Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 812 19 Bratislava

ANALÝZA VÝSLEDKOV VÝSKUMU

Dokumentácia k riadeniu projektu

Tím 10 ResearchRank
Vedúci projektu: Ing. Nadežda Andrejčíková, PhD.
Členovia tímu: Bc. Michael Gloger
Bc. Rastislav Kostrab
Bc. Šimon Kompas
Bc. Tomáš Jánošík
Bc. Daniel KĺČ
Bc. Stanislav Kubica
Bc. Michal Walder
Školský rok 2013 / 2014

1 Úvod

Táto dokumentácia bola vytvorená za účelom sumarizácie všetkých dokumentov súvisiacich s riadením projektu v našom tíme na predmete Tímový projekt. Všetky dokumenty boli tvorené priebežne a sumarizujú čas a obsah vynaloženej práce na projekte.

1.1 Prehľad dokumentu

1. Ponuka
2. Úlohy členov
3. Metodiky
4. Zápisnice zo stretnutí (príloha A)

V prvej časti dokumentu sú predstavení členovia nášho tímu a sú tu uvedené ponuky pre tri vybrané témy, usporiadané podľa priority. V druhej časti sú uvedené manažérske role členov tímu a zodpovednosti každého z nich. Tretia časť obsahuje metodiky - pravidlá vykonávania jednotlivých procesov v rámci kontextu tímového projektu. Medzi prílohy sú zaradené zápisnice z tímových stretnutí.

1.2 Autori jednotlivých častí dokumentácie

Kapitola	**Autor**
1 Úvod	Bc. Michael Gloger
2 Ponuka	Celý tím
3.1 Manažment komunikácie	Bc. Michael Gloger
3.2 Manažment monitorovania	Bc. Tomáš Jánošík
3.3 Manažment rizík	Bc. Stanislav Kubica
3.4 Manažment plánovania	Bc. Rastislav Kostrab
3.5 Manažment podpory vývoja	Bc. Šimon Kompas
3.6 Manažment dokumentácie	Bc. Michal Walder

	Bc. Daniel Kĺč
3.7 Manažment kvality	Bc. Daniel Kĺč
4.1 Metodika tvorby user stories	Bc. Michael Gloger
4.2 Metodika manažmentu zberu požiadaviek	Bc. Michael Gloger
4.3 Metodika vybraných procesov čitateľnosti a konvencií kódu	Bc. Daniel Kĺč
4.4 Metodika testovania	Bc. Daniel Kĺč
4.5 Metodika manažmentu vývoja modulov	Bc. Daniel Kĺč
4.6 Metodika manažmentu chýb	Bc. Stanislav Kubica
4.7 Metodika evidencie úloh	Bc. Rastislav Kostrab
4.8 Metodika dokumentácie zdrojových súborov	Bc. Michal Walder
4.9 Metodika verziovania	Bc. Šimon Kompas
4.10 Metodika prípravy podkladov na stretnutie so zákazníkom	Bc. Tomáš Jánošík
4.11 Metodika iterácií projektu	Bc. Tomáš Jánošík
4.12 Metodika manažmentu chýb	Bc. Stanislav Kubica
4.13 Metodika plánovania úloh	Bc. Rastislav Kostrab
4.14 Metodika manažmentu	Bc. Šimon Kompas

projektového repozitára a verzií	

2 Ponuka

2.1 Predstavenie tímu

2.1.1 Bc. Michael Gloger

sa orientuje na tvorbu webových a mobilných aplikácií pre platformu Android. Pracuje s Oracle, PostgreSQL, webovými službami a s webovou grafikou. Jeho preferovaný jazyk je Java.

2.1.2 Bc. Rastislav Kostrab

sa orientuje na technológie rozsiahlych informačných systémov. Okrem toho sa zaujíma aj o vývoj natívnych aplikácií pre platformu Android. Preferuje open-source technológie a jeho silnou stránkou je programovací jazyk Java. Okrem toho ovláda prácu s databázovými technológiami PostgreSQL a MySQL.

2.1.3 Bc. Šimon Kompas

ovláda technológie ako Java, SQL databázy, trocha menej GWT, spring a ďalšie biznis technológie spojené s jazykom Java. Popri štúdiu sa venuje grafovým algoritmom nad sieťami malého sveta a po novom ho začala zaujímať bioinformatika, konkrétne oblasť DNA a skladania fragmentov do celku. Po technologickej stránke chce obohatiť svoje vedomosti o funkcionálny jazyk Scala, ktorý je možné kombinovať s klasickou Javou a je spustiteľný na JVM.

2.1.4 Bc. Tomáš Jánošík

sa venuje databázovým technológiám, vývoju aplikácii využívajúcich veľké dáta a spracovaniu týchto dát sofistikovanými algoritmami. Skúsenosti má aj s vývojom webových aplikácii a komunikáciou cez Internet, či cez iné siete.

2.1.5 Bc. Daniel Kĺč

sa orientuje popri škole na refaktorizáciu starších a vývoj nových algoritmov v oblasti experimentálnej a jadrovej fyziky na OJF FÚ SAV. V diplomovej práci sa chce venovať v rámci bioinformatiky sekvenovaniu DNA. Špecializuje sa na paralelizáciu a aspekty implementácie spojené so zložitosťou algoritmov a jej redukciou. Vyvíja predovšetkým v jazyku C++.

2.1.6 Bc. Stanislav Kubica

sa orientuje na technológie ako Java, SQL databázy, PHP, JavaScript (jQuery), HTML5, CSS3 a iné webovo orientované technológie. Ovláda databázy MySQL a PostgreSQL. Jeho preferovaným jazykom je Java, prípadne PHP ak sa jedná o webové aplikácie.

2.1.7 Bc. Michal Walder

sa orientuje hlavne na jazyk Java, na vývoj J2EE aplikácií ale aj mobilných aplikácií pre platformu Android. Orientuje s v Enterprise aplikáciách pre platformu Java a ovláda technológie ako JSF, Struts, EJB, JPA a pod. Ovláda databázy MSSQL, MySQL i Oracle. Je orientovaný na open-source riešenia.

2.2 Ponuka 1: Analýza výsledkov výskumu

2.2.1 Návrhy na implementáciu:

Téma ohodnocovania výsledkov výskumu je zaujímavá téma podstatná pre správne rozdeľovanie prostriedkov na výskum, meranie určitej prestíže a v neposlednej rade i zaujímavosti témy, ktorej sa ten ktorý výskum venuje. Tento problém sa dotýka aj nás študentov, pretože každý čerpal z nejakého zdroja informácií, ideálne z nedávneho výskumu. Je teda zaujímavé pre študenta vedieť, či zdroj, z ktorého sa chystá čerpať je hodnoverný, uznávaný i aktuálny.

Podstatnou vlastnosťou nášho tímu je komplexnosť s akou by sme dokázali pristúpiť k problému. Máme v tíme ľudí, ktorí sú schopní a nadšení pre matematickú stránku algoritmov rozlišovania medzi entitami, máme ľudí, ktorí dokážu prísť s inovatívnou vizualizáciou dát a taktiež skúsenosti s veľkým objemom dát.

Zaujímavé by bolo odhaľovať a graficky reprezentovať rôzne závislosti medzi dokumentami, pričom faktorov je veľa a je potrebných veľa experimentov. Je zaujímavé zistiť, aké závislosti ovplyvňujú zvýšený výskyt daného dokumentu v citačných indexoch, čo by malo byť spravidla hlavne kredit jeho autora, avšak môžeme nájsť aj závislosti nečakané. V oblasti riešenia nejednoznačnosti existuje mnoho algoritmov, z ktorých by sme chceli vybrať vhodný pre naše účely. Možnosťou bolo riešiť problém nejednoznačnosti sémantickým prístupom, avšak naše úsilie by sme chceli venovať skôr iným aspektom.

Chceli by sme vytvoriť systém, ktorý by bol nielen schopný údaje získavať a vyhodnocovať, ale taktiež aj prezentovať výsledky ohodnocovania koncovým používateľom, aby sme tak pomohli používateľom pri hľadaní dokumentov.

Systém by sme mohli vylepšiť o grafickú vizualizáciu dokumentov v podobe grafov.

Graf by reprezentoval, ktoré dokumenty majú referenciu na iné dokumenty, a takisto by reprezentoval čas ich vytvorenia a relevantnosť k danej téme, čo by sprehľadnilo evidenciu a vyhľadávanie zdrojov pri výskume. Podobne by bolo možné spraviť aj vyhľadávanie.

2.3 Ponuka 2: Virtuálna FIIT na mobile

2.3.1 Cieľ práce:

Navrhnúť a implementovať mobilnú aplikáciu pre platformy Android, Windows phone, iOS ako aj web, ktorá by so svojím používateľským rozhraním a funkcionalitou zodpovedala požiadavkám každého študenta našej školy. Cieľom je zjednodušiť hľadanie informácií týkajúcich sa štúdia a skrátiť čas ich vyhľadávania.

2.3.2 Výstup z práce:

Multiplatformová webová aplikácia zabalená pomocou PhoneGap–u do natívnych aplikácií. Vzhľadom na to, že používanie notebookov na našej škole je veľmi bežné, tak sme premýšľali sme aj nad čisto webovou variantou informačného systému, ktorý by sa od svojej mobilnej verzie líšil používateľským rozhraním prispôsobeným pre väčšie obrazovky.

2.3.3 Cieľoví používatelia (Personas):

Študenti, učitelia, návštevníci školy, …

Počas vývoja aplikácie by sme chceli prototyp testovať za pomoci jej budúcich používateľov. Takýmto spôsobom by sme zhromažďovali pripomienky a skúmali spôsob interakcie ľudí s navrhnutým používateľským rozhraním.

2.3.4 Manažment tímu

Pre zvýšenie efektivity práce plánujeme použiť známy nástroj pre riadenie rozdeľovania úloh v tímových projektoch: JIRA. Svoje zdrojové súbory, dokumentácie a iné digitálne dáta súvisiace s našou prácou budeme medzi zdieľať pomocou verziovacieho systému GIT (resp. podľa potreby a možností SVN).

2.3.5 Nápady pre zlepšenie funkcionality:

- offline synchronizácia (nie každý má mobilný internet, či eduroam konto)
- informácie o jedálňach, učebniach, študijnom oddelení, ...
- umožnenie vyhľadávania profesora v budove (podľa jeho rozvrhových akcií)
- umožniť používateľom pridávať skratky na jednotlivé funkcionality, ktoré využíva najčastejšie. Podobne ako hlavná stránka nášho AIS-u, alebo HOME screen smartfónov.
- prehľadné pozeranie osobného rozvrhu (prípadne rozvrhov kamarátov)
- rýchle dohľadanie informácií k jednotlivým predmetom (prípadne prelinkovanie s prácami ostatných tímov - napríklad CQA komunitný systém otázok a odpovedí).
- pripomienkovanie (notifikácie) k vybraným rozvrhovým akciám
- push notifikácie pre rôzne školské eventy (info o prihlasovaniach na rozvrhové akcie... a iné informácie bežne oznamované iba mailom v rámci AIS)
- mapa školy
- lokalizácia pomocou QR kódov rozmiestnených po škole, zobrazenie cesty do učební

2.3.6 Potenciál aplikácie

Myslíme, že naša aplikácia by bola úspešná medzi študentmi a zamestnancami fakulty z toho dôvodu, že sami sme študenti a z vlastnej skúsenosti poznáme ako je občas problematické pristupovať k niektorým informáciám ohľadom štúdia.

To neznamená, že AIS je zlý, ale že veľa informácií nie je na tom istom mieste. Čas vyhľadávania sa tak výrazne zvyšuje až kým sa študent radšej opýta svojich kamarátov. Informácie z druhej ruky však už nemusia byť presné a študent si ich nemusel hneď zapísať na prehľadné miesto, kde by ich opäť rýchlo vedel dohľadať.

Chceli by sme aj svojich spolužiakov osloviť (napríklad pomocou ankety) aby sme zistili, aký typ informácií je pre nich najdôležitejší a tieto informácie by sme zaradili medzi tie najľahšie dostupné (počet kliknutí musí byť vždy minimálny- tj žiadne zbytočné ovládanie kt človek nechce používať). Dôležitou funkcionalitou bude možnosť používateľov nastaviť si, ktoré informácie ho zaujímajú a chce ich mať vždy rýchlo prístupné, prípadne ktoré chce mať schované.

2.4 Ponuka 3: Distribuované počítanie na FIIT

Náš cieľ by bolo sprístupniť FIIT GRID a otestovať ho (resp. jeho výkon v rôznych kategóriách). Sprístupniť obojstranne, tj. kontribútorom , ale aj konzumentom výpočtových prostriedkov. Chceli by sme sprístupniť počítanie na desktopoch/serveroch ale aj na mobilných zariadeniach.

V tíme máme členov, ktorí už majú skúsenosti jak s Linuxom tak s Androidom. Bolo by vhodné vytvoriť klienta pre desktop počítače/mobilné zariadenia, ktorý fyzicky nastaví a skonzumuje lokálne výpočtové prostriedky, ale aj webové rozhranie, kde sa konzumenti môžu jednoducho uchádzať o výpočtové prostriedky a na druhej strane kontribútori môžu jednoducho stiahnuť klienta na výpočet.

Funkcionalita klienta môže napríklad zahrňovať čas kedy budú prostriedky zapožičané, objem prostriedkov, alebo napríklad v mobilných zariadeniach aj určenie jednotlivých prostriedkov, ktoré im budú k dispozícii.

Funkcionalita webu môže obsahovať napríklad grafické zobrazenie mapy kontribútorov alebo metriky systému, možnosť stiahnuť si klienta alebo formulár potrebný na uchádzanie sa o prostriedky.

Otestovanie by malo prebehnúť nad reálnym problémom a výpočet by pokiaľ možno mal byť zmysluplný. Je možnosť osloviť týmto smerom napríklad Fyzikálny ústav SAV kde jeden z členov nášho tímu pracuje a získať od nich nejaké reálne, užitočné a dostatočne náročné úlohy. Ďalšou možnosťou je zamyslieť sa nad využitím pri riešení simulačných problémov, kde by sa dal využiť fakt, že údaje sú známe všetkým agentom, potrebné je však pre každého agenta počítať stratégiu. Uplatnenie = simulácia davu, simulácia rannej špičky v MHD...

Pri realizácií tohto projektu je taktiež nutné uvažovať nad teoretickými aspektami distribúcie výpočtov, nech sa maximalizuje využitie zapožičaných prostriedkov.

Vzhľadom na to, že tento projekt je predovšetkým orientovaný na OS Linux, Unix, (prípadne Android a iOS), má zmysel uvažovať nad spôsobom, ako by sa s čo možno najmenším úsilím do GRIDu dali zapojiť počítače s OS windows (a tým pádom by boli napojiteľné do GRIDu aj počítačové učebne lokalizované na FIIT).

K týmto cieľom je možne využiť naše individuálne schopnosti a skúsenosti, ale napríklad je možnosť získať odpovede na konkrétne otázky ľudí ktorý prispievali, konzumovali a vyvíjali CERN GRID, s ktorými sa jeden člen nášho tímu osobne pozná.

2.5 Príloha A: Priorita výberu tém

1. Virtuálna FIIT na mobile
2. Analýza výsledkov výskumu
3. Distribuované počítanie na FIIT
4. Webový komunitný systém otázok a odpovedí
5. Digital SweatShop
6. Trojdimenzionálne UML
7. Vizualizácia informácií v obohatenej realite
8. Zábavný systém pre spolucestujúcich v automobile
9. Interaktívne hry na mobile s multimediálnym obsahom
10. Monitor programátora v IDE
11. Prehliadka kódov v tímových projektoch
12. Robotický futbal
13. Sledovanie pohľadu pri používaní aplikácií

2.6 Príloha B: Rozvrh členov tímu vyhradený na tímové stretnutia

Meno	**PON**	**UTO**	**STR**	**ŠTV**	**PIA**
Rastislav Kostrab	13 - 16	15 +	do 15	9 - 15	-
Michael Gloger	-	15 +	11 - 13/14	do 15 cca	dohodou
Stanislav Kubica	13 - 16	15 +	do 15	9 - 15	-
Michal Walder	18 +	15+	do 13	11 - 15	dohodou
Tomáš Jánošík	do 14	do 11 U 15+	do 13	11 - 15	dohodou
Daniel KĺČ	do 14	do 11, od 15 do 18	do 13	11 - 15	dohodou

Šimon Kompas	12-16 U 18 +	15 +	11-15	18 +	-
Prienik (cca)	**-**	**15 +**	**11 - 14**	**11-14**	**-**

3 Úlohy členov v tíme

Členovia tímu zastávajú nasledovné manažérske role:

Meno	**Rola**
Bc. Michael Gloger	vedúci tímu
Bc. Rastislav Kostrab	manažér plánovania
Bc. Šimon Kompas	manažér podpory vývoja
Bc. Tomáš Jánošík	manažér monitorovania projektu
Bc. Daniel Kĺč	manažér kvality manažér dokumentácie
Bc. Stanislav Kubica	manažér rizík
Bc. Michal Walder	manažér dokumentácie

3.1 Manažment komunikácie

Zodpovedná osoba: Michael Gloger

Úloha manažéra komunikácie v našom tíme je vybrať vhodné komunikačné a kolaboračné nástroje a zabezpečiť aby ich členovia tímu vedeli efektívne využívať. Ďalšou hlavnou úlohou je organizovať tímové stretnutia a pripraviť si plán na každé stretnutie.

3.1.1 Stretnutia k tímovému projektu

V rámci tímového projektu sa konajú dva druhy stretnutí:

- Pravidelné stretnutia s vlastníkom produktu, pani Andrejčíkovou (pravidelne každý štvrtok od 11:00)
- Stretnutia tímu (v skole, niekoľko krát do týždňa)

V rámci pravidelných stretnutí s vedúcim sa preberá plnenie jednotlivých úloh a plánujú sa nové. Na stretnutiach sa zúčastňoval celý tím. Na stretnutiach s vlastníkom produktu sa zúčastňuje celý tím a trvajú približne 2 - 3 hodiny. Spoločné stretnutia mali nasledovný priebeh:

1. Vedúci tímu zhrnul plán na aktuálne stretnutie a uviedol aké výstupy sa z neho očakávajú. V tejto fáze sa jednotlivé body ešte nerozoberali
2. Vedúci tímu zhrnul vlastníkovi produktu čo sa stihlo od posledného stretnutia a dal priestor členom tímu vyjadriť sa jednotlivým bodom, podľa toho kto na čom pracoval.
3. Zhrnuli sa veci, na ktorých sa bude pracovať najbližšie
4. Vedúci tímu pokračoval s plánom stretnutia podľa potreby a diskutoval s členmi tímu a vlastníkom produktu

Stretnutia členov tímu sa konali 2-3 krát do týždňa vo forme stand-up meetingov. Na týchto stretnutiach členovia tímu konzultovali svoju prácu a svoje výstup s vedúcim tímu a s ostatnými členmi.

