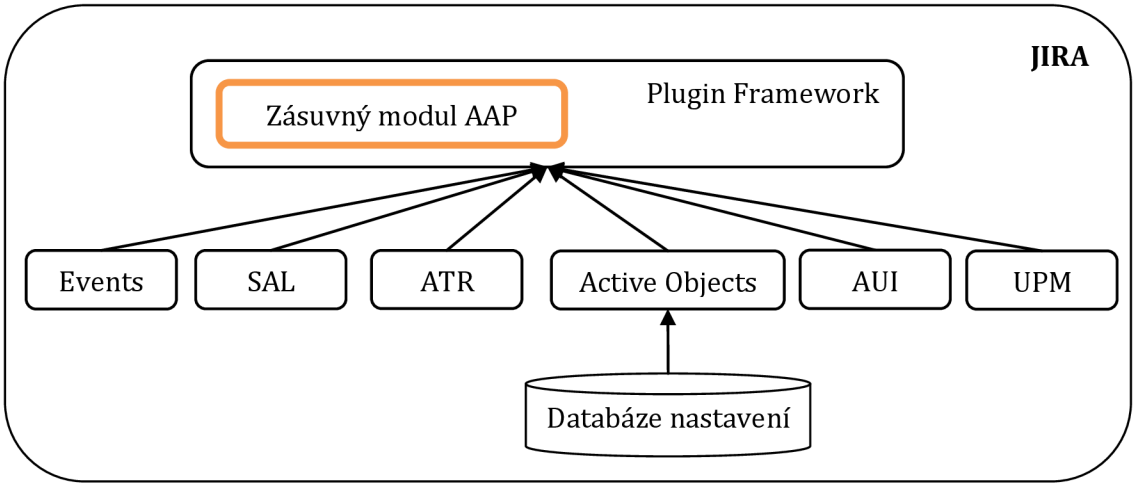


Obrázek 14 Formulář pro editaci existujícího klíčového slova

5.2.2 Architektura

Základní koncept architektury byl vytvořen již jako součást prvotních požadavků, ovšem v průběhu návrhu došlo k určitým změnám, navíc původní koncept není dostatečně detailní. Hlavní změna se dotýká propojení s externími systémy, databáze byla nahrazena vrstvou Active Objects a propojení s portálem nabude realizováno.

Navržený modul využívá také řadu rozhraní systému JIRA, ne jen systém událostí. Do návrhu tak byly doplněny vrstvy využívaných rozhraní. Navázání na Apache Tomcat je z pohledu modulu transparentní přes systém JIRA jako celek, modul s ním nijak explicitně nekomunikuje, z návrhu byl proto odstraněn jako nepodstatný. Výslednou provázanost modulu a systému zachycuje Obrázek 15, zkratky zde uvedené odpovídají prvkům systému JIRA uvedeným v kapitole 3.3.



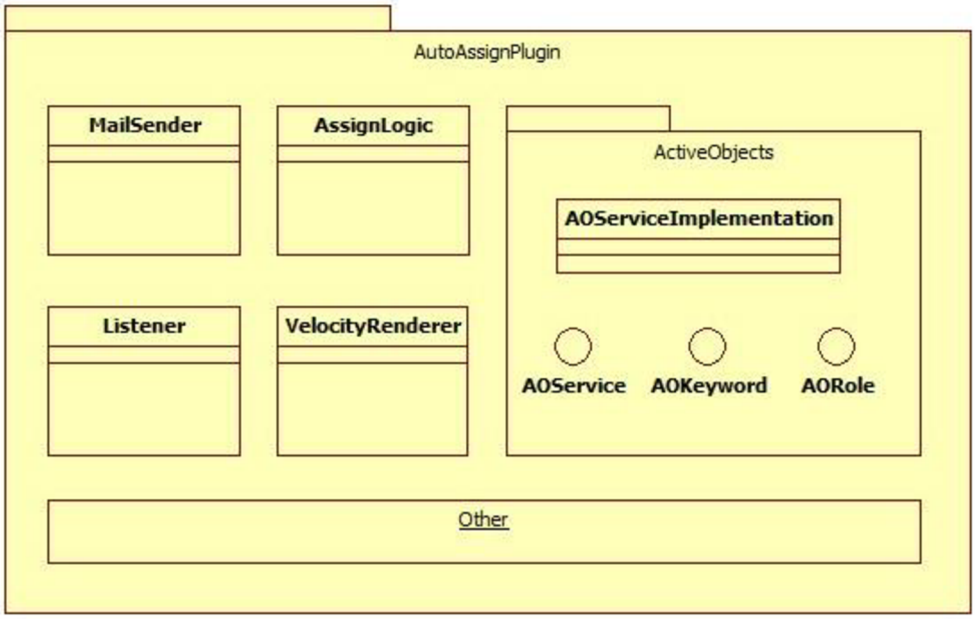
Obrázek 15 Architektura modulu v systému

5.2.3 Implementační třídy

Modul bude vytvořen v jazyce Java, v souladu s prostředím ve vývojovém kitu Atlassianu. Funkcionalita tak bude rozdělena do několika vzájemně komunikujících tříd, viz Obrázek 16.

Jednotlivé třídy mohou přes rozhraní Plugin Frameworku komunikovat s rozhraními dalších vrstev uvedených v celkovém přehledu architektury (Obrázek 15). Navázání na tato rozhraní je vytvářeno dle potřeb konkrétní třídy, vzájemně se neomezují a zároveň ne každá třída potřebuje navázání na tu či onu externí komponentu. Shrnutí do jednoho přístupového bodu z vnějšího pohledu celého systému naplňuje Plugin Framework a pohledu jednotlivých tříd by znamenala zbytečné komplikace, proto nemá význam takový bod vytvářet.

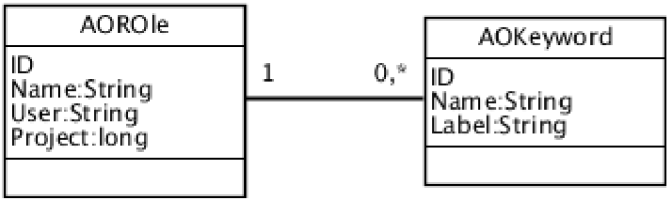
Pro možnost kontroly funkčnosti modulu jako celku musí být součástí každé třídy logování. Využit je systém *log4j,* výchozí pro logy celého systému. Každá třída tak zaznamenává úspěšné i neúspěšné akce a v případě potřeby může administrátor zpětně zjistit, co se v systému dělo.