3.1.2 Komunikačné nástroje

Väčšina komunikácie prebiehala hlavne v rámci tímových stretnutí, avšak občas je nutné veci rieši'' aj pomocou online komunikačných a kolaboračných nástrojov. V rámci našej práce sme používali nasledovné nástroje:

3.1.2.1 Email

Použitý na komunikáciu v rámci celého tímu a komunikáciu s vlastníkom produktu. Pre komunikáciu bol použitý spoločný email tp-1314-10@googlegroups.com.

3.1.2.2 Hangout (Google talk)

Použitý na rýchle dohadovanie sa medzi členmi tímu. Využívaný najmä medzi členmi, ktorí na úlohách kolaborovali.

3.1.2.3 Slack

Slack je služba, ktorá ponúka tímom spoločnú komunikáciu vo forme webového rozhrania a mobilných aplikácií. Výhoda využívania tohto skupinového chatu je to, že konverzácie môžu byť rozdelené do rôznych kanálov a je možné jednoducho v týchto konverzáciách vyhľadávať. Rozhodli sme sa pre Slack namiesto Skype z toho dôvodu, že Skype v súčasnosti nepodporuje ukladanie správ na serveri a možnosti komunikácie sú tu obmedzené.

3.1.3 Kolaboračné nástroje

Pre zefektívnenie práce bolo nutné využívať kolaboratívne nástroje, vďaka ktorým vedeli členovia tímu spolupracovať na úlohách aj keď práve nesedeli pri jednom počítači. Medzi tieto úlohy patrí napríklad písanie dokumentácie a zdrojových kódov.

3.1.3.1 Google Drive

Manažér komunikácie vytvoril v Google Drive priečinok, do ktorého majú prístup všetci členovia tímu. V tomto priečinku sa nachádzajú všetky dokumentácie na ktorých sme potrebovali kolaboratívne pracovať.

3.1.3.2 JIRA

Pomocou JIRA evidujeme svoje šprinty a úlohy. Každá úloha má názov, popis prideleného člena tímu a iné atribúty, na základe ktorých vieme svoje úlohy triediť. Tento nástroj nám tiež umožňuje robiť reporty práce. Každý člen je povinný vykazovať si čas, ktorý danej úlohe venoval a písať sem reporty svojej práce. V každom reporte je uvedené čo stihol, čo nestihol a aký je záver z jeho práce.

3.1.3.3 Stash

Stash poskytuje git repozitáre, ktoré sú priamo integrované s JIRA. Pri commitovaní stačí do sprievodnej správy zadať číslo úlohy a JIRA automaticky tento commit spáruje s danou úlohou.

3.1.3.4 Confluence

V Confluence vytvárame wiki stránky k jednotlivým problematikám, ktoré sme riešili. Každý člen je počas svojej práce do týchto stránok pridávať výstup svojej práce aby ho ostatní členovia odtiaľ mohli čítať.

3.1.3.5 Crucible + FishEye

Ďalší nástroj integrovaný s JIRA. Slúži na vytváranie review kódu k obsahom repozitárov.

3.2 Manažment monitorovania

Zodpovedná osoba: Tomáš Jánošík

Manažér monitorovania projektu je zodpovedný za viditeľnosť stavu projektu, za to, aby poskytoval informácie o tomto stave, za vykazovanie plnenia úloh a za kompletnosť výstupov.

V rámci tejto funkcie teda dochádza ku kontrole ostatných členov tímu, aby zaznamenávali svoje výstupy do prostredia na manažment úloh (JIRY). Vďaka tomuto nástroju máme mnohé činnosti zautomatizované a je jednoduchšie vykazovať výsledky práce. Takúto kontrolu ostatných členov vykonáva manažér monitorovania projektu v súčinnosti s vedúcim tímu. Počas uplynulého času boli priebežne kontrolované úlohy, aby boli kompletné, boli monitorované z hľadiska stavu, množstva vykonanej práce a množstva práce potrebnej na dokončenie.

Tieto metriky boli brané do úvahy v zmysle počtu hodín odhadovanej práce a vykázanej práce. Na podporu týchto metrík slúžia tri atribúty každej úlohy obsiahnuté v JIRE v sekcii Time Tracking:

- Estimated - odhadovaný čas potrebný na splnenie úlohy
- Remaining - zostávajúci čas potrebný na splnenie úlohy
- Logged - čas spotrebovaný na plnenie úlohy

Na základe týchto metrík sa dá pomerne dobre usudzovať, v akej fáze sa daná úloha nachádza. Podpore predstave o stave úlohy napomáha taktiež percentuálne vyjadrenie týchto metrík taktiež obsiahnuté v JIRE.

Okrem týchto metrík bolo monitorovanie stavu úloh realizované na stretnutiach tímu, kde boli členom tímu kladené otázky ohľadne postupu na projekte. Túto časť sme preberali doteraz veľmi neformálne, pričom sme z týchto informácii nevyvodzovali dôsledky, preto by sme chceli túto oblasť viac prepracovať a z výsledkov monitorovania vyvodzovať dôsledky.

Monitorovanie stavu projektu prebiehalo pravidelne na stretnutiach so zákazníkom (pedagogickým vedúcim tímu), kde mu boli odprezentované výsledky práce, ktoré boli následne zákazníkom zhodnotené. V tomto smere môžeme zhodnotiť postup prác ako uspokojivý, pretože vyriešené úlohy, ako vidno zo záznamov zo stretnutí, dosahovali vždy aspoň polovicu všetkých úloh.

Šprint	**Počet úloh splnených**	**Počet úloh čiastočne alebo vôbec nesplnených**
1 “Amstel”	3	2
2 “Budweiser”	4	1
3 “Carlsberg”	7	7
4 “Duff”	11	2

Tretí šprint bol z hľadiska náročnosti ostatných predmetov najťažší, preto sme dosiahli len polovičnú úspešnosť vyriešených úloh. Taktiež sme sa v tomto šprinte rozhodli prehodnotiť dôležitosť niektorých úloh tak, aby sme mali hotový prototyp a aby sme ho podľa zásad agilného vývoja ponúkli zákazníkovi čo najskôr ponúkli na vyskúšanie.

V šprinte 4 sme teda splnili takmer všetky úlohy, ktoré boli zadané, vďaka prístupom, ktoré sme sa naučili, kvalitnému plánovaniu a dobrému pracovnému prístupu.

Na skvalitnenie a podporu priebehu stretnutia so zákazníkom bola vytvorená metodika prípravy na stretnutie, aby toto stretnutie prebiehalo hladko a riešili sa dôležité otázky týkajúce sa ďalšieho vývoja. Táto metodika

má za cieľ odstrániť nedostatky v komunikácii tímu ohľadne stavu ostatných úloh, pretože toto je podstatné pri udržiavaní si predstavy o veľkom obrázku projektu.

V rámci spomínanej metodiky boli vyvinuté postupy prípravy na stretnutie, kde využívame podporné nástroje, ako je JIRA, pričom tieto nástroje sme sa najprv museli naučiť používať, preto výsledky zautomatizovaných a metodicky pokrytých procesov sme očakávali v neskorších šprintoch. Niektoré nástroje sme sa používať naučili, naopak nástroj na prehliadky kódu Fisheye sme doteraz využili len málo.

Využívané mali byť grafy podporujúce transparentnosť stavu projektu. Tieto grafy zobrazujú vyriešené úlohy v zmysle prioritizácie úloh a v opozícii k nevyriešeným úlohám s vysokou prioritou. Bližšie sú tieto grafy rozobrané v spomínanej metodike. Používali sa ako príprava na stretnutie a ako základ prezentovaných splnených a nesplnených úloh.

Dôležitým bodom podpory monitorovania projektu bolo výraznejšie využívanie tabuľky pri tímových stretnutiach, ako aj model architektúry systému znázorňujúci hotové a nehotové komponenty. Na poslednom stretnutí so zákazníkom bol tento model prezentovaný vo forme A2, čo pomohlo pri komunikácii so zákazníkom a pri prezentovaní výsledkov našej práce.

Tieto dva aspekty pomohli v rámci tímu udržiavať "veľký obraz" o projekte, pomohli identifikovať kritické komponenty, ktoré musia byť implementované čo najskôr, ako aj zlepšiť komunikáciu medzi tímovými kolegami, ktorí mali na starosti podobné úlohy.

Na stretnutiach so zákazníkom boli uskutočnené zistenia, že výstupy prác jednotlivých členov nie sú dostatočne popísané v systéme na evidenciu takýchto výstupov (Confluence). často tu chýbali podstatné údaje, ktoré by mali byť verejné pre všetkých členov tímu, avšak nachádzali sa len v privátnej podobe u človeka, ktorý tieto výstupy vyprodukoval.

Naviac bola podoba výstupov neformálna, preto sme v metodike prípravy pokryli aj kontrolu zapísania výstupov do Confluence, aby tak mali ostatní členovia tímu prístup k podstatným informáciám pri čo najmenšom úsilí.

3.3 Manažment rizík

Zodpovedná osoba: Stanislav Kubica

Úlohou manažéra rizík je identifikovať a predvídať problémy, ktoré môžu negatívne ovplyvniť vývoj projektu, prípadne ohroziť jeho celkovú realizáciu. Vo všeobecnosti sa rizikám vyhnúť nedá, je však možné ich do veľkej miery eliminovať vhodným spôsobom riadenia projektu.

3.3.1 Analýza rizík

Hlavné riziká identifikované v prvej fáze projektu:

1. Nesprávne pochopenie problematiky
2. Nedostatok času členov tímu
3. Nedostatočná dokumentácia minuloročného systému

4. Nerealistické časové odhady
5. Nesprávny manažment tímu
6. Nesprávny výber technológií

3.3.2 Popis rizík

Nesprávne pochopenie problematiky: Napriek snahe o čo najlepšie pochopenie problematiky daného projektu, takéto riziko existovalo. Nikto z nášho tímu sa totiž podobnej problematike ešte nikdy nevenoval, čo mohlo v začiatkoch spôsobiť veľké nedostatky pri analýze minuloročného projektu alebo novej funkcionality systému.

Nedostatok času členov tímu: Vzhľadom na náročnosť zimného semestra bolo pravdepodobné, že jednotliví členovia tímu nebudú mať dostatok času na plnenie svojich povinností v pozíciách manažérov tímu. Ďalším dôvodom zváženia tohto rizika bol fakt, že všetci členovia tímu popri štúdiu pracujú v rôznych firmách.

Nedostatočná dokumentácia minuloročného systému: Systém, ktorý bol výsledkom práce predchádzajúceho timu, bol naprogramovaný pomocou technológie C# a PHP, čo mohlo spôsobiť problémy pri spätnej analýze ich kódu. S programovacím jazykom C# totiž z nášho tímu vedeli pracovať len dvaja ľudia. Tiež dokumentácia, ku ktorej sme mali spočiatku prístup, neobsahovala všetky potrebné informácie z dôvodu licenčných podmienok SCOPUS-u a WoS. Vo všeobecnosti dokumentácia predchádzajúceho projektu neobsahovala informácie, potrebné na spätnú analýzu uvedeného projeku.

Nerealistické časové odhady: V jednotlivých šprintoch mohli byť naplánované úlohy s nerealistickými časovými odhadmi. Dôvodom predĺženia času vykonávania môžu byť technické problémy (napr. pri vytváraní databázy, parsovaní Google Scholar, parsovaní a importovaní CREPČ xml) alebo to môže byť dôsledkom druhého rizika a to nedostatkom času jednotlivých členov.

Tomuto riziku sa nebolo možné spočiatku vyhnúť, keďže sme nemali dostatok informácií o problémovej oblasti ktorej sme sa venovali. Neskôr sa manažment časových odhadov zlepšil aj vďaka použitia metodiky “planning poker cards”.

Nesprávny manažment tímu: Vzhľadom na našu neskúsenosť v úlohách manažérov tímu a malé možnosti v zmene manažmentu (a jeho spôsobu) je možným rizikom demotivácia tímu a následné nesplnenie vytýčených cieľov v určenom termíne. Riziko zvyšoval tiež fakt, že v praxi sme sa s agilnou metódou vývoja SCRUM stretli na tomto projekte po prvý krát.

Nesprávny výber technológií: Pri procese analýzy sme sa rozhodli použiť niektoré funkčné časti minuloročného projektu (napr. importéry WOS a SCOPUS) a tým pádom aj technológie, ktoré používajú (C# a MongoDB). Keďže náš systém bude bežať na technológiách Java, PostgreSQL a C, môže použitie ďalších dvoch technológií pôsobiť neprehľadnosť systému a zbytočné komplikácie pri ladení a testovaní kódu.

3.3.3 Aktuálny stav

V súčasnosti máme väčší prehľad o problémovej doméne, ktorej sa náš projekt týka. Tiež sme si osvojili prácu v softvérovom tíme a manažment tímu. Ten nie je stále na dostatočnej úrovni, keďže zastávame manažérske

role po prvý krát. Na predchádzajúcich chybách sme sa však naučili na čo si máme v budúcnosti dať pozor pri riadení tímu.

Ako najväčšie riziká v súčasnosti považujeme nedostatok času členov tímu a nerealistické časové odhady, ktoré spolu priamo súvisia. Hlavne prvé riziko je náročné eliminovať, keďže okrem tohoto projektu pracujeme na iných školských projektoch a zadaniach. Riziko časových odhadov sa nám darí zmenšovať, keďže naše vedomosti o problémovej doméne a teda aj náročnosti vykonávania jednotlivých úloh sú oveľa väčšie ako na začiatku projektu.

3.3.4 Tabuľka rizík

Riziková oblasť	**Riziko**	**Pravdepodobnosť**	**Vplyv**
Znalosti	Neznalosť problémovej domény	vysoká	Potreba štúdia danej problematiky
Znalosti	Nízka kvalita informácií z knižníc	vysoká	Potreba kontroly dát importovaných z knižníc
Tím	Nesprávny manažment	stredná	Potreba študovania manažmentu softvérového tímu a zlepšovania riadenia
Tím	Nedostatočná komunikácia	stredná	Potreba správneho používania dohodnutých informačných kanálov
Znalosti	Neznalosť predchádzajúceho projektu	vysoká	Potreba štúdia predchádzajúceho projektu z projektovej dokumentácie a zdrojových kódov
Znalosti	Chýbajúca dokumentácia predchádzajúceho	stredná	Potreba štúdia zdrojových kódov, prípadne analýza danej funkcionality zo zdrojov

	riešenia		dát (napr. CREPČ)
Tím	Prideľovanie na sebe závislých úloh rôznym ľudom	stredná	Potreba prerozdelenia úloh jednotlivým členom tímu
Tím	Nerovnomerné rozdeľovanie úloh	stredná	Potreba prerozdelenia úloh, aby všetci členovia tímu mali priradené rovnaké percento úloh v danom šprinte
Technológie	Nedostatočné testovanie	stredná	Potreba vytvorenia a priradenia úloh o testovaní vytvorenej funkcionality
Tím	Odchod člena tímu	nízka	Potreba prerozdelenia úloh a povinností
Znalosti	Podcenenie analýzy	vysoká	Potreba ďalšej analýzy danej problematiky.

3.4 Manažment plánovania

Osoba zodpovedajúca za manažment plánovania: Rastislav Kostrab, Bc.

3.4.1 Míľniky

Dátum	**Cieľ**
5.12.2013	Vytvorenie prototypu front-endu
15.2.2014	Dokončenie refaktorizácie minuloročného riešenia
1.4.2014	Počítanie publikačných indexov

10.5.2014	Vytvorenie webovej služby pre porovnavanie záznamov

3.4.2 Harmonogram práce

Nasledujúca tabuľka znázorňuje plánovanie šprintov. Šprinty prebiehajú v rámci vývojovej metodiky SCRUM. Šprinty trvajú prevažne 2 týždne. Každý šprint má svoje poradové číslo, názov, vymedzené obdobie a stručný popis úloh (resp. činností), ktoré sa počas daného obdobia riešia.

3.4.3 Plán šprintov

P.č.	Názov	Obdobie	Činnosti
1	Amstel	10.10.2013 – 24.10.2013	● Vytvorenie webovej prezentácie tímu ● Inštalácia VM ● Zefektívnenie algoritmov párovania (Refactoring) ● Štúdie XML CREPČ ● Návrh možností získavania dát z Google Scholar
2	Budweiser	24.10.2013 – 7.11.2013	● Analýza front-endu systému ● Vytvorenie PostgreSQL databázy ● Analýza citačných štýlov ● Porovnávacie metódy ● Školenie JIRA
3	Carlsberg	7.11.2013 – 21.11.2013	● Naplnenie DB s dátami z XML CrepČ ● Realizácia základnej funkcionality frontendu ● Návrh systému pre správu používateľov ● Návrh normalizácie dát ● Vyriešenie blokovania Google Scholar

			• Naplnenie databázy z Google Scholar parsera • Vytvorenie návrhu používateľského rozhrania • Experiment a výber algoritmu na porovnávanie
4	Duff	21.11.2013 – 5.12.2013	Uskutočnenie zmien v databáze Inštalácia Bamboo Inicializácia GWT projektu Inštalácia aplikačného servera Vytvorenie prihlasovania používateľov Analýza a návrh komunikácie C programov a Javy Inicializácia komponentu Scheduler Návrh normalizácie dát Realizácia základnej funkcionality front-endu

3.4.4 Krátkodobé plány

Nasledujúce tabuľky znázorňujú prehľad krátkodobých plánov, t.j. širšie vyjadrenie plánov pre každý šprint. Každá jedna tabuľka teda predstavuje jeden šprint a stručný popis činností, ktoré sa počas daného časového ohraničenia riešia. Za každú činnosť je vždy zvolený člen tímu, ktorý zodpovedá za úspešné zvládnutie danej činnosti vo vymedzenom čase.