Obrázek 16 Návrh Java tříd

5.2.3.1 Balíček ActiveObjects

Informace potřebné pro práci modulu jsou uloženy v databázi, pro přístup se používá komponenta Active Objects (AO). Ukládány budou pouze proměnné prostých datových typů (typ String a long) a vazby mezi entitami. Vrstva AO zajistí jednoznačné ID každé entity. Strukturu entit modulu v databázi zachycuje Obrázek 17.



Obrázek 17 Struktura databáze

Metody AO volané ostatními částmi popisuje rozhraní AOService, jeho implementace je potom obsahem třídy AOServiceImplementation.

Role má svůj název a přiřazeného uživatele, slouží především ke zpřehlednění nastavení, kdy se klíčová slova váží na roli a ne na jméno člověka, pod jednu roli může spadat 0-n klíčových slov. Rozhraní pro tuto entitu se jmenuje AORole.

Klíčové slovo obsahuje slovo k vyhledání v popisu tiketu a případně i štítek, který má být přidán, pokud je slovo nalezeno. Každé slovo je navázáno právě na jednu roli. Entita je popsána rozhraním AOKeywords.

5.2.3.2 Třída VelocityRenderer

Vykreslování GUI bude interaktivní, jeho obsluhu zajistí speciální třída. Využije se systém šablon Velocity, funkce volané vykreslovacím jádrem budou implementovány právě zde. Součástí implementace bude i ošetření přístupu ke stránce s nastavením.

5.2.3.3 Třída Listener

Hlavní třída modulu, reaguje na události v tiketu a při splnění podmínek zajistí přiřazení. Volá akce jiných tříd a zajišťuje také zápis požadovaných změn zpět do systému.

5.2.3.4 Třída AssignLogic

Třída obsahující výpočetní přiřazovací logiku. V tomto případě pouze prochází popis tiketu, vyhledává klíčová slova a podle výsledků určí řešitele a štítky tiketu.

5.2.3.5 Třída MailSender

Tato třída řeší zasílání e-mailových upozornění v případě, že jsou splněny podmínky pro přiřazení, ale v popisu není nalezeno žádné klíčové slovo. V takovém případě je automaticky zaslán zadavateli tiketu e-mail s žádostí o doplnění informací.

5.2.3.6 Other

Vzhledem k použitým technologiím je kromě samotné implementace potřeba ještě připojit šablony stránek a JavaScriptové soubory pro Velocity. Dále zde jsou zahrnuty soubory s překlady do jiných jazyků a XML soubory s popisem zásuvného modulu pro překlad a instalaci do systému.

5.3 Vývoj

Projekt v původním rozsahu byl naplánován do čtyř vývojových iterací, po změnovém řízení byla přidána ještě iterace pátá. Vzhledem k malému počtu aktivit uživatelů a jejich rozvržení a vzájemným návaznostem byl projekt do iterací rozdělen nikoliv podle případů užití, jak je zvykem ve společnosti, ale podle implementovaných Java tříd, viz Obrázek 16.

Jak bylo uvedeno v prvotních požadavcích, hlavním implementačním jazykem je Java. Doplňující soubory jsou potom podle potřeby i v jiných formátech, např. Velocity, XML, nebo JavaScript. Hlavním podkladem pro implementaci byla vývojářská dokumentace a návody Atlassianu (zdroje [20], [21] a [23]). Veškeré vytvořené soubory jsou k dispozici na přiloženém DVD, včetně návodu na jejich překlad a spuštění aplikace.

První iterace tvořila základ pro další vývoj, jejím cílem bylo vytvořit takovou kostru modulu, kterou bude možné nahrát do systému a která bude reagovat na jeho události. Z pohledu architektury se jednalo o kostru třídy *Listener.* Kromě třídy samotné vznikaly i konfigurační soubory nastavení celého modulu pro překlad a instalaci.

Protože kostra neobsahuje nastavení ani přiřazovací logiku, pro její testování byly vytvořeny zjednodušené verze – hledání slov a přiřazení konkrétnímu uživateli přímo definovaných ve zdrojovém kódu. V průběhu dalších iterací byly tyto části nahrazeny již funkčními implementacemi příslušných tříd.

Druhá iterace rozšiřovala vytvořenou kostru a přidávala do ní části interakce s uživatelem. Implementovány byly třídy zajišťující nastavení – balík *ActiveObjects* pro vkládání a mazání hodnot a *VelocityRenderer* pro vykreslování GUI. Součástí tedy byly i šablony stránek, soubory s jazykovými překlady a další pomocné soubory. V rámci implementace byla tato fáze nejrozsáhlejší.

Pro vyšší uživatelský komfort byl formulář pro práci s nastavením vytvořen s řadou drobných detailů:

- smazání hodnoty vyžaduje potvrzení v dialogovém okně, aby se eliminovalo smazání jedním chybným kliknutím,
- pole nastavení štítků umožňuje vložení více hodnot zároveň (oddělení mezerou),
- klíčová slova mohou být složena z více slov (slovní spojení),
- pole *role* a *klíčové slovo* nesmí být prázdná a toto je ověřováno,
- modul kontroluje existenci projektů a dovolí vkládat nastavení jen pro existující projekty,
- modul kontroluje existenci rolí a umožní vkládat klíčová slova, jen pokud existuje pro daný projekt nějaká role (viz Obrázek 12),
- při smazání role jsou smazána i provázaná klíčová slova, aby nedocházelo k nekonzistencím v databázi, uživatel je na toto upozorněn výstrahou u rolí, které mají navázána klíčová slova (viz Obrázek 13).

Poměrně malá byla **iterace třetí.** Jejím cílem bylo přidat samotnou funkčnost modulu – logiku pro přiřazování obsaženou ve třídě *AssignLogic.* Na základě schůze s konzultantem společnosti však bylo stanoveno, že samotná přiřazovací logika bude velmi jednoduchá a tím se zúžil i rozsah celé fáze v poměru k ostatním.

Pro přiřazovací logiku byly autorkou navrženy dvě varianty, obě vycházely z prohledávání popisu tiketu (pole souhrn, prostředí, popis a štítky). První varianta součtuje počet nalezených klíčových slov podle konkrétního uživatele, napříč jemu přiřazenými rolemi. Tiket potom přiřadí uživateli s nejvyšším počtem nalezených slov. Druhá verze naopak sčítá nalezená slova vzhledem ke konkrétní roli a tiket přiřadí uživateli v roli s nejvyšším počtem

nalezených slov. V rámci modulu byly implementovány obě varianty, aktuálně je používána první uvedená a druhá je neaktivní (v komentáři).

Původně poslední **čtvrtá iterace** přidala do modulu reakci na nevhodně zadaný tiket, výstup předchozí fáze na takový tiket v podstatě nereagoval. Cílem fáze tedy bylo implementovat třídu *MailSender,* která v případě nepřiřazení zasílá notifikace zadavateli tiketu. Kromě třídy samotné byly doplněny také soubory s jazykovými variantami e-mailu, aby implementace splňovala požadavek existence české a anglické verze.

Zasílání e-mailů stejně jako ostatní třídy v maximální možné míře využívá služby rozhraní JIRA. Nastavení adres uživatelů či serveru odchozích zpráv tak přebírá ze systému, čímž se modul zpřehledňuje a zároveň zachovává konzistenci s ostatními e-mailovými notifikacemi zasílanými při jiných událostech.

Jak již bylo zmíněno, **pátá iterace** byla přidána až po druhém změnovém řízení. Cílem iterace bylo zapracovat požadavek na editaci hodnot, jednalo se tak o úpravu dvou již existujících tříd:

- *VelocityRenderer* – přidání funkcí potřebných pro nový formulář editace a
- *Other* – přidání šablon a překladů pro nový formulář.

S novým formulářem souvisí také ukládání provedených změna tím pádem práce s třídou *ActiveObjects.* Zde však nebyly žádné změny potřeba, využívá se přímo funkcí rozhraní, které mimo jiné přímo poskytuje i funkce nastavení hodnot atributů a následného uložení změn do databáze.

Implementované řešení využívalo řadu nástrojů poskytovaných v rámci podpory vývojářů přímo tvůrcem systému JIRA – Atlassianem. Zároveň se autorka snažila v co nejvyšší míře využívat funkce poskytované rozhraním JIRA. Implementace se tak svým způsobe, zjednodušila, protože řadu problémů a potřeb šlo řešit využitím některé existující funkcionality. Navíc využití existujících funkcí zajišťuje konzistenci se systémem, změní-li se její vnitřní implementace, bude modul automaticky využívat vždy aktuální verzi.

Nevýhodou použití rozhraní byla nutnost se s ním seznámit. V průběhu vývoje bylo potřeba zjišťovat, jaké funkce jsou dostupné, za jakých podmínek, jak je lze navázat do vlastního kódu, co je naopak potřeba řešit samostatně apod. Ve výsledku tak je implementace jednodušší, ale její vytvoření nebylo o mnoho časově méně náročné.

5.4 Testování

Testování produktu probíhalo průběžně při implementaci a to na lokálním prostředí během vývoje. Vzhledem k velikosti projektu nebyly jednotlivé výsledky iterací nasazovány na testovací prostředí, ale tento postup byl aplikován až na výsledný produkt.

Testování na lokálním prostředí s sebou neslo řadu přínosů i několik citelných nevýhod. Plusem byla rychlá dostupnost a ověření implementované funkcionality, vývojář nemusel čekat na testera, ale prováděl základní testy sám. Vývojový kit Atlassianu umožňuje mimo jiné také upgrade verze zásuvných modulů za běhu díky komponentě FastDev, do lokálního testovacího prostředí tak mohly být velmi rychle nahrány změny a okamžitě otestovány.

Nevýhodou lokálních testů byl pomalý start celého testovacího kitu, v případě JIRA se totiž jedná se totiž o serverovou aplikaci a její spouštění na běžném vývojovém stroji (notebooku) zabere velmi dlouhou dobu. Ze stejného důvodu nejsou ani reakce systému při testech příliš svižné, každopádně jejich použitelnost je stále dostatečná.

Výsledný produkt byl nasazen na testovací serverové prostředí – kopii produkčního systému JIRA společnosti. V současné době probíhají jeho testy a během blízké doby by produkt měl být nasazen pro běžné užívání v ostrém provozu.

5.4.1 Nalezené chyby

V průběhu testování byla odhalena řada drobných chyb, vzhledem k provádění testů přímo vývojáři však tyto nedostatky byly ihned napravovány. Jednalo se například o chybné ověření práv k úpravám, zapomenutí uložení změn nastavení do databáze, apod. Všechny takové nedostatky však vyžadovaly jen drobnou rychlou úpravu kódu a z pohledu projektu nezpůsobily problém.

Mezi významné nalezené chyby patří problém s aktualizací při současném použití dalšího zásuvného modulu reagujícího na vytvoření tiketu. Podle logu systému bylo přiřazení řešitele modulem provedeno, ovšem ihned poté byl tiket znovu aktualizován jiným modulem a ten změnu řešitele nereflektoval (tj. přepsal jej zpět na „nepřiřazeno“). Tento problém se objevil při testech nové verze JIRA, nasazené ve společnosti během ledna.

Problém byl nakonec vyřešen, zvláštní chování bylo způsobeno rozdílem při práci s odkazem na tiket – získání citovatelné verze přes funkci API se chovalo nekorektně, zatímco pouhé přetypování původního odkazu již problém nezpůsobovalo. Kvůli odstraňování této chyby se však projekt zdržel a muselo dojít ke změnovému řízení.

5.5 Projektové řízení

Tato kapitola se věnuje aspektům řízení projektu AutoAssign během jeho realizace. Jsou zde uvedeny primárně ty kroky, které se nepřekrývají s jinými oblastmi, a tudíž nejsou uvedeny v předcházejících kapitolách. Rozdělení kapitoly reflektuje rozdělení projektu do časových fází.