3.4.4.1 Šprint č. 1 (Amstel)

Úloha	**Zodp. osoby**
Vytvorenie webovej prezentácie tímu • Vytvorenie webovej stránky pre tím č. 10	Michal Glogner, Bc.
Inštalácia VM • Inštalácia Ubuntu 12.04 Server na školský server • Inštalácia a konfigurácia systému Jira	Michal Glogner, Bc. Šimon Kompas, Bc.
Zefektívnenie algoritmov párovania (Refactoring)	Daniel KĺČ, Bc.

	Tomáš Jánošík, Bc.
Štúdie XML CREPČ	Rastislav Kostrab, Bc Stanislav Kubica, Bc.
Návrh možností získavania dát z Google Scholar	Šimon Kompas, Bc. Michal Walder, Bc.

3.4.4.2 Šprint č.2 (Budweiser)

Úloha	**Zodp. osoby**
Analýza front-endu systému ● Analýza front-endu systému ● Skúmanie jeho prepojenia na DB ● Návrh riešenia (použijeme vytvorený vs. vytvoríme nový)	Bc. Michael Gloger
Porovnávacie metódy ● Implementácia porovnávacích metód ● Porovnanie na syntetických dátach	Bc. Daniel KĺČ, Bc. Tomáš Jánošík
Vytvorenie PostgreSQL databázy ● Vytvorenie dátového modelu v PostgreSQL ● Naplnenie PostgreDB reálnymi dátami ● Pripravenie Tomášovi a Danovi dáta pre porovnávanie	Rastislav Kostrab, Bc. Stanislav Kubica, Bc.
Školenie JIRA	Bc. Šimon Kompas,

• Zorganizovať školenie o používaní Jiry	Bc. Michael Gloger

3.4.4.3 Šprint č.3 (Carlsberg)

Úloha	**Zodp. osoby**
Naplnenie DB s dátami z XML CREPČ • Vytvorenie parsera XML CREPČ • Implementácia systému pre import XML CREPČ do DB	Rastislav Kostrab, Bc.
Návrh systému pre správu používateľov • Analýza riešenia správy používateľov z minulého roku • Refaktoring správy používateľov z minuloročného projektu • Návrh dátového modelu správy používateľov	Stanislav Kubica, Bc.
Realizácia základnej funkcionality frontendu • Prihlasovacie okno do systému • Prepojiť prihlasovacie okno s tabuľkou používateľov • Testovanie prihlasovacieho okna • Príprava prostredia na tvorbu frontendu	Šimon Kompas, Bc.
Návrh normalizácie dát • Učiaci algoritmus na naučenie prahovej hodnoty pre porovnávanie mien • Algoritmus na výber kandidátov pre porovnanie • Comparacny engine	Daniel Kĺč, Bc. Tomáš Jánošík, Bc.
Vyriešenie blokovania Google Scholar	Michal Walder, Bc.
Naplnenie databázy z Google Scholar parsera	Michal Walder, Bc.

Vytvorenie návrhu používateľského rozhrania	Michal Glogner, Bc.
Experiment a výber algoritmu na porovnávanie	Daniel Kĺč, Bc.

3.4.4.4 Šprint č.4 (Duff)

Úloha	Zodp. osoby
Uskutočnenie zmien v databáze Pridanie stĺpca pre indikáciu nespracovaných záznamov pridanie tabuľky histórií zmien v DB Pridanie stĺpca zdroja a tabuľky obsahujúcej zdroje	Stanislav Kubica, Bc.
Inštalácia Bamboo Získanie licencie k Bamboo	Michal Glogner, Bc.
Inicializácia GWT projektu Vytvorenie Java balíčka pre GWT v Java projekte	Rastislav Kostrab, Bc.
Inštalácia aplikačného servera Inštalácia aplikačného servera Tomcat 7.0 na server projektu Nakonfigurovanie aplikačného servera	Rastislav Kostrab, Bc.
Vytvorenie prihlasovania používateľov vytvorenie serverovej časti pre GWT pre správu prihlasovania používateľov	Michal Glogner, Bc.
Analýza a návrh komunikácie C programov a Javy Zistenie možnosti volania C programov z Javy Definicia komunikačného rozhrania medzi komparátormi a Javou	Daniel Kĺč, Bc.
Inicializácia komponentu Scheduler Vytvorenie Java balíčka pre komponent Scheduler v Java projekte	Tomáš Jánošík, Bc.
Návrh normalizácie dát	Tomáš Jánošík, Bc.

Realizácia základnej funkcionality front-endu	Šimon Kompas, Bc. Michal Walder, Bc.

3.5 Manažment podpory vývoja

Zodpovedná osoba: Bc. Šimon Kompas

Úloha manažéra podpory vývoja v tíme je spravovať, konfigurovať, integrovať a poskytovať vybrané nástroje, ktoré pri vývoji a manažmente projektu uľahčujú kolaboratívny vývoj a vytvárajú podmienky pre jednoduchší manažment projektu. Ďalšou úlohou je správa používateľských účtov a práv k použitým nástrojom a ich funkcionalitám.

3.5.1 Nástroje pre podporu vývoja a manažmentu

3.5.1.1 Verziovací systém, repozitáre a pridružené nástroje

V tíme používame Git ako systém na správu repozitárov a verzií. Tieto repozitáre sme integrovali do aplikácie Atlassian Stash, ktorá ponúka webové rozhranie s prehľadným zobrazením repozitáru a so ochudobnenou paletou operácií vykonávaných nad repozitárom. Ako offline náhradu za Git sme ponúkli používanie grafickej aplikácie Atlassian SourceTree, ktorá ponúka obohatenú paletu operácií nad repozitármi. K aplikácií SourceTree a spôsobu tvorby zmien v nej bola vytvorená základná metodika pre vývojárov v tíme.

Tieto nástroje majú pomôcť v tíme s kolaboratívnou prácou na vývoji systému so zreteľom na sledovateľnosť zmien, prácu s aktuálnymi súbormi a dátami, čím by mali výrazne znížiť časové náklady na vývoj a zjednodušiť plánovanie.

Aplikácia Stash interne využíva Git ako verziovací systém a je nainštalovaná na mimoškolskom serveri vedúcej projektu. Prístup k aplikácií je cez webové rozhranie aplikácie Stash.

3.5.1.2 Code-reviews

Na podporu code-review bol nainštalovaný, nakonfigurovaný nástroj Atlassian Crucible + FishEye, ktorý podobne ako Stash ponúka prezeranie repozitáru, stromu verzií, súbor v repozitári, commitov obohatených o funkcionalitu tvorby code reviews. Táto aplikácia je tak isto ako Stash nainštalovaný na mimoškolskom serveri vedúcej projektu. Prístup k aplikácií je cez webové rozhranie aplikácie Crucible + FishEye.

3.5.1.3 Podpora manažmentu

Na mimoškolský server vedúcej projektu boli nainštalované ďalšie 2 nástroje spoločnosti Atlassian: Jira a Confluence, ktoré budú bližšie popísané v nasledujúcej časti. Všetky na server nasadené nástroje (Jira, Confluence, Crucible + FishEye, Stash) sú navzájom integrované tak, aby ich používanie bolo čo najjednoduchšie a intuitívne. Taktiež bol nakonfigurovaný SMTP server, ktorý je využitý v týchto nástrojoch na odosielanie mailových notifikácií.

Ako podporné nástroje pre manažment, kolaboráciu a komunikáciu sme zvolili už spomenuté Atlassian produkty: Jira s webovým rozhraním, ktorá slúži primárne na manažment úloh. Pre využitie metodiky scrum

bol do Jira nainštalovaný plugin Jira-Agile, ktorý ponúka zjednodušený nástroj pre projekty organizované touto metodikou. Ako wiki projektu slúži webová aplikácia Confluence.

3.5.1.4 Integrácia produktu

Pre podporu manažmentu integrácie a samotnej integrácie systému plánujeme využiť aplikáciu Atlassian Bamboo, ktorá by mala slúžiť ako nástroj na pokročilú integráciu počnúc kompiláciou cez automatizáciu testov až po nasadenie na server, čím chceme zjednotiť a urýchliť proces tvorby softvérového produktu ako celku. Tento nástroj sme chceli podobne ako ostatné nainštalovať na mimoškolský server vedúcej projektu, ale doposiaľ sme nezískali akademickú licenciu a zvažujeme možnosť integrácie s existujúcou inštanciou aplikácie na fakultnom serveri.

3.6 Manažment dokumentácie

Zodpovedná osoba : Bc. Michal Walder
Bc. Daniel Kĺč

3.6.1 Konvencie dokumentovania zdrojových kódov v Jave

Pri softvérových projektoch je dôležité určiť vhodné, spoločné konvencie dokumentovania zdrojových kódov. Je potrebné klásť dôraz na vhodné písanie komentárov v zdrojových kódoch pre jazyk Java tak, aby mohla byť jednoducho vygenerovaná programátorská dokumentácia pomocou nástroja JavaDoc.

JavaDoc je určený na generovanie štruktúrovanej programátorskej dokumentácie zo zdrojových súborova komentárov v Jave. Dokumentácia generovaná nástrojom JavaDoc je vo forme štruktúry HTML stránok. Samotný zápis inštrukcií pre JavaDoc, ktoré sa majú spracovať sa umiestňuje do špeciálneho komentáru začínajúceho /** a uzavretého */. Tento zápis sa umiestňuje pred deklaráciu súboru, triedy či metódy.

Viac o konvenciách dokumentácie zdrojových kódov v Jave v časti - "Manažment zdrojových kódov. Dokumentovanie zdrojových kódov pomocou JavaDoc".

3.6.2 Tvorba dokumentácie

V rámci tímového projektu sme vytvoriť tri tipy dokumentov:

- Dukumentácie k projektu (produkt a riadenie)
- Metodiky
- Zápisy zo stretnutí

Všetky dokumenty majú byť napísané v programe Microsoft Office 2010 poprípade OpenOffice.

Pri vytváraní dokumentácie k riadeniu ako aj k inžinierskemu dielu je potrebné jednotný formát písania. Preto sa v nasšich dokumentoch použijú takéto nastavenia písma:

Fonty

Font bežného písma: Arial

Veľkosť bežného písma: 11pt

Farba bežného písma: čierna

Na zvýraznenie sa použije buď **hrubé** alebo *šikmé* písmo.

Nadpisy nepoužívame, pretože sú nastavované priamo v dokumente k dokumentácií.

Tabuľky

Hrúbka okrajov tabuľky: 1pt

Farba hlavičky tabuľky: svetlo modrá

Odrážky

Typ nečíslovanej odrážky: plný krúžok

Typ nečíslovanej odrážky - 2 úroveň: prázdny krúžok

Typ číslovanej odrážky: číslo s bodkou

Popisy obrázkov

Umiestnenie: pod obrázkom

Zarovnanie: na stred

Začiatok popisu: Obr. x. (kde x je číslo), nasleduje popis obrázku s prvým veľkým písmenom

Pri dokumentácii k inžinierskemu dielu je štruktúra (podľa možnosti) nasledovná:

- Úvod
- Analýza
- Špecifikácia požiadaviek
- Návrh
- Implementácia
- Šprinty

Názov dokumentácie má formát Dokumentacia-typdokumentacie-verzia.doc

Príklad : Dokumentacia-riadenia-1.1.doc

3.6.3 Tvorba zápisnice zo stretnutia

Pre vytvorenie zápisnice zo spoločného stretnutia sa používajú rovnaké konvencie písania dokumentov ako v dokumentáciách. Tá je uvedená v časti - “Tvorba dokumentácie”.

Nadpis zápisu zo stretnutia má formát :

Zápisnica - Stretnutie č.X

Font : Arial

Veľkosť : 18pt (tučné)

Farba : Modrá

Dokument zápisnice zo stretnutia má štruktúru v tvare :

- Téma stretnutia - stručný opis témy stretnutia (jedna až dve vety)
- Dátum stretnutia
- Čas stretnutia
- Miesto stretnutia
- Stretnutie viedol - Člen tímu, ktorý viedol stretnutie
- Prítomní - Zoznam prítomných
- Neprítomní - Zoznam neprítomných
- Zapisovateľ - Zapisovateľ stretnutia
- Obsah stretnutia - Obsah stretnutia štruktúrovaný do maximálne 2-úrovňových odrážok
- Úlohy z predchádzajúceho šprintu
- Ostávajúce úlohy
- Úlohy ďalšieho šprintu

Príklad tabuliek s úlohami:

ID	Popis úlohy	Zodpovednosť	Stav
3.1	Doplniť obsah, členov tímu do webovej prezentácie tímu	Bc.Michael Gloger	50%
3.2	Návrh nových spôsobov efektívnejšieho párovania	Bc. Daniel KĺČ,	OK

		Bc.Tomáš Jánošík	
3.3	Mapovanie XML CREPČ a návrh dátového modelu	Bc.Rastislav Kostrab, Bc.Stanislav Kubica	50%
3.4	Vylepšenie prototypu crawler/parser nad portálom Google Scholar	Bc.Michal Walder	OK
3.5	Konfigurácia nástrojov pre podporu vývoja, SMTP server a pre rekvizít	Bc.Šimon Kompas, Bc.Michael Gloger	OK

3.7 Manažment kvality

Zodpovedná osoba : Daniel KÍč

3.7.1 Určenie štandardov a konvencií

Na začiatku si náš tím určil formát zápisnice zo stretnutia. Ďalej v niekoľkých iteráciach sa vytvoril dokument, ktorý popisuje základné princípy konvencií písania a udržovania zdrojového kódu. Tento dokument kopíruje všeobecne trendy a zásady programovania. Obsahuje všeobecné zásady na udržovanie:

- čitateľnosti kódu
- granularity kódu
- písania dobre štruktúrovaného a ľahko testovateľného kódu
- zásady udržovania modularity
- pokyny k unifikácií riešenia chybových stavov
- základné pokyny k testovaniu

3.7.2 Skrátená verzia dokumentu o udržovaní konvencií kódu

3.7.2.1 Čitateľnosť kódu

Čitateľnosť kódu je intuitívna predstava o význame kódu po prvom prečítaní človekom s dostatočnou kvalifikáciou. Závisí od čitateľného zápisu a zrozumiteľnom zápise entít kódu. Zápis entít kódu je štýl zápisu jazykových konštrukcií vybraného programovacieho jazyka, komentárov a identifikátorov. Z týchto je predovšetkým pre tímovú prácu zaujímavý zápis komentárov a identifikátorov.

Obsah komentárov je v kompetencií manažmentu dokumentácie. Z hľadiska čitateľnosti je dôležitý komentár všade tam, kde je dôležité vysvetliť určitý aspekt implementácie alebo zdôrazniť určitú dôležitú vlastnosť. Taktiež komentáre označujú etapy algoritmov, pre lepšiu oddelenosť logických krokov.

Zápis identifikátorov pozostáva z popisu a mena. Meno odpovedá funkcií a je vytvorené určitým procesom konkretizácie, ktorý je popísaný v metodike vybraných procesov čitateľnosti a konvencií kódu. Komentár je nutný tam, kde je potrebné zdôrazniť vlastnosť alebo je to z globálneho hľadiska nutné. Popis tohto procesu je taktiež v metodike vybraných procesov čitateľnosti a konvencií kódu.

3.7.2.2 Udržovanie granularity kódu

Ide o správne rozčlenenie algoritmu na logické kroky vykonávania. Podrobný popis je obsiahnutý v metodike vybraných procesov čitateľnosti a konvencií kódu.

3.7.2.3 Písania dobre štruktúrovaného a ľahko testovateľného kódu

Ide o logické rozčlenenie implementácie na producentov funkcionality a mediatorov funkcionality. Producenti produkujú podľa parametrov a stavu systému funkcionalitu. Vracajú buďto produkt alebo bližšie nešpecifikovanú chybu. Mediator je funkcionalita, ktorá volá producentov v sequencií, ktorá produkuje ich výstup. Výstupom je produkt alebo špecifikovaná chyba, ktorá je určitým spôsobom interpretovaná do zrozumiteľnej reči.

Takáto štruktúra kódu umožňuje jednoduché štrukturálne testy funkcionality. Spravidla producenti môžu byť otestovaný pre širokú množinu vstupov, kde sa testuje len to, či vráti chybu alebo produkt. Takto otestovaný producenti môžu byť prehlásený za funkčných v bežných situáciách. Na to nadväzuje kontrola mediátorov, pri ktorých sa testuje vrátenie produktu alebo vrátenie konkrétneho chybného hlásenia. Keďže po prvej úrovni testov vieme modelovať správanie producentov, testy mediátorov sú špeciálne dizajnované na vrátenie špecifickej chyby, zlyhanie špecifického producenta volaného daným mediátorom, alebo vrátenie produktu v konkrétnej konfigurácií.

Zatiaľ čo pri producentoch je možné v testovanie zahrňujúce v podstate všetky bežné chyby a výstupy, pri mediátoroch to spravidla nie je možné. Snaha pri mediátoroch je otestovať čo najväčší počet “bežných” situácií.

3.7.2.4 Zásady udržovania modularity

V podstate ide o dodržovanie špecifikácie modulu. V prípade nedodržania ide o procesy popisujúce rozdelenie modulu, posunutie nešpecifikovanej funkcionality do modulu, ktorý daný modul volá alebo rozšírenie špecifikácie modulu. Vzhľadom na to, že v prvých týždňoch sa vyvíjali komponenty riešenia menšie samostatné bloky a teda analýzy modularity zatiaľ neboli nutné, tak sa bližšie týmto procesom bude venovať druhá metodika z predmetu MSI.

3.7.2.5 Pokyny k unifikácií riešenia chybových stavov

Ide o definíciu toho, čo je chybový stav, čo je chybová hláška a čo je ošetrenie chybového stavu. V rámci projektu je pre nás

- chybový stav: Kód označujúci buďto zlyhanie komponentu alebo označujúci konkrétne zlyhanie súčasti funkcionality
- chybová hláška: Je hláška v jazyku používateľa, ktorá oznamuje zlyhanie zrozumiteľným textom, ktorý má zároveň uspokojiť používateľa a zároveň má vývojárovi povie, čo zlyhalo.
- Ošetrenie chybového stavu: Je to proces, pri ktorom sa programom na príslušnom mieste detekuje zlyhanie, program následne vypíše zmysluplným spôsobom danú chybu a nakoniec pokračuje vo vykonávaní predpripraveným spôsobom definovaným pre danú chybu.