5.5.1 Zahájení

Cílem projektu byla automatizace první úrovně Helpdesku společnosti. Důvodem pro takovou potřebu je především vyhovující aktuální stav. Helpdesk ve společnosti má dvě úrovně, na první přichází všechny tiketu a určený pracovník je potom rozesílá dále konkrétním řešitelům.

Problém tohoto systému je především potřeba pracovníka, který se věnuje uvedené činnosti (po dobu realizace projektu to byla autorka práce). V případě nedostupnosti pracovníka z důvodu např. krátkodobé nemoci, služební cesty apod. tikety zůstávají nepřeposílány a uživatelé tak čekají na jejich vyřízení až do návratu pracovníka. Uvedená situace je problémová v případě urgentních požadavků, například nefunkčnosti některého stěžejního systému.

Stěžejními kroky zahájení je identifikace prvotních požadavků a návrh systému, což bylo provedeno v rámci analýzy. Dalším krokem je identifikace zainteresovaných stran, v případě našeho projektu se jedná o tyto osoby či skupiny:

- konzultant společnosti (zákazník),
- autorka práce,
- vedoucí této diplomové práce,
- zaměstnanci ICT a
- uživatelé systému (ostatní zaměstnanci společnosti).

Kromě vedoucí práce jsou všechny osoby zevnitř společnosti, komunikace s nimi tak může být neformální a není většinou potřeba zvláštního plánování schůzek.

Všechny zainteresované strany mají primární zájem na úspěchu projektu. Jediný, kdo by mohl být zainteresován negativním způsobem, jsou ti uživatelé, kteří nezadávají do popisu tiketů korektní data a budou jim proto zasílány e-mailové notifikace s žádostmi o doplnění. Protože však takových tiketů je minimum a navíc i při aktuálním stavu jsou jejich zadavatelé vyzváni k doplnění, netvoří tato skupina příliš významný problém.

V každém projektu se vyskytují rizika, která je nutno také identifikovat a řídit. V případě malého interního projektu byl vytvořen seznam rizik a jejich kategorizací uvedený v tabulce Tabulka 1 Rizika projektu. Tento seznam je výchozí a předpokládá se jeho aktualizace podle potřeb a vývoje projektu.

Riziko	Dopad	Pravděpodobnost
Bojkot projektu některou zainteresovanou stranou	Velký	Nízká
Nedostatek financí	Nerelevantní	Nerelevantní
Nedostupnost lidských zdrojů	Velký	Nízká
Nedostupnost technického vybavení	Střední	Nízká
Nezkušenost týmu	Střední	Vysoká
Problémy s návazností na systém JIRA	Velký	Nízká
Předčasné ukončení	Velký	Nízká
Příliš krátké termíny	Střední	Střední
Špatná specifikace	Velký	Nízká
Špatný odhad pracnosti	Střední	Střední
Změna požadavků	Střední	Vysoká
Změna verze systému	Velký	Vysoká

Tabulka 1 Rizika projektu

Hodnoty uvedené v tabulce Tabulka 1 ve sloupcích „dopad“ a „pravděpodobnost“ dále specifikuje Tabulka 2. Protože rozpočet není pro tento projekt relevantním měřítkem, jsou procentuální hodnoty dopadu vztaženy k době trvání projektu, tj. o kolik by se projekt při vzniku rizika prodloužil. V případě pravděpodobnosti hodnoty procent zase vyjadřují předpokládanou šanci, že riziko nastane, na škále 0% (nevyskytne se) až 100% (jistota).

Dopad	Předpokládané prodloužení [%]
Velký	20-30
Střední	10-20
Nízký	5-10
Nerelevantní	1-5

Pravděpodobnost	Předpokládaný výskyt [%]
Vysoká	60-100
Střední	30-60
Nízká	10-30
Nerelevantní	0-10

Tabulka 2 Procentuální vyjádření dopadu a pravděpodobnosti rizika

5.5.2 Plánování

V rámci zahájení projektu byla provedena analýza požadavků a také návrh systému, tyto jsou blíže popsány v kapitole 5.1. Na jejich základě byly již během zahájení stanoveny odhady pracnosti a sestaven prvotní časový plán projektu. V rámci plánování byly tyto prvotní odhady dále rozšířeny, proto jsou uvedeny v této části.

Původně stanovená **pracnost** projektu byla omezena rozsahem stáže autorky na oddělení ICT společnosti, tj. cca čtvrtinu pracovního úvazku po dobu čtyř měsíců, což odpovídá rozsahu přibližně 180 hodin.

Technické zdroje potřebné pro projekt nebyly prakticky ničím omezeny, testovací prostředí JIRA běží na serveru neustále a potřebné vývojové prostředí je zajištěno na služebním

počítači, který je plně k dispozici výhradně autorce projektu. Lidské zdroje jsou omezeny především dostupností autorky projektu a dále pak časovými možnostmi konzultantů společnosti a zaměstnanců ICT.

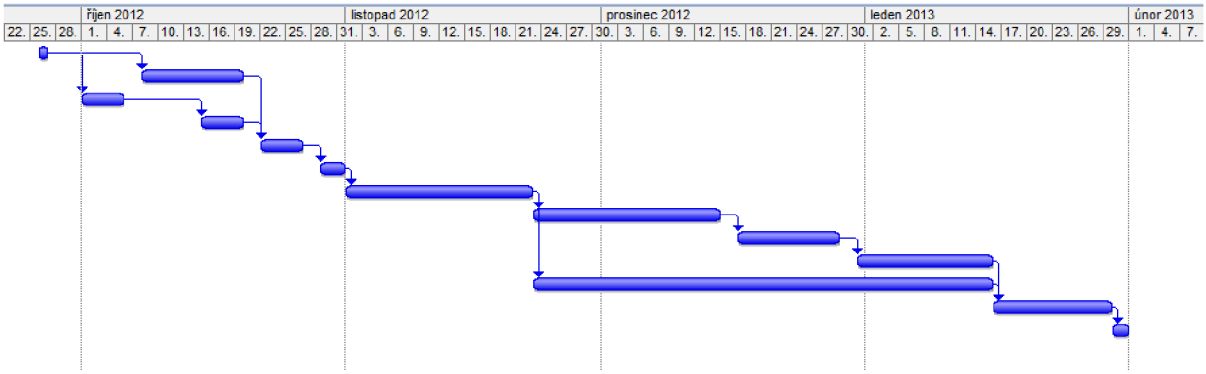
5.5.2.1 Harmonogram

Prvotní časový harmonogram byl stanoven na základě odhadů pracnosti a dostupnosti zdrojů. Jeho popis obsahuje Tabulka 3 Rozpis aktivit projektu, příslušný Ganttův diagram ilustruje Obrázek 18. Při čtení plánu je nutno brát v úvahu předpokládaný částečný úvazek na projektu, jeden den v plánu tak neodpovídá osmi hodinám práce, ale dvěma. V případě překryvu více aktivit plánu odpovídají uvedené dvě hodiny součtu těchto aktivit.

Velké množství zdrojů zabere učení se a prvotní odhad tak byl pouze velmi obecný. Návrhy, plány a další dokumenty proto byly doplňovány průběžně a původní rozsah se měnil, více viz kapitola 5.5.3. Vzhledem k interní povaze projektu však toto nebylo na závadu.

Název úkolu	Doba trvání	Zahájení	Dokončení	Předchůdci
Zahájení projektu	1 den	26.9. 12	26.9. 12	
Seznámení se systémem JIRA	10 dny	8.10. 12	19.10. 12	1
Analýza prvotních požadavků	5 dny	1.10. 12	5.10. 12	1
Návrh modulu	5 dny	15.10. 12	19.10. 12	3
Návrh architektury	5 dny	22.10. 12	26.10. 12	4;2
Návrh GUI	3 dny	29.10. 12	31.10. 12	5
Implementace - iterace I	16 dny	1.11. 12	22.11. 12	6
Implementace - iterace II	16 dny	23.11. 12	14.12. 12	7
Implementace - iterace III	10 dny	17.12. 12	28.12. 12	8
Implementace - iterace IV	12 dny	31.12. 12	15.1. 13	9
Testování	38 dny	23.11. 12	15.1. 13	7
Finální testování	10 dny	16.1. 13	29.1. 13	11;10
Nasazení do provozu	2 dny	30.1. 13	31.1. 13	12

Tabulka 3 Rozpis aktivit projektu



Obrázek 18 Ganttův diagram

5.5.3 Řízení realizace

V průběhu realizace projektu došlo k naplnění jednoho z identifikovaných rizik a také ke změně požadavků, oba případy vedly ke změnovému řízení a sestavení nových plánů.

5.5.3.1 Změna verze systému

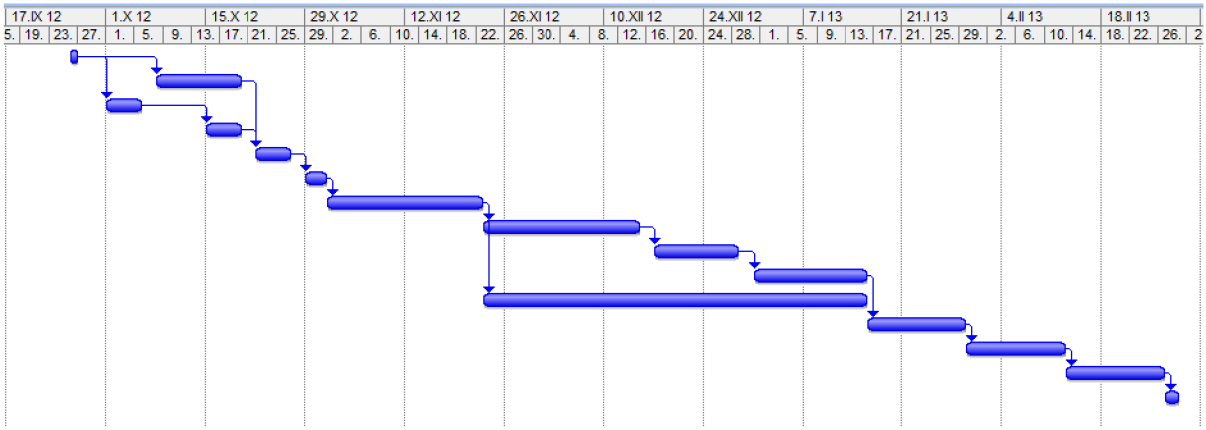
Modul měl být původně vyvíjen pro JIRA verzi 5.1, ovšem v průběhu měsíce ledna došlo ve společnosti k nasazení nové verze systému 5.2 – a modul muset být přizpůsoben této verzi. Nová verze neměla obsahovat žádné zcela zásadní změny, dal se tak předpokládat bezproblémový přechod.

V průběhu testování se však ukázalo, že modul nefunguje tak, jak má, a došlo tak k naplnění jednoho z identifikovaných rizik. Tikety zůstávaly nepřiřazeny i při splnění podmínek (více viz kapitola 5.4.1). Oprava chyby zabrala velké množství času, bylo proto nutné zahájit změnové řízení, protože původní plán projektu nemohl být dodržen.

Na schůzce se zákazníkem bylo domluveno řešení nastalé situace posunem konce projektu o jeden měsíc vůči původnímu plánu. Lidské zdroje vyžadované delším trváním nebyly problém díky tomu, že se autorka mohla projektu dále věnovat v rámci tvorby této práce a to i nad rámec rozsahu původní stáže. Upravený plán popisuje Tabulka 4 Aktivity projektu po první změně a Obrázek 19 Ganttův diagram po první změně.