3.7.2.6 Základné pokyny k testovaniu

Testovanie prebieha za účasti aspoň dvoch osôb na dvoch úrovniach. Testovania sa zúčastní aspoň manažér kvality a vývojár ktorý funkcionalitu implementoval. Prvá úroveň je štrukturálne testovanie funkcionality dvoma sériami vstupov. Prvá séria vstupov je séria definovaná daným vývojárom, druhá séria je upravená a rozšírená prvá verzia vstupov o vstupy definované manažérom kvality. Druhá úroveň je testovanie na reálnych vstupoch.

3.7.3 Kontrola zápisníc zo stretnutí

Kontrola zápisníc prebehla a zápisnice boli označené za vyhovujúce manažérom kvality.

(http://team10-13.ucebne.fiit.stuba.sk/dokumenty.html)

3.7.4 Kontrola dokumentov

Dokumenty sú priebežne kontrolované členmi tímu a aj manažérom kvality. Vzhľadom na momentálnu prezenciu čiastkových dokumentácií (keďže mnoho z práce, ktorá doteraz bola dokončená boli experimenty, alebo samostatné programové súčasti) tak najintenzívnejšia kontrola je realizovaná tímovým kolegom, ktorý na danej úlohe pracuje spolu s autorom dokumentácie (keďže všetky úlohy sú momentálne realizované v pároch).

3.7.5 Kontrola kódov

Vzhľadom na malý počet kódov, ktoré sú bežnými metódami kontrolovateľné (väčšina práce je momentálne v oblasti databáz, webu a sťahovania údajov z webu), tak kontrola prebehla len nad komparátormi publikácií, ktoré po určitej diskusií v tíme (a menších úpravách) boli akceptované.

4 Metodiky

4.1 Metodika tvorby používateľských príbehov

Autor: Michael Gloger

Táto kapitola obsahuje podrobne opísaný proces získavania a formulácie používateľských príbehov. Tento proces je rozšírený o rozbitie user story na podúlohy. Obsahuje tiež opis jednotlivých rolí a zodpovedností ktoré v tomto procese vystupujú.

4.1.1 Vstup procesu

- Požiadavky od budúceho používateľa systému / vlastníka produktu

4.1.2 Výstup procesu

- Naformulované používateľské príbehy v štandardizovanom tvare
- Zoznam podúloh, ktoré musia byť splnené pre dokončenie user story

4.1.3 Role a zodpovednosti

Rola	**Zodpovednosť**
Používateľ, Vlastník produktu	Naformulovať svoje požiadavky na systém, ktoré nemusia byť ešte presne formulované
Vedúci tímu	Manažovanie komunikácie so zákazníkom, zabezpečenie komunikačných kanálov a získanie zoznamu požiadaviek od klienta
Vývojár, Konzultant	Formulácia používateľských požiadaviek do príbehov, ich korekcia, interpretácia, analýza a následne identifikácia podúloh spojených so splnením celej user story

4.1.4 Proces tvorby používateľských príbehov

Krok	**Názov**
1.	Spracovanie vstupu
2.	Formulácia
3.	Vytvorenie podúloh
4.	Zoradenie podľa priority
5.	Uloženie výstupu

4.1.4.1 Spracovanie vstupu

Vstup od klienta je vyjadrený voľnou rečou a je nutné ho previesť na bodový zápis, ktorého časti sú zoradené podľa okruhu, ktorého sa týkajú.

Pri získavaní a validácií požiadaviek sú nutné nasledovné kroky:

- Ak je požiadavka nepresná alebo moc všeobecná treba klienta požiadať o upresnenie
- Požiadavky, ktoré sú mimo dohodnutého scope treba filtrovať a ďalej diskutovať
- Z bodového zápisu treba podobné požiadavky spojiť do jednej

Výstup: *Bodový zápis požiadaviek*

Zodpovední: *Vedúci tímu, Vlastník produktu*

4.1.4.2 Formulácia

Z bodového zápisu určíme pre každú požiadavku:

- role
- ciele/túžby
- dôvody/benefity

Následne tieto požiadavky preformulujeme podľa jedného z nasledujúcich štandardizovaných tvarov:

1. "Ako <rola> chcem <cieľ/túžba>, aby <dôvod>"
2. "Ako <rola> chcem <cieľ/túžba>
3. "Aby som vedel <dôvod/benefit> ako <rola>, chcem <cieľ/túžba>"
4. "Ako <rola> <kedy>, chcem <cieľ/túžba>, aby <dôvod/benefit>"

Príklady:

1. "Ako zamestnanec chcem vidieť svoje odpracované hodiny, aby som si mohol viesť lepšie evidenciu"
2. "Ako projektový manažér chcem vedieť robiť reporty práce zamestnancov"
3. "Aby som vedel lepšie vyhľadávať zamestnancov ako vedúci, chcem ich vedieť vyhľadávať podľa krstného mena a priezviska"
4. "Ako vedúci počas letných prázdnin, chcem vedieť si zobraziť plán dovoleniek mojich zamestnancov aby som vedel lepšie plánovať projekty"

Ak sú požiadavky nie sú opísané veľmi detailne tak ich preformulujeme podľa jednoduchšej user story, ako je napríklad možnosť č. 2. V opačnom prípade si vyberieme jednu z ostatných možností.

Výstup: *Sformulované používateľské príbehy*

Zodpovední: *Vedúci tímu, Vývojár, Konzultant (asistencia vlastníka produktu)*

4.1.4.3 Vytvorenie podúloh

Na user story môže v jednom čase pracovať viac členov tímu. Pre lepšiu evidenciu práce a úloh je ich teda nutné rozdeliť na podúlohy a to tak, aby na každej podúlohe pracoval maximálne jeden člen tímu.

Príklad dekompozície user story:

User story: "Ako výskumný pracovník chcem zoradiť svoje publikácie podľa počtu ohlasov, aby som si vedel lepšie robiť evidenciu citácií"

Podúlohy:

- Pridanie ikonky filtra zoradenia do stránky
- Vytvorenie dopytu do databázy
- Pridanie dopytu do databázy do controllera

Dekompozícia úloh je závislá na súčasnom stave projektu. Ak neexistuje žiadna stránka zobrazenia publikácií je nutné do podúloh zaradiť aj jej vytvorenie a pridanie.

Výstup: *Podúlohy vychádzajúce z používateľských príbehov*

Zodpovední: *Vedúci tímu, Manažér plánovania, Vývojár, Konzultant*

4.1.4.4 Zoradenie podľa priority

Určenie priority používateľských príbehov a podúloh je dôležité aby sme najprv splnili tie úlohy, ktoré sú pre klienta najviac dôležité. Zoradenie priority príbehov ma na starosti vlastník produktu a zoradenie podúloh manažér plánovania a vedúci tímu.

Proces zoradenie používateľských príbehov podľa priority prebieha nasledovne:

1. Vlastníkovi produktu sa predložia všetky príbehy napísane na papieriky
2. Vlastník produktu tieto papieriky zoradí podľa priority
3. Pod každú user story následne zoradí manažér plánovania a vedúci tímu s asistenciou ostatných členov papieriky s podúlohami.

Výstup: *Zoradené príbehy a podúlohy podľa priority*

Zodpovedný: *Vlastník produktu, Vedúci tímu*

4.1.4.5 Uloženie výstupu

Zoradené používateľské príbehy a ich podúlohy, je nutné kvôli lepšej evidencii uložiť do nástroja na projektový manažment. Postup sa môže mierne líšiť vzhľadom na diverzitu týchto nástrojov.

Pridanie user story do JIRA:

1. V hornom menu vyberieme možnosť *“Create issue”*
2. Do *“Issue type”* vložíme typ úlohy *“Epic”*
3. Do “Summary” vyberieme názov user story (respektíve podúlohy)
4. Do *“Priority”* vyberieme jednu z priorít. Napríklad: *“Minor, Major, …”*
5. Do *“Assignee”* vyberieme zodpovedného člena tímu
6. Do *“Description”* pridáme podrobný popis

Pridanie podúloh k user story v JIRA:

Postup je rovnaký ako pri pridaní user story, ale s nasledovnými zmenami:

- V kroku č. 2. *“Issue type”* nebude *“Epic”* ale akýkoľvek iný z dostupných. Napríklad *“Bug, Task”*
- Pribudne ešte jeden krok a to:
    - Do *“Epic link”* sa zo zoznamu vyberie user story, ku ktorej daná úloha patrí

Výstup: *Používateľské príbehy a úlohy evidované v nástroji na projektový manažment (napríklad JIRA)*

Zodpovedný: *Vedúci tímu*

4.2 Metodika manažmentu zberu požiadaviek

Autor: Michael Gloger

Zber požiadaviek a všeobecne manažment komunikácie so zákazníkom tvorí dôležitú rolu v rámci vývoja softvérových systémov. Zlyhanie tejto komunikácie, respektíve jej zanedbanie, má vážny a negatívny vplyv na výslednú kvalitu vyvíjaného produktu.

Táto kapitola obsahuje dokumentáciu k hornej metodike manažmentu zberu požiadaviek a z neho vyplývajúcich procesov. Na začiatku práce sú vymenované všetky role a zodpovednosti, prislúchajúce k jednotlivým procesom. Každý proces je opísaný z hornej (manažérskej) úrovne a detailnejšie pomocou scenára postupnosti krokov. K procesom sú priradené vstupy a výstupy. Na konci práce sú tieto procesy poprepájané pomocou diagramu aktivít na základe ich vstupov, výstupov, štartovacích pravidiel a rozhodnutí.

4.2.1 Role a zodpovednosti

Táto časť obsahuje tabuľku jednotlivých rolí a zodpovedností, priradených k jednotlivým procesom.

Rola	Proces	Zodpovednosť
Vedúci tímu	3.1 Zriadenie komunikačného kanálu	Registrácia komunikačného nástroja
		Vybavenie licencie
		Vytvorenie používateľských účtov
	3.2 Zber požiadaviek na primárny systém	Naplánovanie osobného stretnutia
		Spracovanie zápisu zo stretnutia
	3.3 Vytvorenie používateľských príbehov	Analýza štrukturovaných požiadaviek
		Návrh používateľských príbehov
		Konzultácia používateľských príbehov a náčrtu rozhrania
	3.4 Tvorba modelu prípadov použitia	Spracovanie používateľských príbehov
		Konzultácia prípadov použitia
	3.5 Prezentácia iterácie produktu zákazníkovi	Napísanie dokumentácie o výsledku práce
		Pripravenie prezentácie produktu
		Odprezentovanie iterácie produktu
Manažér podpory vývoja	3.1 Zriadenie komunikačného kanálu	Inštalácia a konfigurácia nástroja
		Vytvorenie základného návodu na používanie
Vlastník projektu	3.2 Zber požiadaviek na primárny systém	Opis požiadaviek na primárny systém
		Zoradenie požiadaviek podľa priority
	3.3 Vytvorenie používateľských príbehov	Schválenie príbehov a rozhrania
	3.4 Tvorba modelu prípadov použitia	Schválenie modelu prípadov použitia
	3.5 Prezentácia iterácie produktu zákazníkovi	Odovzdanie ďalších požiadaviek
Systémový dizajnér	3.3 Vytvorenie používateľských príbehov	Vytvorenie náčrtu používateľského rozhrania
	3.4 Tvorba modelu prípadov použitia	Vytvorenie diagramu prípadov použitia
Celý tím	3.1 Zriadenie komunikačného kanálu	Otestovanie nástroja

4.2.2 Proces zriadenia komunikačného kanálu

Cieľom tohto procesu je zriadenie nového vybraného komunikačného kanálu pomocou komunikačného nástroja. Ku komunikačným nástrojom patria napríklad nástroje na zdieľanie súborov, požiadaviek, úloh a spoločné elektronické kalendáre. V tomto procese nie je opísaný výber nástroja, ale spustenie procesu je podmienené už vybraným nástrojom, ktorý je vstupom pre štart procesu.

Role Vedúci tímu, vlastník projektu, manažér podpory vývoja

Vstup Vybraný nový komunikačný nástroj

Výstup Nakonfigurovaný komunikačný nástroj a pripravené používateľské účty pre členov tímu a vlastníka projektu

Scenár

1. **Registrácia komunikačného nástroja**
   Vedúci tímu kontaktuje prevádzkovateľov nástroja a registruje tím.
2. **Vybavenie licencie**
   Vedúci tímu po potvrdení registrácie tímu predloží prevádzkovateľom informácie o firme, prípadne o akademickej inštitúcii, ktorej je členom. Ak sa jedná o komerčnú licenciu, tak vedúci tímu zabezpečí platobnú transakciu.
3. **Inštalácia a konfigurácia nástroja**
   Vedúci tímu poverí manažéra podpory vývoja odovzdáva mu všetky potrebné informácie pre inštaláciu a konfiguráciu nástroja, ktoré mu po registrácii predložil prevádzkovateľ služby. Manažér podpory vývoja následne nainštaluje a nakonfiguruje komunikačný nástroj pre špecifikované potreby tímu.
4. **Vytvorenie základného návodu na používanie**
   Manažér podpory vývoja vytvorí základný návod na používanie nainštalovaného komunikačného nástroja.
5. **Vytvorenie používateľských účtov**
   Vedúci tímu si vypýta kontaktné informácie (ak ich ešte nemá kompletné) od vlastníka projektu a registruje jeho a všetkých členov svojho tímu do komunikačného nástroja
6. **Otestovanie nástroja**
   Vedúci tímu odovzdá vlastníkovi projektu a členom tímu prihlasovacie údaje a vydá pokyn pre otestovanie nástroja. Ak nástroj je plne funkčný tak proces končí a v opačnom prípade sa vráti na krok 3. kde manažér podpory vývoja vykoná potrebné zmeny.

4.2.3 Proces zberu požiadaviek na primárny systém

Proces, ktorý sa opakuje počas celého vývoja softvéru pomocou agilnej metodiky. Počas tohto procesu prebieha priama komunikácia medzi vlastníkom produktu a vedúcim projektu.

Role Vedúci tímu, vlastník projektu, manažér plánovania, zapisovateľ

Vstup Požiadavky na systém od vlastníka projektu

Výstup Spracované požiadavky, zoradené podľa priority

Scenár

1. **Naplánovanie osobného stretnutia**

   Vedúci tímu zorganizuje osobné stretnutie, na ktorom sa zúčastní systémový architekt, manažér plánovania a vlastník produktu.

2. **Opis požiadaviek na primárny systém**

   Vlastník produktu je vyzvaný k opísaniu jeho požiadaviek na primárny systém. Vlastník produktu opisuje svoje požiadavky voľnou rečou a zapisovateľ zapisuje všetky informácie.

3. **Zoradenie požiadaviek podľa priority**

   Po opise požiadaviek voľnou rečou je vlastník produktu vyzvaný k zoradeniu jeho požiadaviek podľa priorít.

4. **Spracovanie zápisu zo stretnutia**

   Zápis zo stretnutia je spracovaný vedúcim projektu a požiadavky sú prepísane do štruktúrovaného zápisu.

4.2.4 Proces vytvorenia používateľských príbehov

V tomto procese sa transformujú vstupné požiadavky na používateľské príbehy.

Role Vedúci tímu, manažér plánovania, vlastník produktu, systémový dizajnér

Vstup Štruktúrovaný zápis požiadaviek na systém

Výstup Používateľské príbehy

Scenár

1. **Analýza štruktúrovaných požiadaviek**

   Štruktúrované požiadavky sú normalizované vedúcim tímu a podobné úlohy, ktoré odrážajú tú istú funkcionalitu sú zlúčené.

2. **Návrh používateľských príbehov**

   Vedúci tímu v spolupráci s manažérom plánovania navrhne znenie používateľských príbehov.

3. **Vytvorenie náčrtu používateľského rozhrania**

   Systémový dizajnér vytvorí náčrt používateľského rozhrania zo vstupných požiadaviek.

4. **Konzultácia používateľských príbehov a návrhu rozhrania**

   Vedúci tímu odprezentuje vytvorené používateľské príbehy a návrh používateľského rozhrania vlastníkovi produktu a požiada o ďalšie pripomienky. Pripomienky vlastníka produktu sú zapracované.

5. **Schválenie príbehov a rozhrania**

   Ak vlastník produktu súhlasí s návrhom používateľských príbehov a rozhrania, tak sú príbehy prepísané do komunikačného nástroja správy úloh.

4.2.5 Proces tvorby modelu prípadov použitia

Proces v ktorom sú z používateľských príbehov vytvorené modely prípadov použitia pomocou modelovacieho nástroja.

Role Vedúci tímu, vlastník produktu, systémový dizajnér

Vstup Štruktúrovaný zápis požiadaviek na systém, používateľské príbehy

Výstup Model prípadov použitia

Scenár

1. **Spracovanie používateľských príbehov**
   Vedúci tímu spracuje používateľské príbehy a navrhne z nich zoznam prípadov použitia
2. **Konzultácia prípadov použitia**
   Vedúci tímu konzultuje navrhnuté prípady použitia s členmi tímu a s vlastníkom produktu a zapracúva pripomienky.
3. **Vytvorenie diagramu prípadov použitia**
   Systémový dizajnér vytvorí zo zoznamu prípadov použitia diagram.
4. **Schválenie modelu prípadov použitia**
   Vedúci tímu odprezentuje diagram prípadov použitia klientovi. V tejto fáze ho vyzve k zamysleniu sa nad ďalšími funkcionálnymi požiadavkami na primárny systém. Ak vlastník produktu má ďalšie požiadavky tak sa vracia k procesu číslo zberu požiadaviek.

4.2.6 Proces prezentácie iterácie produktu zákazníkovi

V tomto procese je výsledok šprintu odprezentovaný vlastníkovi produktu a sú od neho prijímané pripomienky. Počas tohto procesu sa na základe požiadaviek od vlastníka produktu spúšťa proces zberu požiadaviek.