Název úkolu	Doba trvání	Zahájení	Dokončení	Předchůdci
Zahájení projektu	1 den	26.9. 12	26.9. 12	
Seznámení se systémem JIRA	10 dny	8.10. 12	19.10. 12	1
Analýza prvotních požadavků	5 dny	1.10. 12	5.10. 12	1
Návrh modulu	5 dny	15.10. 12	19.10. 12	3
Návrh architektury	5 dny	22.10. 12	26.10. 12	4;2
Návrh GUI	3 dny	29.10. 12	31.10. 12	5
Implementace - iterace I	16 dny	1.11. 12	22.11. 12	6
Implementace - iterace II	16 dny	23.11. 12	14.12. 12	7
Implementace - iterace III	10 dny	17.12. 12	28.12. 12	8
Implementace - iterace IV	12 dny	31.12. 12	15.1. 13	9
Testování	38 dny	23.11. 12	15.1. 13	7
Testování nové verze JIRA	10 dny	16.1. 13	29.1. 13	10
Oprava chyby	10 dny	30.1. 13	12.2. 13	12
Finální testování	10 dny	13.2. 13	26.2. 13	13
Nasazení do provozu	2 dny	27.2. 13	28.2. 13	14

Tabulka 4 Aktivity projektu po první změně



Obrázek 19 Ganttův diagram po první změně

5.5.3.2 Změnový požadavek – editace nastavení

Další změnové řízení bylo způsobeno přidáním nového funkčního požadavku. Kromě vkládání a mazání hodnot měla být umožněna také jejich editace, původní rozsah se totiž při testování ukázal jako neefektivní pro práci.

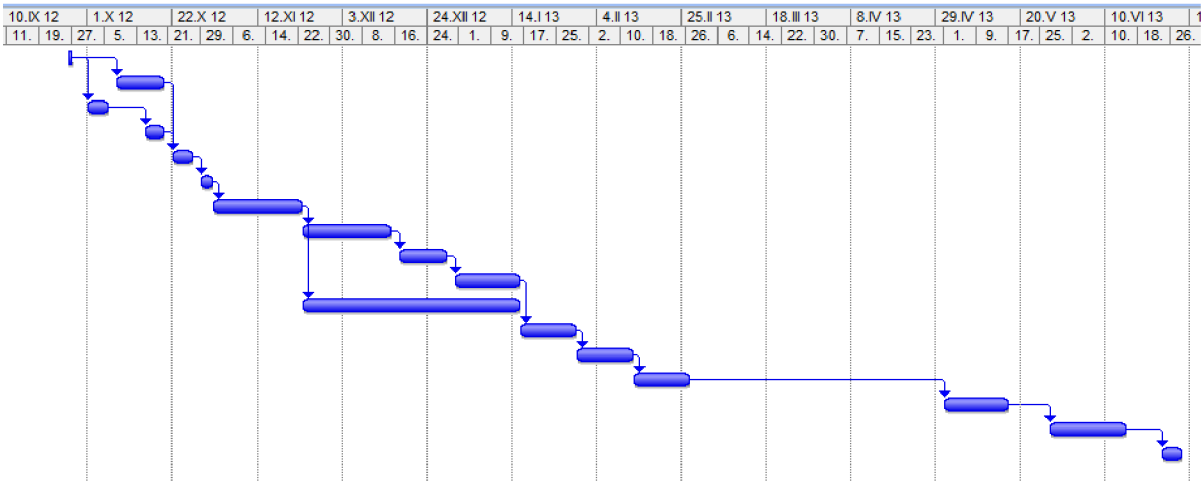
Příkladem situace, kdy původní verze není vyhovující, může být odchod zaměstnance z firmy a související změna obsazení role. V takovém případě totiž zůstává role a také k ní příslušející klíčová slova nezměněna, ovšem řešitel je jiný. Stará verze návrhu umožňovala pouze smazat roli a znovu ji vytvořit s novým uživatelem, což ale znamenalo také znovu-vytvoření všech souvisejících klíčových slov, což je velmi neefektivní a časově náročné.

Změnový požadavek byl schválen k implementaci, bez něj by totiž byl modul v reálu nepoužitelný. Jako řešení změny rozsahu bylo opět použito časového posunu projektu, tentokrát však nebyla situace tak snadná jako v předchozím případě. Autorka práce byla vytížena jinými povinnosti, projekt tak musel být po dobu několika týdnů odložen a poté znovu aktivován. Do aktuálního plánu tak kromě posunu byla přidána také další fáze implementace, viz Tabulka 5 a Obrázek 20. Dodatečná implementace zabere více dní než předchozí fáze, zaprvé autorka může práci věnovat méně času denně (jeden den v plánu tak odpovídá cca hodině práce) a zadruhé je po přerušení nutno počítat s osvěžením znalostí.

Z tohoto důvodu v době dokončování této práce projekt stále běží, poslední fáze testování a nasazení do ostrého provozu mají být dokončeny během měsíce června.

Název úkolu	Doba trvání	Zahájení	Dokončení	Předchůdci
Zahájení projektu	1 den	26.9. 12	26.9. 12	
Seznámení se systémem JIRA	10 dny	8.10. 12	19.10. 12	1
Analýza prvotních požadavků	5 dny	1.10. 12	5.10. 12	1
Návrh modulu	5 dny	15.10. 12	19.10. 12	3
Návrh architektury	5 dny	22.10. 12	26.10. 12	4;2
Návrh GUI	3 dny	29.10. 12	31.10. 12	5
Implementace - iterace I	16 dny	1.11. 12	22.11. 12	6
Implementace - iterace II	16 dny	23.11. 12	14.12. 12	7
Implementace - iterace III	10 dny	17.12. 12	28.12. 12	8
Implementace - iterace IV	12 dny	31.12. 12	15.1. 13	9
Testování	38 dny	23.11. 12	15.1. 13	7
Testování nové verze JIRA	10 dny	16.1. 13	29.1. 13	10
Oprava chyby	10 dny	30.1. 13	12.2. 13	12
Testování oprav	10 dny	13.2. 13	26.2. 13	13
Implementace - iterace V	12 dny	1.5. 13	16.5. 13	14
Finální testování	15 dny	27.5. 13	14.6. 13	15
Nasazení do provozu	5 dny	24.6. 13	28.6. 13	16

Tabulka 5 Aktivity projektu po druhé změně



Obrázek 20 Ganttův diagram po druhé změně

5.5.4 Ukončení

Vzhledem k aktuálně probíhajícímu testování zatím nebyl projekt ukončen, v tuto chvíli však aktuální stav testování naznačuje tomu, že se aktuální podoba implementace nasadí do provozu. Celkově by tedy projekt měl být zakončen jako úspěšný a také jako úspěšně řízený.

Nejkomplikovanější částí implementace projektu bylo jeho navázání na existující systém, tato část vyžadovala rozsáhlé studium a omezovala prostředky použitelné pro realizaci. Na druhou stranu bylo možné využít pokročilé vývojové prostředí a řadu funkcionalit rozhraní systému JIRA.

Kromě naplnění stanovených cílů a účelu orientovaného na společnost měl projekt také řadu přínosů pro autorku práce. Hlavními z nich bylo získání praktických zkušeností v oblasti projektového řízení a možnost vyzkoušení aplikace vlastního návrhu řešení.

5.6 Dosažené výsledky

Cíle sledované samotným projektem a jejich naplnění byly diskutovány v předchozí kapitole, ovšem tato práce jako celek překračuje rozsah projektu a zahrnuje i další aspekty a cíle, především v souvislosti s návrhem nového postupu řízení pro malé projekty. Z tohoto důvodu jsou dosažené výsledky práce diskutovány v této kapitole, odděleně od výsledků projektu.

Z pohledu návrhu řízení malých projektů se v praxi prověřilo, že je možné projekt podle tohoto postupu řídit. Zjednodušené verze dokumentace se prokázaly jako dostačující, stejně tak vynechané kroky při řízení projektu nechyběly, naopak vytvořené výstupy byly užitečné.

Na základě získaných zkušeností by však autorka u dalších projektů věnovala více času zpracování požadavků, v tomto případě mohla byt chybějící funkcionalita editace nastavení identifikována a zapracována mnohem dříve než u závěrečného testování. V případě projektu AutoAssign nebyly důsledky pozdní změny fatální, časové prodloužení projektu však rozhodně nebylo zanedbatelné a v jiných případech by mohlo vést k neúspěšnosti celého projektu.

Důležitým poznatkem je nutnost řízení rizik, při realizaci se několikrát ověřilo, že rizika a změny ohrožují i velmi malé projekty a mohou mít fatální důsledky. V případě našeho projektu naštěstí dopad naplnění rizik byl eliminován velkou časovou rezervou. U jiných podobných projektů tomu tak však nemusí být a velmi pravděpodobně také nebude, autorka proto apeluje na nepodceňování rizikových faktorů.

Společně s navrženým postupem řízení se autorka snažila aplikovat kompetence uváděné standardem IPMA, zvláště pak ty, související s jednáním s lidmi. V rámci možností sebe-posouzení považuje využití kompetencí za úspěšné, ovšem stále s prostorem k dalšímu zlepšování.

Celkově lze řízení projektu podle návrhu označit za úspěšné a použitelné pro další projekty podobného typu v budoucnu.

6 Závěr

V rámci studia podkladů k práci jsem se seznámila s řadou metodik a standardů projektového řízení i definicí procesů jeho praktické aplikace v konkrétní komerční společnosti. Také jsem detailně nastudovala informace o tiketovacím systému JIRA, a to z uživatelského, administrátorského i vývojářského pohledu.

Znalosti získané během fáze studia jsem reálně aplikovala při návrhu řízení malých interních projektů pro společnost. Tento návrh zohledňuje ověřené postupy řízení ve společnosti, obecné standardy i zvláštnosti malých projektů.

Navržený koncept jsem následně ověřila praktickým použitím na projektu AutoAssign. Cílem projektu bylo automatizovat práci s tikety ve společnosti, konkrétně první vrstvu helpdeskové podpory. Tento projekt potvrdil aplikovatelnost vytvořeného návrhu řízení a také ověřil úroveň získaných znalostí o systému JIRA. V době dokončování práce projekt stále běžel, ovšem vše nasvědčuje tomu, že bude úspěšně dosaženo jeho cílů, tj. výstup projektu bude nasazen do ostrého provozu ve společnosti a zajistí okamžité zpracování tiketů příchozích na hepldesk, nezávisle na dostupnosti pracovníka první úrovně podpory.

Implementovaný systém je možné dále rozvíjet, za vhodné rozšíření lze navrhnout sofistikovanější algoritmus určení řešitele tiketu, optimalizaci formulářových polí nastavení modulu, rozšíření možností nastavení o přizpůsobení obsahu zasílaných notifikačních e-mailů nebo vylepšení vzhledu stránky s nastavením.

Navržený postup řízení malých projektů je samozřejmě také možné dále vylepšovat, ideálně pomocí praktických zkušeností získaných jeho dalšími aplikacemi v reálném prostředí.

V rámci celé práce se povedlo naplnit stanovené cíle, byl vytvořen popis řízení vhodný pro malé interní projekty a jeho ověření v praxi bylo také úspěšné. Primárním přínosem práce pro společnost je prakticky použitelný modul a aplikovatelný postup řízení. Přínosem pro autorku je především získání cenných zkušeností ze všech oblastí vývoje software, především pak z řízení projektů.