Role Vedúci tímu, vlastník produktu

Vstup Požiadavky na systém

Výstup Vytvorená iterácia produktu na základe vstupných požiadaviek

Scenár

1. **Napísanie dokumentácie o výsledku práce**
   Vedúci tímu napíše dokumentáciu o práci, ktorá prebehla na aktuálnej iterácii produktu a napíše súhrn implementovaných funkcionalít. Dokument obsahuje tiež zoznam splnených a nesplnených požiadaviek zo šprintu.
2. **Pripravenie prezentácie produktu**
   Vedúci tímu si pripraví prezentáciu implementovanej funktionality v produkte.

3. **Odprezentovanie iterácie produktu**

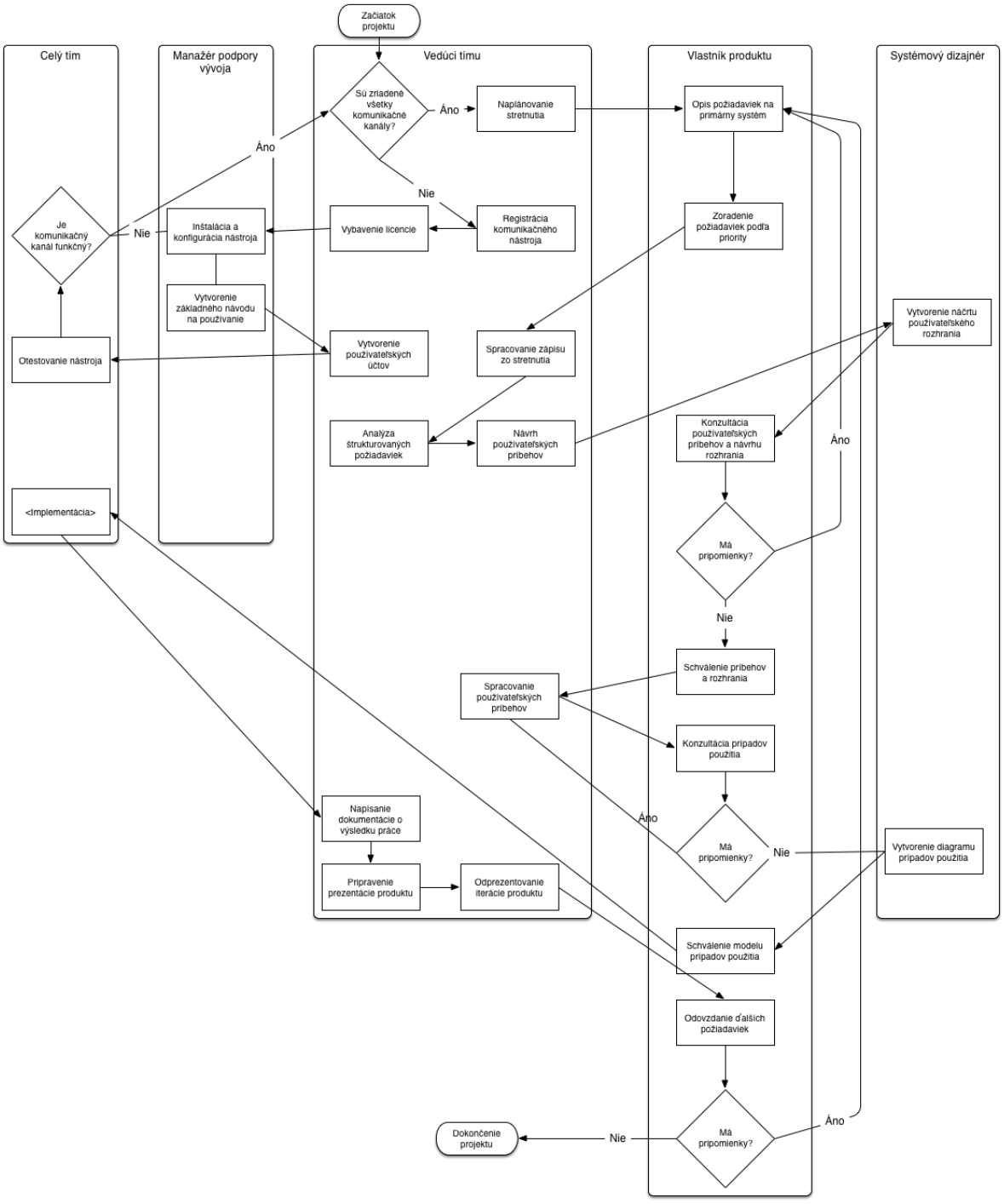
   Osobné stretnutie vedúceho tímu s vlastníkom projektu a predvedenie funkcionality.

4. **Odovzdanie ďalších požiadaviek**

   Ak vlastník produktu má ďalšie pripomienky na systém, prípadne ďalšie požiadavky tak sa spustí proces zberu požiadaviek.

4.2.7 Prepojenie procesov

Táto časť obsahuje diagram aktivít, v ktorom sú poprepájané scenáre opísaných procesov.



4.3 Metodika vybraných procesov čitateľnosti a konvencií kódu

Autor: Daniel KĺČ

Cieľom kapitoly je popísať vybrané procesy, ktoré je treba používať na zmaximalizovanie čitateľnosti, prehľadnosti a testovateľnosti kódu ako aj na uľahčenie manipulácie. Udržovanie vnútornej a vonkajšej integrity ako aj čitateľnosti zdrojových kódov je kontinuálny proces, ktorý prebieha počas celého vývoja projektu, resp. vo všetkých častiach, ktoré v sebe zahrňujú tvorbu, opravu a refaktorizáciu zdrojových kódov.

4.3.1 Role zahrnuté v spomínaných vybraných procesoch

- Vývojár
    - Zodpovedný za integritu a čitateľnosť ním napísaných kódov
    - Zodpovedný za spoľahlivosť ním napísaných kódov
- Manažér kvality
    - Zodpovedný za kontrolu zdrojových kódov – čitateľnosti, integrity a manažovateľnosti

4.3.2 Procesy definované v oblasti čitateľnosti a testovania kódov

Procesy v tejto podkapitole sú definované v bodoch a bližší popis bodov je v ďalšej podkapitole.

4.3.2.1 Proces: Pomenovanie identifikátora

Proces je používaný pri definícií identifikátorov ako sú mená tried, premenných, funkcií, metód a podobne.

Iniciátor: Vývojár

Postup:

1. Identifikujte oblasti pôsobenia a významu v lokálnom kontexte
2. Vytvorte pomenovania
3. V prípade konfliktov s lokálnym prostredím alebo významom použite konkretizáciu
4. Skontrolujte vytvorené pomenovanie

4.3.2.2 Proces: Popis identifikátora alebo modulu

Na zaručenia čitateľnosti kódu je potrebné popísať veľký podiel identifikátorov a aj moduly, aby sa zabezpečila ich interpretovateľnosť a jasnosť.

Iniciátor: Vývojár

Postup:

1. Zvoľte identifikátor
2. Zistite všetky dostupné informácie o ňom

3. Ak je identifikátor premenná
    a. Popíšte premennú
    b. Overte či popis obsahuje všetky potrebné informácie
4. Ak je identifikátor funkcia/metóda
    a. Popíšte funkcie
    b. Popíšte vstupy
    c. Popíšte výstupy
    d. Popíšte chybové stavy
5. Ak je identifikátor trieda
    a. Identifikujte oblasti pôsobenia
    b. Identifikujte špeciálne vlastnosti
    c. Identifikujte špeciálne stavy
6. Ak popisujeme modul
    a. Identifikujte oblasti zamerania definovanej funkcionality
    b. Identifikujte autora
    c. Identifikujte vonkajšie závislosti

4.3.2.3 Proces: Udržovanie granularity kódov

Na zaručenia čitateľnosti, interpretovateľnosti a v neposlednej rade aj manažovateľnosti kódov, je treba udržovať granularitu kódov.

Iniciátor: Vývojár, manažér kvality

Postup:

1. Špecifikujte algoritmus na implementovanie
2. Dekomponujte problém na menšie samostatné bloky – etapy
3. Identifikujte redundantné bloku
4. Identifikujte algoritmy na nižšej úrovni abstrakcie
5. Pokračujte v konkretizácií až dosiahneme elementárne algoritmy
6. Určte redundantné kroky, vhodné na zovšeobecnenú implementáciu v knižniciach

a. Implementácia knižnice/pomocného modulu/triedy/funkcie

7. Implementujte pod-algoritmy
8. Implementujte samotný algoritmus

4.3.3 Podrobný opis krokov

4.3.3.1 Pomenovanie identifikátora

4.3.3.1.1 Identifikácia oblasti pôsobenia a významu v lokálnom kontexte

Identifikuje sa, o aký identifikátor ide, v akom prostredí bude fungovať a aký bude jeho význam pre program/modul/triedu/funkciu/metódu.

4.3.3.1.2 Vytvorenie pomenovania

Vytvorí sa pomenovanie, ktoré odpovedá použitiu a významu premennej.

4.3.3.1.3 V prípade konfliktov s lokálnym prostredím alebo významom konkretizácia

Ak premenná nedostatočne popisuje svoje význam, špecifické vlastnosti alebo je nutné zdôrazniť oblasť použitia poprípade koliduje jej pomenovanie s inou premennou, je pridané druhé pomenovanie cez podtržník. Proces konkretizácie pokračuje podľa potreby dokým je pomenovanie dostatočne konkrétne a zároveň jednoznačné.

4.3.3.1.4 Kontrola vytvoreného pomenovania

Vytvorené pomenovanie je porovnané s dôležitými vlastnosťami a významom daného identifikátora a podľa potreby je zredukované alebo rozšírené ak neodpovedá účelu.

4.3.3.2 Popis identifikátora alebo modulu

4.3.3.2.1 Zvolíme identifikátor

Zvolíme identifikátor, pre ktorý chceme preskúmať potrebu popisu.

4.3.3.2.2 Zistíme všetky dostupné informácie o ňom

Zistíme kontext identifikátora, dôležité vlastnosti, význam v rámci prostredia, funkciu a využitie identifikátora.

4.3.3.2.3 Ak je identifikátor premenná

Ak je identifikátor premenná a je to lokálna premenná menšej metódy alebo premenná bez špecifických vlastností, ktorej popis vystihuje všetky detaily implementácie a logického významu, tak popis nie je nutné uviesť.

Inak ak je premenná významná lokálna premenná alebo globálna premenná, je nutné uviesť špeciálne vlastnosti a význam.

Ak je premenná externá, je treba uviesť zdroj.

4.3.3.2.4 Ak je identifikátor funkcia/metóda

V prípade funkcie je nutné uviesť krátky popis zamerania.

Popis parametrov, formát, obsah.

Popis výstupu, formát obsah.

Popis chýb, spôsob indikovania chyby, formát.

4.3.3.2.5 Ak je identifikátor trieda

V prípade, že je identifikátor tried je nutné uviesť krátky popis.

Popis špeciálnych vlastností.

Popis indikovania chybového stavu triedy (ak je v triede globálne riešená indikácia chýb).

4.3.3.2.6 Ak popisujeme modul

Ak je popisovaný modul je treba uviesť špecializáciu modulu.

Uvedenie autora modulu.

Identifikácia vonkajších závislostí.

4.3.3.3 Udržovanie granularity kódov

4.3.3.3.1 Špecifikácia algoritmu na implementovanie

Je potrebné identifikovať algoritmus, ktorý sa bude implementovať.

4.3.3.3.2 Dekompozícia problému na menšie samostatné bloky – etapy

Prebehne dekompozícia algoritmu na menšie, samostatne vykonateľné etapy. Tieto etapy sú logickými krokmi algoritmu a udržujú rovnakú úroveň abstrakcie.

4.3.3.3.3 Identifikácia redundantných blokov

Identifikácia krokov algoritmu, ktoré sa budú vykonávať viac krát.

4.3.3.3.4 Identifikácia algoritmov na nižšej úrovni abstrakcie

Identifikácie krokov algoritmu, ktoré sú samé o sebe algoritmami.

4.3.3.3.5 Pokračovanie v konkretizácií až dosiahneme elementárne algoritmy

Pre algoritmy, ktoré sú súčasťou tzv. super-algoritmu je nutné znovu odštartovať tento celý proces. Proces udržovania granularity iteruje na pod-algoritmoch až dosiahne úroveň elementárnych algoritmov. Tieto elementárne algoritmy sú na požadovanej úrovni granularity.

4.3.3.3.6 Určenie redundantných krokov, vhodných na zovšeobecnenú implementáciu v knižniciach

Určia sa redundantné kroky na elementárnych algoritmoch a pod-algoritmoch, ktoré sú zovšeobecniteľné a zároveň tento druh algoritmu je redundantný.

Implementácia týchto súčastí kódu do knižnice (čím sa zníži logická redundancia).

Následkom týchto krokov sa zníži logická aj faktická redundancia a zároveň sa zvýši čitateľnosť a interpretovateľnosť kódu.

4.3.3.3.7 Implementácia pod-algoritmov

Implementujú sa algoritmy, ktoré sú súčasťou super-algoritmu.

4.3.3.3.8 Implementácia algoritmu

Z dekomponovaných súčastí a vytvorených knižníc sa implementuje samotný algoritmus, ktorý bol objektom záujmu.

4.4 Metodika testovania

Autor: Daniel KĺČ

Cieľom kapitoly je opísať procesy, ktoré je nutné vykonať pri zostavovaní testov a pri samotnom testovaní.

4.4.1 Potrebné vedomosti

Sekcia obsahuje zoznam okruhov tém, ktoré by čitateľ mal poznať, aby vedel metodiku vykonať.

- Testovanie, taxonómia, prístupy.
- Konvencie kódov.
- Modelovanie prípadov použitia.

4.4.2 Manažment udržovania testovania

Testovanie je súčasť cyklov vývoja softvéru, ktorý validuje funkčnosť softvéru z rôznych hľadísk. Kategórie sú rôzne, napríklad „works as expected“, „meets requirements“, „environment independence“, „satisfies stakeholders“ a i.

4.4.2.1 Účastnícke role

- Vývojár – podieľa sa na tvorbe testov, robí statické a dynamické testovanie, robí white box testovanie.
- Manažér kvality – robí statické testovanie, overuje výsledky dynamického testovania, robí black box testing, white box testovanie.
- Zákazník – Robí black box testovanie.

4.4.2.2 Procesy definované v dokumente

4.4.2.2.1 Proces: Statické testovanie

1. Určia sa kódy na inšpekciu.
2. Kontrola čitateľnosti a konvencií kódu.
3. Kontrola dodržania špecifikácie. Tento bod zahŕňa skontrolovanie špecifikácie po stránke navrhnutej funkcionality a algoritmickej podstaty.
4. Kontrola korektnosti implementácie vonkajšieho správania funkcionality. Kontrola ošetrenia chybných stavov, dodržania špecifikácia rozhrania funkcionality.

4.4.2.2.2 Proces: Dynamické testovanie

1. Určí sa kód na testovanie.
2. Dekompozícia častí kódu na testovateľné súčasti. Testovateľné súčasti sú časti kódu ktoré majú interpretovateľný výstup (napríklad sa ťažko testu rutina, ktorá zafarbí obrazovku na modro, na toto slúžia priebežné vizuálne testy) a zároveň nie sú závislé od externej entity mimo systému (napríklad je ťažko testovať internetové rozhranie, na to slúžia integračne a výkonnostné testy).
3. Vytvorenie prípadov vstupných údajov a ich výstupných údajov.
4. Formulácia testov. Pričom test je krátka rutina testujúca očakávaný výstup pri daných vstupoch.
5. Analýza výsledkov. Niekedy je výstup správny a chyba je v zlom pochopení algoritmu pri modelovaní výstupov.

4.4.2.2.3 Proces: White-box testovanie

1. Určí sa kód na testovanie.
2. Určia sa jeho prípady použitia.
3. Kompetentná osoba vykoná prípady použitia. Popri vykonávaní testuje rôzne odchýlky od namodelovaných prípadov (kliknutie inam ako je očakávané, zadanie inej hodnoty a i.
4. Vytvorí sa záznam o neočakávanom chovaní funkcionality.

4.4.2.2.4 Proces: Black-box testovanie

1. Určí sa funkcionalita, ktorá bude predmetom testu.
2. Vytvorí sa stručný dokument o možnostiach funkcionality.
3. Vytvoria sa prípady použitia danej funkcionality.
4. Nezaškolená osoba vykoná test pomocou dokumentu z bodu 2. Sledujú sa odchýlky od modelovaných prípadov použitia a chovanie systému na neočakávané ale aj očakávané situácie.
5. Vytvorí sa záznam o neočakávanom chovaní systému.

4.5 Metodika manažmentu vývoja modulov

Autor: Daniel KĺČ

Metodiky pokrýva úkony spojené s udržovaním striktnej modularity a poriadku v systéme. Toto je kontinuálny proces počas vývoja programu.

4.5.1 Nadväzujúce metodiky

- Metodika testovania
- Metodika refaktorizácie

4.5.2 Manažment udržovania striktnej modulárnosti

Na zaručenie znovu použiteľnosti komponentov je potrebné striktne dodržovať špecifikáciu modulov. Táto ja zvyčajne narušená chybami v špecifikácií, či už ide o zle špecifikovanú oblasť, alebo zanedbaný aspekt modulu. V týchto prípadoch udržovanie špecifikácie používa určité procesy, ktoré vzniknutú situáciu vyriešia.