Bibliografie

1. PITAŠ, J. a KOLEKTIV. *Národní standard kompetencí projektového řízení verze 3.2*. Vydání třetí - doplněné a aktualizované. Brno: Společnost pro projektové řízení, o.s. 2012. ISBN 978-80-260-2325-8.
2. KELÍŠEK, A. a R. ONDREJKA. Teoretické východiská k plánovaniu projektov. In: LACKO, B. *Projektový management 2011*. Brno: Akademické nakladatelství CERM, s.r.o. 2011, s. 3-16. ISBN 978-80-7204-770-3.
3. DOLEŽAL, J. et al. *Projektový management podle IPMA*. Druhé vydání. Praha: Grada Publishing, a.s. 2012, 528 s.. ISBN 978-80-247-4275-5.
4. *A guide to the project management body of knowledge: (PMBOK guide)*. 4th ed. Newtown Square: Project Management Institute, Inc. 2008. ISBN 978-1-933890-51-7.
5. BENTLEY, C. *PRINCE2 pro řízení malých podniků*. Překlad Michal PŮDA. Praha: EuroAnalysis s.r.o. 2010. ISBN 978-80-254-7236-1. Kniha je v souladu s revizí PRINCE2: 2009.
6. *A guide to the project management body of knowledge: (PMBOK guide)*. Fifth edition. Newtown Square: Project Management Institute, Inc. 2013. ISBN 978-1-935589-67-9.
7. LACKO, B. Aplikace metody RIPRAN v softwarovém inženýrství. In: *Tvorba softwaru 2001: celostátní konference : [29.5.-31.5.2001, Ostrava : sborník přednášek]*. Ostrava: Tanger, 2001, s. 97-103. ISBN 80-85988-59-3.
8. SPOLEČNOST PRO PROJEKTOVÉ ŘÍZENÍ. *Certifikační orgán Společnosti pro projektové řízení, o.s.* [online]. 2013 [cit. 2012-02-12]. Dostupné z: http://www.ipma.cz/
9. ŠVIRÁKOVÁ, E. a J. DOLEŽAL. *Řízení projektů I*. Zlín: Univerzita Tomáše Bati, 2010. ISBN 978-80-7318-990-7.
10. PITAŠ, J. et al. Příloha č.3 Výkladový slovník pojmů: Verze 3.2. In: *Národní standard kompetencí projektového řízení verze 3.2* [online]. Vydání třetí - doplněné a aktualizované. Brno: Společnost pro projektové řízení o.s. 2012 [cit. 2013-02-10]. ISBN 978-80-260-2325-8. Dostupné z: http://www.ipma.cz/web/files/IPMA-CzNCB-slovnik-pojmu-v3.2.pdf
11. ČESKÁ KOMORA PMI. Certifikace. *Stránky České komory PMI* [online]. 2013, verze 08 Březen 2013 [cit. 2013-03-12]. Dostupné z: http://www.pmi.cz/index.php/cs/professional-development/certification.html
12. SCHWALBE, K. *Řízení projektů v IT: Kompletní průvodce*. Vydání první. Brno: Computer Press, 2011. ISBN 978-80-251-2882-4.

13. APM GROUP LTD. *Official PRINCE2® Website* [online]. 2013 [cit. 2013-03-15]. Dostupné z: http://www.prince-officialsite.com/
14. *Managing successful projects with PRINCE2*. Fifth edition. The Stationery Office, 2009. ISBN 978-0-11-331059-3.
15. MAULE, P. Porovnání PRINCE2 a PMBOK. *Systémová integrace* [online]. Praha: Česká společnost pro systémovou integraci, 2004, č. 4, s. 84-99 [cit. 2013-03-10]. ISSN 1804-2716. Dostupné z: www.cssi.cz/cssi/system/files/all/SI_04_4_maule.pdf
16. SPOLEČNOST. *Wiki společnosti* [online]. [cit. 2013-Leden-09]. Dostupné z: Veřejně nedostupné.
17. *MoSCoW Prioritisation* [online]. [cit. 2013-Leden-09]. Dostupné z: http://www.coleyconsulting.co.uk/moscow.htm
18. ROSENAU, M. D. *Řízení projektů: příprava a plánování, zahájení, výběr lidí a jejich řízení, kontrola a změny, ...*. Vyd. 2. Brno: Computer Press, 2003. ISBN 80-722-6218-1.
19. KANTOROVÁ, K. *Management a podnikání II.* [online]. Ostrava: Ostravská univerzita, 2007 [cit. 2013-03-10]. Dostupné z: http://projekty.osu.cz/pvsos/doc/management.pdf
20. *Documentation for JIRA 5.1* [online]. 2012 [cit. 2013-Leden-09]. Dostupné z: https://confluence.atlassian.com/display/JIRA051/JIRA+Documentation
21. *Atlassian JIRA 5.1 API* [online]. 10. Červenec. 2012 [cit. 2013-Leden-09]. Dostupné z: http://docs.atlassian.com/jira/5.1/
22. Atlassian Marketplace. *User picker from project role - Issue alternative Assignee* [online]. 2009 [cit. 2013-Leden-09]. Dostupné z: https://marketplace.atlassian.com/plugins/com.iamhuy.jira.plugin.issue-alternative-assignee
23. *Atlassian Developers Portal* [online]. [cit. 2013-Leden-09]. Dostupné z: https://developer.atlassian.com/
24. ATLASSIAN. *Atlassian Marketplace* [online]. 2013 [cit. 2013-02-18]. Dostupné z: https://marketplace.atlassian.com/
25. DOAR, M. B. *Practical JIRA administration*. Sebastopol: O'Reilly Media, 2011. ISBN 978-144-9305-413.
26. Practical JIRA development. *Storing data in JIRA properties tables* [online]. 2012 [cit. 2013-Leden-09]. Dostupné z: http://jiradev.blogspot.cz/2010/09/storing-data-in-jira-properties-tables.html
27. DOAR, M. B. *Practical JIRA plugins*. First Edition. Sebastopol: O'Reilly Media, 2011. ISBN 978-1-449-30827-8.

28. *Active Objects* [online]. 2012 [cit. 2013-Leden-09]. Dostupné z: https://developer.atlassian.com/display/AO/Active+Objects

29. *The Apache Velocity Project* [online]. 29. Listopad. 2010 [cit. 2013-Leden-09]. Dostupné z: http://velocity.apache.org/

Příloha A: Project Delivery Process

Část práce vychází z popisu procesu Project Delivery Process společnosti, pro kterou byla práce implementována. Vzhledem k její veřejné nedostupnosti je součástí práce jako příloha, kvůli rozsahu je umístěno pouze na přiloženém DVD.

Příloha B: Seznam obrázků

Příloha C: Seznam tabulek

Příloha D: Obsah přiloženého DVD

Součástí práce jsou také další přílohy, obsažené na přiloženém DVD. Struktura DVD je následující:

- jira
    - src – složka se zdrojovými kódy aplikace
    - navody
        - navod.docx – návod na překlad a spuštění
        - navod.pdf – PDF verze návodu
        - prirucka.docx – uživatelský návod k nastavení modulu a jeho použití
        - prirucka.pdf – PDF verze uživatelské příručky
- technicka_zprava
    - technicka_zprava.docx – editovatelná verze této zprávy
    - technicka_zprava.pdf – technická zpráva ve formátu PDF
- dokumentace
    - Příloha A: Project Delivery Process