4.5.2.1 Role zahrnuté v spomínaných procesoch

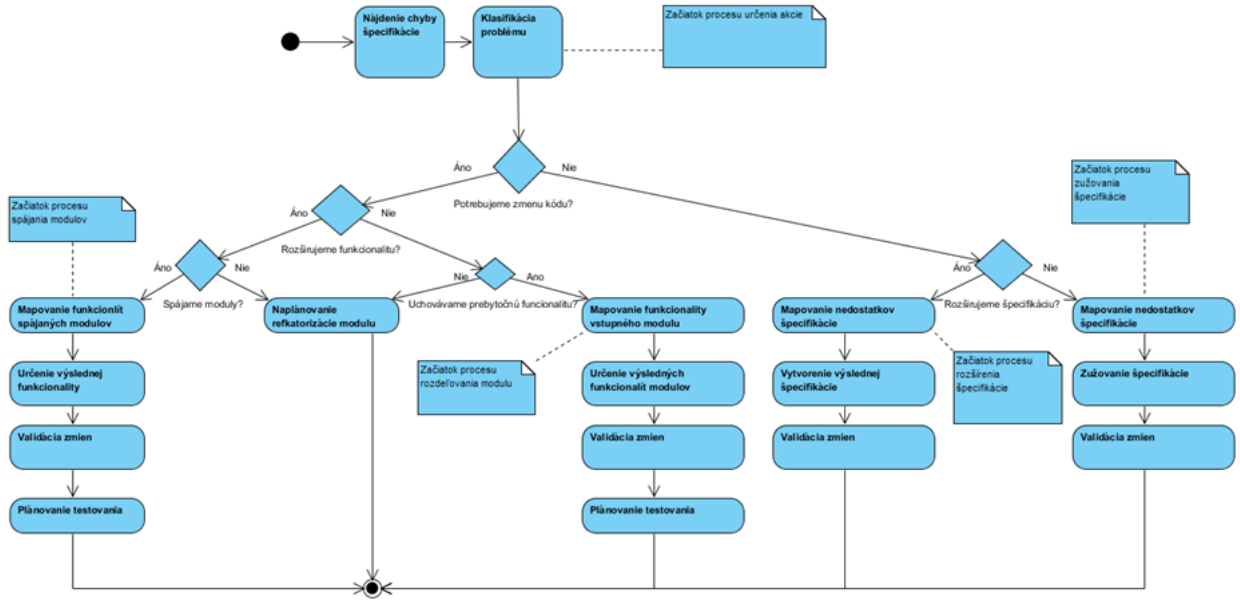
Rola	Zodpovednosť	Proces
Vývojár	Upozornenie na nedodržanie špecifikácie	Proces určenia akcie (5.1)
Vývojár	Fyzická práca s modulom v rámci novej špecifikácie	Proces rozdelenia modulu na viac menších modulov (5.4)
Vývojár	Fyzická práca s modulom v rámci novej špecifikácie	Proces spájania modulov (5.5)
Manažér kvality	Zistenie nedodržania špecifikácie	Proces určenia akcie (5.1)
Manažér kvality	Kategorizácia problému a určenia akcie	Proces určenia akcie (5.1)
Manažér plánovania	Naplánuje refaktorizáciu	Proces určenia akcie (5.1)

Manažér plánovania	Naplánuje testovanie	Proces rozdelenia modulu na viac menších modulov (5.4)
Manažér plánovania	Naplánuje testovanie	Proces spájania modulov (5.5)
Manažér rizík	Odhad dopadu zmeny funkcionality v module	Proces určenia akcie (5.1)
Manažér dokumentácie	Zmení špecifikáciu modulu	Proces rozšírenia špecifikácie modulu (5.2)
Manažér dokumentácie	Zmení špecifikáciu modulu	Proces zúženia špecifikácie modulu (5.3)

4.5.2.2 Procesy definované v oblasti udržovania striktnej modularity

- Proces určenia akcie nad modulom(5.1).
- Proces rozšírenia špecifikácie modulu (5.2).
- Proces zúženia špecifikácie modulu (5.3).
- Proces rozdelenia modulu na viac menších modulov (5.4).
- Proces spájania modulov (5.5).

4.5.2.3 Hierarchia uvedených procesov



Obrázok 1: Obrázok popisuje hierarchiu procesov opísaných v tomto dokumente a ich najdôležitejších akcií. Pomocou uvedeného rozhodovacieho stromu sa v procese určenia akcie nad modulom určí, aká je povaha problému nájdeného v danom module a následne sa určí proces, ktorý daný problém napraví. Je 5 možných akcií ktoré proces môže určiť a to 4 procesy, ktoré sú opísané v tomto dokumente a môže tiež určiť, že je potrebná rekatorizácia s ohraničeným vplyvom len na daný modul , ktorej priebeh ja definovaný v metodike refaktorizácie. Začiatky všetkých piatich procesov popísaných v tomto dokumente sú označené poznámkou v uvedenom diagrame.

4.5.3 Podrobný popis modulov

4.5.3.1 Proces určenia akcie

Proces je vykonaný, ak sa zistí, že modul nezodpovedá špecifikácií.

4.5.3.1.1 Popis procesu

Iniciátor: Vývojár, manažér kvality

Vstup: Modul, popis rozdielu špecifikácie a implementácie

Výstup: Akcia nápravy

Role: Manažér kvality, manažér rizík, manažér plánovania

4.5.3.1.1.1 Postup

1. Kategorizovanie problému

a. Ak modul obsahuje funkcionalitu navyše, kategorizujte prienik oblastí využitia.

b. Ak modul neobsahuje všetku funkcionalitu, aproximujte dopad na prípady použitia modulu.

2. V prípade ak funkcionalita navyše má silný prienik s oblasťou použitia modulu

   a. Aplikujte proces 5.2.

3. V prípade ak funkcionalita navyše má malý prienik s oblasťou použitia modulu

   a. Aplikujte proces 5.4.

4. V prípade ak chýbajúca funkcionalita spôsobuje, že špecifikácia modulu je podobná s iným modulom

   a. Aplikujte proces 5.5.

5. V prípade, že chýbajúca funkcionalita je označená za prebytočnú

   a. Aplikujte proces 5.3.

6. V ostatných prípadoch

   a. Označte modul ako nevyhovujúci.

   b. Vytvorte refaktorizačný zámer a naplánujte refaktorizáciu.

7. Naplánujte určené akcie.

4.5.3.1.2 Podrobný popis krokov

4.5.3.1.2.1 Kategorizovanie problému

Využije sa tabuľka [špecifikovaná funkcionalita X dodaná funkcionalita]. Zoberú sa jednotlivé prípady použitia a vyznačí sa používanie funkcionality. Z tohto je vidieť, či funkcionalita chýba, je šitá na mieru, alebo je prebytočná. Taktiež je tu vidieť, či sa z hľadiska využívania funkcionality nerozdeľujú prípady použitia na skupiny.

4.5.3.1.2.2 V prípade ak funkcionalita navyše má silný prienik s oblasťou použitia modulu

Ak sa na tabuľke z kroku 5.1.2.1 ukáže, že funkcionalita má naozaj opodstatnenie a že sa výrazne neseparuje od zvyšku, aplikujeme proces 5.2.

4.5.3.1.2.3 V prípade ak funkcionalita navyše má malý prienik s oblasťou použitia modulu

Ak sa na tabuľke z kroku 5.1.2.1 ukáže, že funkcionalita má naozaj opodstatnenie ale že sa výrazne separuje od zvyšku, aplikujeme proces 5.4.

4.5.3.1.2.4 V prípade ak chýbajúca funkcionalita spôsobuje, že špecifikácia modulu je podobná s iným modulom

Ak sa na tabuľke z kroku 5.1.2.1 ukáže, že špecifikovaná funkcionalita je naozaj zbytočná a zároveň že funkcionalita modulu robí modul podobný s iným modulom, aplikujeme proces 5.5.

4.5.3.1.2.5 V prípade, že chýbajúca funkcionalita je označená za prebytočnú

Ak sa na tabuľke z kroku 5.1.2.1 ukáže, že špecifikovaná funkcionalita je naozaj zbytočná a zároveň modul zostáva unikátny v dostatočnej miere, tak aplikujeme proces 5.3.

4.5.3.1.2.6 V ostatných prípadoch

Ak sa na tabuľke z kroku 5.1.2.1 ukáže, že špecifikovaná funkcionalita je potrebná a zároveň nie je implementovaná v module alebo nastane ľubovoľný iný problém, vytvorí sa plán refaktorizácie modulu, resp. zoznam akcií, ktoré je treba s modulom vykonať kým bude môcť byť znova otestovaný.

4.5.3.1.2.7 Naplánujte určené akcie

Manažér rizík odhadne riziká realizácie danej akcie a následne manažér plánovania naplánuje vykonanie týchto akcií.

4.5.3.2 Proces rozšírenia špecifikácie modulu

Proces je používaný pri zistení, že modul má viac funkcionality, ako má v špecifikácií a bol určený na rozšírenie špecifikácie procesom 5.1.

4.5.3.2.1 Popis procesu

Iniciátor: Manažér kvality, vývojár

Vstup: Akcia nápravy, pôvodná špecifikácia, zoznam rozdielov

Výstup: Nová špecifikácia

Role: Manažér kvality, manažér dokumentácie

4.5.3.2.1.1 Popis

1. Vykonajte kategorizáciu rozdielov.
2. Určite časti špecifikácie potrebujúce zmenu.
3. Určite korešpondenciu novej verzie špecifikácie s aktuálnou situáciou.
4. Ak nová špecifikácia nie je uspokojujúcom stave, pokračujte krokom 1.
5. Inak označte špecifikáciu za novú špecifikáciu modulu.

4.5.3.2.2 Podrobný popis krokov

4.5.3.2.2.1 Vykonajte kategorizáciu rozdielov

Použije sa tabuľka z 5.1.2.1. Na nej sa vyznačí, ktorá funkcionalita potrebuje byť došpecifikovaná v špecifikácií modulu.

4.5.3.2.2.2 Určite časti špecifikácie potrebujúce zmenu

V existujúcej špecifikácií sa určia podľa zoznamu novej funkcionality časti, ktoré potrebujú byť rozšírené alebo zmenené.

4.5.3.2.2.3 Určite korešpondenciu novej verzie špecifikácie s aktuálnou situáciou

Vykoná sa zmena špecifikácie podľa určenej potreby.

4.5.3.2.2.4 Ak nová špecifikácia nie je uspokojujúcom stave, pokračujte krokom 1

Určí sa pokrytie novej špecifikácie tabuľky z 5.1.2.1. V prípade potreby ďalších úprav sa pokračuje krokom 5.2.2.1.

4.5.3.2.2.5 Inak označte špecifikáciu za novú špecifikáciu modulu

Označte špecifikáciu za vyhovujúcu.

4.5.3.3 Proces zúženia špecifikácie modulu

Proces je používaný pri zistení, že modul má menej funkcionality, ako má v špecifikácií a bol určený na zúženie špecifikácie procesom 5.1.

4.5.3.3.1 Popis procesu

Iniciátor: Manažér kvality, vývojár

Vstup: Akcia nápravy, pôvodná špecifikácia, zoznam rozdielov

Výstup: Nová špecifikácia

Role: Manažér kvality, manažér dokumentácie

4.5.3.3.1.1 Postup

1. Vykonajte kategorizáciu rozdielov.
2. Určite časti špecifikácie potrebujúce zmenu.
3. Určite korešpondenciu novej verzie špecifikácie s aktuálnou situáciou.
4. Ak nová špecifikácia nie je uspokojujúcom stave, pokračujte krokom 1.

5. Inak označte špecifikáciu za novú špecifikáciu modulu.

4.5.3.3.2 Podrobný popis krokov

4.5.3.3.2.1 Vykonajte kategorizáciu rozdielov

Použije sa tabuľka z 5.1.2.1. Na nej sa vyznačí, ktorá funkcionalita potrebuje byť odstránená zo špecifikácií modulu.

4.5.3.3.2.2 Určite časti špecifikácie potrebujúce zmenu

V existujúcej špecifikácií sa určia podľa zoznamu zbytočnej funkcionality časti, ktoré potrebujú byť odstránené alebo zmenené.

4.5.3.3.2.3 Určite korešpondenciu novej verzie špecifikácie s aktuálnou situáciou

Vykoná sa zmena špecifikácie podľa určenej potreby.

4.5.3.3.2.4 Ak nová špecifikácia nie je uspokojujúcom stave, pokračujte krokom 1

Určí sa pokrytie novej špecifikácie tabuľky z 5.1.2.1. V prípade potreby ďalších úprav sa pokračuje krokom 5.2.2.1.

4.5.3.3.2.5 Inak označte špecifikáciu za novú špecifikáciu modulu

Označte špecifikáciu za vyhovujúcu.

4.5.3.4 Proces rozdelenia modulu na viac menších modulov

Proces je používaný pri zistený, že modul má viac funkcionality, ako má v špecifikácií a bol určený na rozdelenia na viac modulov procesom 5.1.

4.5.3.4.1 Popis procesu

Iniciátor: Manažér kvality, vývojár

Vstup: Akcia nápravy, pôvodná špecifikácia, zoznamy navrhovaných funkcionalít výstupných modulov

Výstup: Množina nových modulov a ich špecifikácií

Role: Vývojár, manažér plánovania, manažér kvality

4.5.3.4.1.1 Postup

1. Lokalizujte časti funkcionality v pôvodnom module.
2. Namapujte navrhované funkcionality nových modulov s existujúcou funkcionalitou.
3. Vykonajte rozdelenie funkcionality do nových modulov.

4. Vytvorte schémy závislostí nových modulov.
5. Vytvorte špecifikácie nových modulov.
6. Porovnajte vstupné požiadavky s výstupom.
7. Ak výstup nie je vyhovujúci, pokračujte krokom 3.
8. Naplánujte testovanie.

4.5.3.4.2 Podrobný popis krokov

4.5.3.4.2.1 Lokalizujte časti funkcionality v pôvodnom module

Fyzicky nájdite implementáciu funkcionality v module.

4.5.3.4.2.2 Namapujte navrhované funkcionality nových modulov s existujúcou funkcionalitou

Urobte tabuľku [funkcionalita modulu n X funkcionalita pôvodného modulu] pre všetky navrhované moduly.

4.5.3.4.2.3 Vykonajte rozdelenie funkcionality do nových modulov

Fyzicky preneste prienikovú funkcionalitu z tabuliek z 5.4.2.2.

4.5.3.4.2.4 Vytvorte schémy závislostí nových modulov

Pre každý nový modul vytvorte schému závislostí odvodenú z implementovanej funkcionality.

4.5.3.4.2.5 Vytvorte špecifikácie nových modulov

Na základe schémy závislostí a implementovanej funkcionality vytvorte novú špecifikáciu pre všetky nové moduly.

4.5.3.4.2.6 Porovnajte vstupné požiadavky s výstupom

Na základe tabuľky z 5.4.2.2 porovnajte prenesenú a plánovanú funkcionalitu. Ďalej posúďte špecifikáciu.

4.5.3.4.2.7 Ak výstup nie je vyhovujúci, pokračujte krokom 3

V prípade nevyhovujúcej situácie, pokračujte v 5.4.2.3.

4.5.3.4.2.8 Naplánujte testovanie

Manažér plánovania naplánuje testovanie nových modulov.

4.5.3.5 Proces spájania modulov

Proces je používaný v dvoch prípadoch:

- Pri testovaní sa zistí, že modul obsahuje menej funkcionality ako bolo špecifikované, proces 5.1 určí, že zúžená funkcionalita je v poriadku, ale že nový modul s novou funkcionalitou je podobný inému modulu natoľko, že ich je potrebné zlúčiť kvôli logickej integrite systému.
- Pri testovaní alebo dokumentovaní sa zistí, že špecifikácie dvoch modulov sú podobné natoľko, že ich je potrebné z hľadiska logickej integrity systému zlúčiť.

4.5.3.5.1 Popis procesu

Iniciátor: Vývojár, manažér kvality, manažér dokumentácie

Vstup: Akcia nápravy, špecifikácie danej množiny modulov

Výstup: Implementácia a špecifikácia výstupného modulu

Role: Vývojár, manažér kvality, manažér plánovania

4.5.3.5.1.1 Postup

1. Lokalizujte časti funkcionality pôvodných modulov.
2. Krížovo namapujte funkcionalitu na seba.
3. Určenia časti funkcionality, ktorá sa zachová.
4. Extrakcia funkcionality do jedného modulu.
5. Určenie závislostí nového modulu.
6. Vytvorenie novej špecifikácie.
7. Porovnanie vstupnej a výstupnej funkcionality z hľadiska prípadov použitia.
8. Ak modul nezodpovedá po stránke použiteľnosti pôvodnému stavu, pokračujte krokom 2.
9. Naplánujte testovanie.

4.5.3.5.2 Podrobný popis krokov

4.5.3.5.2.1 Lokalizujte časti funkcionality pôvodných modulov

Fyzicky lokalizujte všetku funkcionalitu vo všetkých moduloch, ktorými sa zaoberáme.

4.5.3.5.2.2 Krížovo namapujte funkcionalitu na seba

Vytvorí sa zoznam funkcionality, kde sa odstránia redundantné implementácie. Ďalej sa vytvorí tabuľka [zoznam funkcionality X zoznam modulov].

4.5.3.5.2.3 Určenie časti funkcionality, ktorá sa zachová

Použijú sa prípady použitia pre spojený výsledný modul na určenie relevantnej funkcionality. Táto sa potom zachová.

4.5.3.5.2.4 Extrakcia funkcionality do jedného modulu

Extrahujú sa relevantné časti funkcionality, ktoré boli označené v bode 5.5.2.3 a ich fyzická poloha bola určená v bode 5.5.2.1.

4.5.3.5.2.5 Určenie závislostí nového modulu

Určia sa vzťahy, ktoré je nutné z pôvodných modulov zachovať na základe tabuľky [zachovaná funkcionalita X pôvodné závislosti modulov] a následnou unifikáciou výslednej množiny závislostí.

4.5.3.5.2.6 Vytvorenie novej špecifikácie

Na základe kompozitnej špecifikácie pôvodných modulov a funkcionality, ktorá bola na výsledný modul použitá sa vytvorí špecifikácia nového modulu.

4.5.3.5.2.7 Porovnanie vstupnej a výstupnej funkcionality z hľadiska prípadov použitia

Na základe prípadov použitia pre pôvodné moduly, sa určí, nakoľko dokáže nový modul uspokojiť potreby, jak po fyzickej stránke, tak po stránke rozhraní.

4.5.3.5.2.8 Ak modul nezodpovedá po stránke použiteľnosti pôvodnému stavu, pokračujte krokom 2

Ak sa ukáže, že nový modul má nedostatky v použiteľnosti v bode 5.5.2.7, tak sa pokračuje bodom 5.5.2.2.

4.5.3.5.2.9 Naplánujte testovanie

Manažér plánovania naplánuje testovanie nového modulu.

4.6 Metodika manažmentu chýb

Autor: Stanislav Kubica

4.6.1 Procesy evidovania a spracovania chyby

4.6.1.1 Pridanie nájdenej chyby

Proces predstavuje kroky vedúce od nájdenia chyby vo vytváranom softvérovom systéme po jej evidenciu v systéme Jira.

Stav procesu chyby po vykonaní procesu	Nová Pridelená
Identifikované role	Autor záznamu o chybe

Postup pridania chyby:

1. Identifikovanie chyby autorom záznamu o novej chybe
2. Vyhľadanie chyby v systéme Jira podľa jej názvu, prípadne identifikátora
3. Ak chyba neexistuje vytvoriť novú (stav „Nová“), inak vložiť doplňujúce informácie k existujúcej (stav sa nemení)

4.6.1.2 Vyhodnotenie chyby

Popisuje kroky, ktoré je potrebné vykonať pred priradením chyby vývojárovi (analytikovi, návrhárovi a pod).

Stav procesu chyby pred vykonaním procesu	Nová
Stav procesu chyby po vykonaní procesu	Zrušená Duplikovaná Pridelená
Identifikované role	Projektový manažér

Postup vyhodnotenia chyby:

1. Projektový manažér si zobrazí nové (nepriradené) chyby v systéme Jira

2. Zanalyzuje, či je chyba relevantná, ak je pokračuje sa krokom 3. V opačnom prípade sa chyba označí ako „Zrušená“

3. Skontroluje či podobná/rovnaká chyba už v systéme neexistuje. Ak existuje priradí ju k existujúcej (čím sa označí ako „Duplikovaná“). V opačnom prípade ju pridelí ako úlohu na vypracovanie niekomu z analytikov, návrhárov alebo vývojárov (podľa povahy úlohy) – úloha prechádza do stavu „Pridelená“

4.6.1.3 Vyriešenie chyby

Popisuje kroky vedúce k vypracovaniu priradenej chyby.

Stav procesu chyby pred vykonaním procesu	Pridelená
Stav chyby počas vykonávania procesu	Vypracovávaná
Stav procesu chyby po vykonaní procesu	Vypracovaná
Identifikované role	Analytik/Návrhár/Vývojár

Postup vyriešenia chyby:

1. Vykonávateľ opravy (Analytik/Návrhár/Vývojár) označí chybu ako „Vypracovávanú“ a začne s jej opravou.

2. Po opravení chyby spíše potrebnú dokumentáciu a označí ju ako „Vypracovanú“

4.6.1.4 Otestovanie vypracovanej chyby

Popisuje kroky, ktoré musí vykonať tester po opravení chyby jej vykonávateľom.

Stav procesu chyby pred vykonaním procesu	Vypracovaná
Stav procesu chyby po vykonaní procesu	Zatvorená Pridelená
Identifikované role	Tester

Postup otestovania opravenej chyby:

1. Tester si zobrazí vypracovanú chybu v systéme Jira
2. Po oboznámení sa s typom chyby, jej výskytom a pod. otestuje danú súčasť systému, aby zistil, či bol daný problém vyriešený. Ak bol problém vyriešený, tak chybu zatvorí. V opačnom prípade ju opäť pridelí pôvodnému vykonávateľovi.

4.6.2

4.6.3 Pridanie nájdenej chyby

Táto časť obsahuje detailný popis pridania nájdenej chyby na najnižšej úrovni. Chyby sú evidované v systéme Jira. Rolu autora záznamu môže zastávať každý člen tímu, ktorý nejakú chybu identifikuje.

4.6.3.1 Identifikovanie chyby autorom nového záznamu o chybe

1. Po nájdení chyby zistiť o akú chybu sa jedná (chyba analýzy, návrhu alebo vývoja) a ktorého podsystému (alebo časti systému) sa týka (napr. frontend, comparator, crawlery a pod.).
2. V prípade chyby týkajúcej sa zdrojového kódu (behu programu) sa pokúsiť o jej detailnejšiu analýzu:
    1. Za akých podmienok chyba nastala?
    2. Ak sa chyba vyskytla na frontend-e, aký operačný systém a webový prehliadač bol použitý
    3. Ak je to možné, získať snímky obrazovky popisujúce chybu
3. Po získaní potrebných informácií pokračovať krokom 2

4.6.3.2 Vyhľadanie chyby v systéme Jira podľa jej názvu, prípadne identifikátora

Vyhľadanie chyby v systéme Jira prebieha nasledovne:

1. Autor chyby sa musí do systému prihlásiť so svojím platným prihlasovacím menom a heslom. Následne uvidí svoju nástenku s informáciami o pridelených úlohách (záleží od individuálnych nastavení)
2. Kliknutím na „Issues“ => „Search“ for issues si autor zobrazí všetky úlohy. Pre pohodlnejšie vyhľadávanie je možné nastaviť filter, aby zobrazoval iba chyby. Ten je možné nastaviť po kliknutí na „Type: All“ a zaškrtnutí možnosti „Bug“. Následne je možné vyhľadávať chyby napísaním jej názvu (prípadne identifikátora) do poľa „Contains text“

4.6.3.3 a) Vytvorenie novej úlohy

V prípade, že sa autorovi chyby nepodarí danú chybu v systéme nájsť, je potrebné vytvoriť záznam o novej chybe a to nasledovne:

1. Autor na ľubovoľnej obrazovke systému Jira klikne na tlačidlo „Create Issue“. Následne sa zobrazí formulár pre vytvorenie nového záznamu

   Formulár je potrebné vyplniť podľa nasledujúceho príkladu:

Chyba: autor našiel chybu v prihlasovaní používateľov

Názov položky formulára	**Hodnota (popis hodnoty)**
Project	Tímový Projekt
Issue Type	Bug
Summary	[FRONTEND] Nefunkčné prihlasovanie používateľov ([identifikator_casti_systemu] Zhrnutie chyby)
Priority	Major (je potrebné vyhodnotiť prioritu chyby)
Due Date	(nechať prázdne)
Assignee	Unassigned (vyplní project manager pri vyhodnocovaní chýb)
Environment	(nechať prázdne)
Description	Po kliknutí na tlačidlo „Prihlásiť“ v okne so zoznamom publikácií určitého vedeckého pracovníka sa zobrazí chyba 500. V iných oknách táto funkcionalita funguje bez problémov OS: Windows 8 Browser: Google Chrome 10 (napísať podrobný opis chyby spolu so všetkými informáciami, ktoré

	má autor k dispozícii)
Original Estimate	(nechať prázdne – vypĺňa project manager)
Remaining Estimate	(nechať prázdne)
Attachment	(vložiť všetky súbory týkajúce sa danej chyby, ktoré má autor k dispozícii) napr. login_screenshot.jpg, mail_od_zakaznika.txt
Labels	Frontend (označenie podsystému, ktorého sa chyba týka)
Epic Link	(nechať prázdne – epicy v Jire nepoužívame)

4.6.3.4 b) Pridanie doplňujúcich informácií k existujúcej úlohe

Ak sa autorovi podarí nájsť existujúcu chybu, ktorá je rovnaká ako identifikovaná, otvorí ju kliknutím na ňu v zozname chýb (ten si zobrazí podľa návodu vyššie). V pravom paneli sa zobrazia informácie o danej chybe.

Ak sa jedná o chybu, ktorá je v jednom zo stavov zrušená alebo zatvorená, je potrebné pred pridaním zmien kliknúť na tlačidlo „Reopen Issue“ s ponechaním pôvodného vykonávateľa. Po znovuotvorení chyby napísať do komentára dôvod znovuotvorenia a kliknutím na tlačidlo „Edit“ zobraziť okno na upravovanie chyby. Do neho autor vloží doplňujúce informácie o chybe podľa 3. a).

Keďže sa editovaná chyba môže nachádzať v rôznych stavoch, je potrebné ponechať polia, ktoré sa nevypĺňajú (podľa tabuľky vyššie), s pôvodnými hodnotami.

4.7 Metodika evidencie úloh

Autor: Rastislav Kostrab

4.7.1 Atribúty problémov a ich hodnoty

4.7.2 Atribúty problému a ich popis

atribút	**vysvetlenie**
Projekt	Rodičovský projekt, do ktorého problém patrí
Kľúč	Unikátny identifikátor, pod ktorým je problém uvedený
Názov	Jednoriadkový popis problému
Typ	Enumerácia (tabuľka č.3)
Stav	Enumerácia (tabuľka č.4)
Priorita	Enumerácia (tabuľka č.5)
Riešenie	Enumerácia (tabuľka č.6)
Ovplyvnené verzie	Verzie projektu, ktoré sú daným riešením ovplyvnené
Opravené verzie	Verzie projektu, ktoré boli alebo budú na základe daného problému opravené
Komponenty	Komponenty projektu, s ktorými daný problém súvisí
Kľúčové slová	Kľúčové slová, ktoré popisujú daný problém
Prostredie	Hardvérové alebo softvérové prostredie, s ktorým daný problém súvisí
Popis	Popis daného problému
Odkazy	Odkazy na súvisiace problémy

Pridelený	Pridelený pracovník pre riešenie daného problému
Reportér	Pracovník, ktorý vytvoril problém
Hlasy	Číslo, ktoré vyjadruje, koľko pracovníkov zahlasovalo za daný problém
Pozorovatelia	Číslo vyjadrujúce, koľko pracovníkov pozoruje riešenie daného problému
Do	Dátum, vyjadrujúci deň, kedy problém má byť vyriešený
Vytvorené	Dátum a čas, kedy bol problém vytvorený
Aktualizované	Dátum a čas, kedy bol problém aktualizovaný
Vyriešené	Dátum a čas, kedy bol problém vyriešený
Časový predpoklad	Predpokladaný čas, ktorý je pridelený na vyriešenie daného problému
Ostáva	Vyjadrenie, koľko času do vyriešenia daného problému ostáva od predpokladaného času riešenia
Vykázané	Výkaz od pracovníka, ktorý na danom probléme pracuje
Vývoj	Ak sa používa Stash alebo BitBucket, tento atribút obsahuje odkaz na vetvu, v ktorej sa daný problém rieši

4.7.2.1 Enumerácie

Nižšie sú v tabuľkách vypísané všetky enumerácie pre vybrané atribúty.

4.7.2.1.1 Enumerácie atribútu typ

Enumerácia	**Typ**
Chyba	Daný problém je chybou v produkte

Vylepšenie	Problém sa venuje zlepšeniu funkcionality
Nová funkcionalita	Problém sa venuje vývoju novej funkcionality
Úloha	Problém je úloha, ktorá sa musí splniť

4.7.2.1.2 Enumerácie atribútu priorita

Enumerácia	**Priorita**
Blokujúca	Najvyššia priorita, zabraňuje plneniu ostatných problémov
Kritická	Ide o prioritu, pri ktorej je potrebné problém urgentne vyriešiť
Majoritná	Závažný problém
Minoritná	Nízka priorita
Triviálny	Najnižšia priorita

4.7.2.1.3 Enumerácie atribútu stav

Enumerácia	**Priorita**
Otvorený	Problém je otvorený a čaká na pridelenie pracovníka, ktorý na ňom bude pracovať
Pracuje sa na tom	Na probléme sa pracuje
Vyriešený	Problém je vyriešený (nie ešte uzavretý)
Znovu otvorený	Problém je znovu otvorený z dôvodu nekompletného riešenia
Uzatvorený	Problém je kompletne vyriešený a uzavretý

4.7.2.1.4 Enumerácie atribútu riešenie

Enumerácia	Riešenie
Vyriešené	Problém bol vyriešený
Nebude vyriešené	Problém nebude vyriešený
Duplikované	Problém je zadaný duplicitne
Nekompletné	Nedostatok informácii pre správane vyriešenie problému

4.7.3 Procesy evidencie úloh

4.7.3.1 Vytvorenie problému

4.7.3.1.1 Postup vytvorenia problému

Nasledujúci postup popisuje kroky, po dosiahnutí ktorých v systéme JIRA vznikne problém:

1. Analyzovanie problému.
2. Skontrolovanie, či už v systéme podobný problém existuje.
3. Ak problém existuje, nevytvára sa ďalší, ale upraví sa nájdený podľa analýzy z bodu 1. Ak problém neexistuje, ide sa na bod 4.
4. Vytvorí sa problém v systéme JIRA

4.7.3.1.2 Stav a riešenie po procese vytvorenia problému

Stav	● Otvorený ● Pracuje sa na tom ● Znovu otvorený
Riešenie	nezadané
Role	● zákazník ● vývojár

	• tester • reportér chyby

4.7.3.2 Vykázanie práce

4.7.3.2.1 Postup vykázania práce

Nasledujúci postup popisuje kroky pri vykázaní práce v rámci daného problému:

1. V systéme JIRA sa nájde problém pre ktorý sa bude vykazovať.
2. V probléme sa vytvorí výkaz práce.
3. Výkaz sa potvrdí a skontroluje, prípadne opraví.

4.7.3.2.2 Stav a riešenie po procese vykázania práce

Stav	• Pracuje sa na tom
Riešenie	nezadané
Role	• Vývojár

4.7.3.3 Vyriešenie problému

4.7.3.3.1 Postup vyriešenia problému

Nasledujúci postup popisuje kroky pri označení problému za vyriešený. Riešenie nemusí však znamenať, že funkcionalita, v rámci ktorej problém vznikol, je opravená.

1. Analyzovanie riešenia.
2. V systéme JIRA sa nájde problém pre ktorý sa bude označovať za vyriešený.
3. Problém sa označí za vyriešený a zvolí sa jeho riešenie.

4.7.3.3.2 Stav a riešenie po procese vyriešenia problému

Stav	• Vyriešený • Uzatvorený
Riešenie	• Vyriešené

	● Nebude vyriešené ● Duplikované ● Nekompletné
Role	● Vývojár ● Projektový manažér

4.7.4 Proces: Vykázanie práce

4.7.4.1 V systéme JIRA sa nájde problém pre ktorý sa bude vykazovať

1. Po ukončení čiastkovej práce (čiastkovej v čase, nie celého problému) v rámci zadaného problému pre daného pracovníka, sa zadá do vyhľadávania problém, na ktorom pracoval. Vyhľadávanie problému je možné cez:

- rýchle hľadanie – „Quick Search“
- hľadanie problému - „Search for Issues“

2. Po zobrazení výsledkov vyhľadávania sa zvolí hľadaný problém. Po jeho zvolení sa problém zobrazuje používateľovi

4.7.4.2 V probléme sa vytvorí výkaz práce

1. Zistia sa potrebné údaje o ukončenej čiastkovej práci:
    - čas, za ktorý sa čiastková práca vykonala,
    - čomu sa čiastková práca venovala,
    - čo sa počas čiastkovej práce vyriešilo.
2. Ak je splnený predošlý čiastkový bod č.2, môže sa pokračovať na bod č. 3.
3. Klikneme na tlačidlo vykazovania - „Log Work“.
4. Po vykonaní bodu č. 3 sa zobrazí okno pre vpisovanie údajov o výkaze práce.
5. Vypíše sa systémovo povinný údaj o čase strávenom počas čiastkovej práci na probléme („Time Spent“) a popis vykonanej práce („Work Description“). Ostatné polia sa nevypisujú.

4.7.4.3 Výkaz sa potvrdí a skontroluje, prípadne opraví

1. Výkaz sa potvrdí tlačidlom potvrdenia - „Log“.

2. Po vykonaní bodu č. 1 sa zobrazí vo výkazoch daný potvrdený výkaz. Skontrolujú sa údaje vpísané do okna vykázania práce. Ak nie je potrebné vo výkaze niečo meniť, proces sa ukončí. V opačnom prípade sa pokračuje na bod č.3.
3. Klikne sa tlačidlo editovania a pokračuje sa bodom 1.

4.8 Metodika dokumentácie zdrojových súborov

4.8.1 Role a ich zodpovednosti

Vo vývojárskom tíme je niekoľko rolí, ktoré majú dočinenia so správnym komentovaním zdrojových kódov. Ich zoznam a popis zodpovedností je uvedený v tabuľke nižšie.

Rola	Zodpovednosť
Manažér kvality	● Zodpovedný za správne okomentovanie zdrojových kódov podľa daných pravidiel ● Kontroluje správnosť zápisu komentárov zdrojových kódov ● Ak zistí nedodržiavanie daného spôsobu komentovania programátorom, je v jeho kompetencii napomenúť ho
Programátor	● Vytvára a komentuje zdrojové kódy

4.8.2 Komentovanie zdrojových kódov

Existencia internej kultúry komentovania zdrojových kódov je dôležité hlavne z dôvodu konzistentnosti zdrojových súborov a znovu použiteľnosti. S touto internou kultúrou musia byť oboznámení všetci programátori, ktorí sa na vývoji podieľajú. Manažér kvality musí mať stanovené možné postihy pre programátorov v prípade nedodržiavania tejto internej kultúry komentovania zdrojových kódov.

4.8.3 JavaDoc

JavaDoc je nástroj, ktorý umožňuje vytvárať programátorskú dokumentáciu k vašim projektom v rovnakom štýle ako je samotná dokumentácia k Jave. Z toho vyplýva, že samotné vytváranie takej dokumentácie sa musí vykonávať cez štandardný nástroj, v našom prípade JavaDoc. Samotný zápis inštrukcií pre JavaDoc, ktoré sa majú spracovať sa umiestňuje do špeciálneho komentáru začínajúceho /** a uzavretého */. Tento zápis sa umiestňuje pred deklaráciu súboru, triedy či metódy.

Názov	Popis
@author	Meno autora
@since	Verzia od ktorej je daný blok dostupný

@exception	Popis výnimky
@param	Popis jednotlivých parametrov predávaných metóde.
@return	Komentár k tomu čo vracia metóda
@see	Popis referencii
@version	Aktuálna verzia

4.8.4 Proces komentovania zdrojových kódov

Samotný proces komentovania zdrojových kódov je možné rozdeliť do viacerých krokov. Po vykonaní prvého kroku (Vytvorenie elementu do ktorého sa bude písať komentár) sa môže programátor rozhodnúť, ktorým krokom bude pokračovať. Môže ako prvý okomentovať súbor, potom triedy a ich metódy ale aj naopak. Všetky komentáre sa píšu v slovenčine bez diakritiky.

Poradie	Názov	Kapitola
1	Vytvorenie elementu do ktorého sa bude písať komentár	5.1
2	Vytvorenie komentáru ku súboru	5.2
3	Vytvorenie komentáru k triede	5.3
4	Vytvorenie komentáru k metóde	5.4

4.8.5 Vytvorenie elementu do ktorého sa bude písať komentár

Pred samotným písaním opisu (komentáru) danej programovej štruktúry je potrebné vytvoriť blok, do ktorého sa bude komentár vypisovať.

Vstup : Neokomentovaný súbor, trieda alebo metóda

Výstup : Okomentovaný súbor, trieda alebo metóda

Zodpovedný : Programátor

- Programátor vytvorí súbor, triedu alebo metódu
- Na začiatok vloží element do ktorého sa bude písať komentár

Eclipse IDE :

- Eclipse automaticky generuje JavaDoc komentár s príslušnou značkou
    - Na začiatku súboru a pred triedami sa generuje autor (názov PC) komentáru
    - Pred metódami sa generuje zoznam parametrov a return
- Napíšte /** + Enter

```
1 package my.pckg;
2 /**
3  *
4  * @author Michal-PC
5  *
6  */
7
8 public class Person {
9     private String name;
10    private String surname;
11    private int age;
```

Obr. Príklad vytvorenia elementu pre súbor/triedu

4.8.6 Vytvorenie komentáru ku súboru

Každý súbor projektu bude zdokumentovaný a bude obsahovať nasledovné položky.

- Stručný popis obsahu súboru, štruktúrovaný do viet, umiestnený pred JavaDoc značkami.
- @author - Ak je autorov súboru viac, každý si pridá svoju značku @author do popisu súboru

Vstup : Existujúci súbor a element do ktorého sa má vpísať komentár pre súbor

Výstup : Vytvorený komentár pre súbor

Zodpovedný : Programátor

Eclipse IDE :

```
package my.pckg;
/**
 * Tento subor obsahuje triedu Person.
 * @author Michal-PC
 * @author Simon-PC
 *
 */

public class Person {
    /**
```

Obr. Príklad vytvorenia komentáru k súboru

4.8.7 Vytvorenie komentáru k triede

Každá trieda bude zdokumentovaná a bude obsahovať nasledovné položky.

- Detailný popis triedy a jej účelu, štruktúrovaný do viet, umiestnený pred JavaDoc značkami.

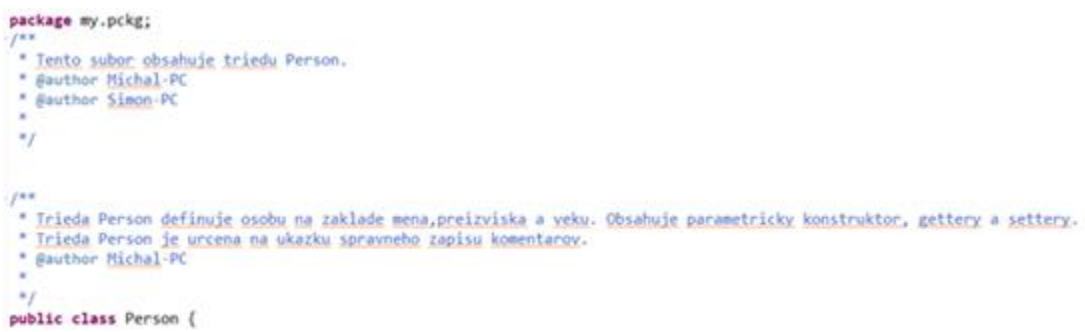
- @author - Ak je autorov súboru viac, každý si pridá svoju značku @author do popisu metódy

Vstup : Existujúca trieda a element pre komentár

Výstup : Vytvorený komentár pre triedu

Zodpovedný : Programátor

Eclipse IDE :



Obr. Príklad vytvorenia komentáru k triede

4.8.8 Vytvorenie komentáru k metóde

Každá metóda bude zdokumentovaná a bude obsahovať nasledovné položky.

- Detailný popis metódy a jej účelu, štruktúrovaný do viet, umiestnený pred JavaDoc značkami.
- @param - Názov parametra a stručný popis k nemu
    - 1..N
- @return - Definovanie návratovej hodnoty a stručný popis k nej
- @see - Metóda, ktorá danú metódu volá alebo metódy a triedy, ktoré majú s metódou súvis a programátor by ich mal vidieť pre dobré porozumenie danej metódy
    - 1..N v tvare : #nazovMetody(Dátové typy parametrov oddelene čiarkou)
    - Príklad : @see *#createRecord(int, String, String)*
- Gettery a settery sa nekomentujú
- Ak je konštruktor preťažený, komentuje sa vždy len jeden (Prvý v zdrojovom kóde)

Vstup : Existujúca metóda a element pre komentár

Výstup : Vytvorený komentár pre metódu

Zodpovedný : Programátor

Eclipse IDE :

```
    }
    /**
     * Metoda, ktora vracia priezvisko velkymi pismenami. Priezvisko je parameter metody.
     * @param surname Priezvisko osoby
     * @return surname - Priezvisko osoby velkymi pismenami
     * @see #print()
     */
    public String getSurnameUpperCase(String surname){
        return surname.toUpperCase();
    }

    public void print(){
        System.out.println(getName()+" "+getSurnameUpperCase(getSurname())+" ["+getAge()+"]");

    }
}
```

Obr. Príklad vytvorenia komentáru k metóde

4.8.9 Generovanie JavaDoc

Po správnom okomentovaní súborov, tried a metód si môže programátor vygenerovať programátorskú dokumentáciu vo forme štruktúry HTML stránok.

Postup :

1. Project -> Generate Javadoc
2. Nastavenie cesty k javadoc.exe, výber projektov a definovanie miesta uloženia.
3. Finish

Generované stránky :

- allclasses-noframe.html - Obdoba predchádzajúceho, pre prehliadače, ktoré nepodporujú framy
- allclasses-frame.html - Zoznam všetkých tried a všetkých balíkov (je to ľavý frame)
- deprecated-list.html - Zoznam deprecated API všetkých balíkov
- help-doc.html - Help pre užívateľov, popis obecnej hierarchie
- index.html - Hlavná stránka dokumentácie
- index-all.html - Defaultný index všetkého (tried, metód...)
- overview-tree.html - Hierarchia tried všetkých balíkov
- package-list.html - Zoznam mien jednotlivých balíkov
- serialized-form.html - Zoznam serialized vo všetkých balíkoch
- stylesheet.css - CSS (definícia fontov a farieb...) pre dokumentáciu

4.9 Metodika verziovania

4.9.1 Použitý nástroj

Na demonštráciu tvorby zmien v centralizovanom systéme je použitý nástroj SourceTree, ktorý je nahraditeľný za iné nástroje (git, nástroje integrované do IDE), ktoré obsahujú rovnakú funkcionalitu ako ponúka nástroj SourceTree.

4.9.2 Role procesu

Rola	Popis
vývojár	● v kontexte tvorby zmien táto rola zahŕňa všetkých používateľov repozitára, ktorý majú prístup k centrálnemu repozitáru s právami čítania a zápisu. ● v procese tvorby zmien je vývojár primárna rola, ktorá proces iniciuje a vo väčšine prípadov tento proces aj terminuje. Výnimkou môže byť riešenie nejednoznačných konfliktov.
správca repozitára	● v kontexte tvorby zmien je táto rola sekundárna a vystupuje tu pri riešení nejednoznačných konfliktov. Konflikty by sa pri správnom rozdelení a naplánovaní úloh nemali stávať, ale aj napriek tomu môže takáto situácia nastať. ● táto rola môže byť nahradená rolou správca modulu, ktorá rieši nejednoznačné konflikty na úrovni modulov, kde každý modul má svojho vlastného správcu.

4.9.3 Prerekvizity a nadväzné procesy

Pre tvorbu zmien nad repozitárom projektu je potrebné, aby pred tvorbou zmien boli vykonané nasledovné kroky:

1. správca repozitáru vytvorí a inicializuje centrálny repozitár
2. každý vývojár si vytvorí lokálnu kópiu centrálneho repozitáru
3. nasleduje proces tvorby zmien a verziovania, ktorý je detailne popísaný v tejto metodike
4. po vytvorení zmien nasleduje proces spájania vytvorenej funkcionality s hlavnou vývojovou vetvou

Procesy 1, 2 a 4 sú hlbšie popísané v dokumente *Manažment projektového repozitáru*.

4.9.4 Proces: verziovanie

Proces tvorby zmien a verziovania je rozdelený do 3 krokov (podprocesov), ktoré sú na seba nadväzné.

4.9.4.1 Aktualizácia repozitára

Popis: pre potreby práce s aktuálnou verziou vyvíjaného softvéru je potrebná pravidelná aktualizácia, ktorá sa musí vykonávať vždy pred začatím tvorby zmien.

Postup:

1. V ľavej časti hlavného okna (zoznam repozitárov) dvojklikom zvolíme repozitár nad ktorým chceme vyvíjať. Ak tento repozitár už máme otvorený môžeme si ho zvoliť v lište s otvorenými repozitármi v pracovnom paneli.

2. Po zvolení repozitáru nasleduje výber typu aktualizácie repozitára

- Nezlučovacia aktualizácia (Fetch)
- Zlučovacia aktualizácia (Pull = Fetch + Merge)

4.9.4.1.1 Nezlučovacia aktualizácia

Použitie: nezlučovacia aktualizácia sa používa, ak je potrebné stiahnuť zmeny z centrálneho repozitára a zároveň tieto zmeny zachovať v tvare akom sú v centrálnom repozitári. Je to presná kópia vetiev s odovzdanými súbormi pričom sú zachované aj lokálne zmeny, čim môžu vzniknúť konfliktné súbory.

Postup:

1. V hornej lište zoznam operácií vývojár zvolí operáciu Fetch.
2. Následne označí oba checkboxy, čo spôsobí aktualizovanie zoznamu repozitárov a zároveň odstránenie lokálnych vetiev, ktoré sú v centrálnom repozitári odstránené.
3. Operácia sa potvrdí pomocou tlačidla OK.

4.9.4.1.2 Zlučovacia aktualizácia

Použitie: zlučovacia aktualizácia sa používa, ak je potrebné stiahnuť zmeny z centrálneho repozitára a zároveň tieto zmeny zlúčiť s vývojovou vetvou.

Postup:

1. V hornej lište zoznam operácií vývojár zvolí operáciu Pull.
2. V rámci projektu sú všetky repozitáre nastavené tak, aby obsahovali práve 1 vzdialený repozitár obvykle nazývaný „origin“. Pokiaľ bude zachovaná táto konvencia bude v zozname vzdialených repozitárov práve 1 možnosť s názvom „origin“, ktorú treba zvoliť.

3. Na rozdiel od nezlučovacej aktualizácie, ktorá aktualizuje všetky zmeny na všetkých vetvách, zlučovacia aktualizácia spája práve 2 vetvy, jednu vzdialenú vetvu z centrálneho repozitára a zvolenú „checkout-nutú“ lokálnu vetvu. Je nutné aby polia názvy týchto polí boli zhodné. Ak by neboli zhodné nešlo by o aktualizáciu, ale o spájanie vývojových vetiev, čo je samostatný proces popísaný v dokumente Manažment projektového repozitáru.

4. V možnostiach je potrebné označiť prvé 3 checkboxy. Prvý checkbox „Commit merged changes immediately“ zaručuje že súbory, ktoré sa podarilo zlúčiť budú odovzdané do centrálneho repozitára. Druhý checkbox „Include messages from commits being merged in merge commit“ spojí informácie o odovzdaniach a pri úspešnej aktualizácií sa tieto informácie zobrazia pri zlučovacom uzle v strome verzií. Tretí checkbox „Create new commit even if fast-forward is possible“ vytvorí samostatné odovzdanie.

4.9.4.2 Zvolenie pracovnej verzie

Popis: každá vykonaná zmena vychádza z určitej verzie, ktorá sa nachádza v istej vývojovej vetve. Preto je potrebné vedieť ako zvoliť verziu z ktorej bude vychádzať nasledovná zmena.

Koncept udržateľného a efekivného repozitára

Obr. Model udržateľného a efektívneho repozitára

Vetva	Popis

master	slúži na uchovávanie produkčných verzií softvéru
hotfix	táto vetva sa využíva v situáciách, kedy je potrebná rýchla oprava chyby produkčnej verzie, ktorá je závažného typu
release	táto vetva sa využíva pri tvorbe novej produkčnej verzie. Hlavným cieľom je zparaleizovať vývoj s tvorbou novej verzie
develop	vetva v ktorej sa má robiť vývoj. Do tejto vetvy by sa mali odovzdávať také súbory, ktoré nevytvárajú novú pridanú funkcionalitu, napr.: opravy chýb, zmeny existujúcich funkcionalít
feature	vetva určená na vývoj nových funkcionalít. Každá funkcionalita bude mať svoju vlastnú feature vetvu

Postup:

1. Výber vetvy do ktorej má byť zmena zaradená.
2. Ak vetva neexistuje (môže sa stať pri vytváraní novej funkcionality), tak vývojár vytvorí vetvu s názvom jej názvom. Výnimkou je vetva funkcionalít (feature) kedy sa je konvenciou nazývať feature-<<názov_funkcionality>>. Napr.: je potrebné vytvoriť nové prihlasovacie okno, tak sa vytvorí vetva s názvom feature-login_page_v2
    a. Ak nová vetva vychádza z momentálnej pracovnej verzie, tak sa zvolí „working copy parent“. Nová vetva bude vychádzať z vetvy, ktorá je práve používaná.
    b. Ak nová vetva vychádza z inej ako poslednej revízie, tak je potrebné zvoliť „specified commit“ a následné pomocou pickera (tlačidlo s označením „...“) sa otvorí okno strom verzí. Tento prístup sa používa, keď je potrebné, aby nová funkcionalita vychádzala z istej produkčnej verzie, a tým sa predišlo obsahu nežiaducich súborov, alebo zmien v pracovnej verzií.
3. Ak vetva existuje, vývojár zvolí poslednú odovzdanú verziu v strome verzí, čím sa vo väčšine prípadov predíde vytváraniu enormného počtu vetiev. Ale môžu nastať prípady, kedy sa viacnásobnému vetveniu nedá predísť, napr.: pri prototypovaní a skúšaní viacerých prístupov na vyriešenie rovnakého problému.
4. Následné si voľbu verzie potvrdíme operáciou checkout zo zoznamu operácií, ktorá nám otvorí nové okno, ktoré je iba informatívne a stačí výber potvrdiť tlačidlom „OK“.

4.9.4.3 Tvorba a odovzdanie zmien

Popis: tvorba zmien je najzakladanejšou funkcionalitou nad repozitárom. Repozitár je integrovaný s ďalšími nástroji pre podporu vývoja (Jira, Crucible + FishEye, Stash), ktoré uľahčujú vývoj, ale aj reportovanie a manažérske oblasti projektu preto je potrebné vytvárať odovzdania správne.

Postup:

1. V aktualizovanom repozitári môže teraz vývojár vytvárať, editovať a vymazávať súbory tak, ako predurčuje dokument Manažment kvality projektu.

2. V pracovnom paneli je umožnené pomocou drag-n-drop, alebo pomocou operácií zobrazenými medzi oknami s lokálnymi zmenami a oknom súborov zaradených do nasledovného odovzdania (commitu). Ako už bolo spomenuté „Staged files“ obsahujú súbory, ktoré budú zaradené do ďalšieho odovzdania a „Working copy files“ je zoznam súborov s lokálnymi zmenami oproti aktualizovanej zvolenej verzií.

3. Zoznam súborov zaradených do nasledovného odovzdania by mal mať formu najmenšej funkčnej jednotky, čo znamená, že zmenené súbory nespôsobia chyboví stav ostatným vývojárom a zároveň obsahuje len tie súbory, ktoré navzájom so sebou súvisia.

4. Odovzdanie súborov zahájime pomocou operácie commit, kde je potrebné správne nazvať odovzdanie. Názov odovzdania by mal vyzerať nasledovne TP-<<číslo_úlohy_v_JIRA>>: <<názov_úlohy>>: <<vykonaná_zmena>>. Napr.: TP-14: Naplnenie DB s dátami z XML Crepč: Hibernate configuration.

5. Checkbox „Create Pull request“ nie je potrebné používať a „Amend latest commit“ je zakázané používať, lebo ovplyvňuje odovzdané súbory bez vytvorenia samostatného odovzdania, čím sa stráca prehľadnosť. Checkbox „Push commits immediatly to“ umožňuje automatické odovzdanie zmien na server okamžite po odovzdaní na lokálnej úrovni. Tu je potrebné vybrať zo zoznamu vzdialených repozitárov práve „origin“.

6. Ak v kroku 5 neboli zmeny odovzdané na server je potrebné tieto zmeny odovzdať čím skôr, ale s ohľadom na funkčnosť projektu pomocou operácie Push. Otvorí sa okno, v ktorom stačí iba potvrdiť odovzdanie.

4.10Metodika prípravy podkladov na stretnutie so zákazníkom

Táto metodika je súčasťou širšieho okruhu manažmentu monitorovania projektu, kde sa nachádzajú rôzne iné metodiky a postupy, ktoré táto oblasť manažmentu pokrýva. Identifikovanými oblasťami sú:

- monitorovanie projektu počas aktuálneho šprintu
- monitorovanie projektu pred pravidelným stretnutím so zákazníkom
- monitorovanie projektu počas pravidelných stretnutí so zákazníkom
- monitorovanie projektu počas plánovania nasledujúceho týždňa/šprintu
- monitorovanie projektu realizované každým členom zvlášť v kontexte svojich priradených úloh
- monitorovanie projektu počas retrospektívy a sledovanie procesu monitorovania
- monitorovanie transparentnosti projektu tak, aby ho bolo možné monitorovať

V tejto metodike sa budeme venovať príprave podkladov na stretnutie so zákazníkom. Je zaradená v širšom kontexte monitorovania stavu projektu pred pravidelným stretnutím so zákazníkom.

4.10.1 Dôležitosť

Táto metodika je základným predpokladom schopnosti odkomunikovať stav projektu zákazníkovi, pretože pomáha utvárať si členom tímu, avšak najmä lídrovi tímu a manažérovi monitorovania projektu, predstavu o tom, v akom stave sa ich projekt nachádza, čo sa podarilo urobiť, či sa to podarilo urobiť tak, aby bol zákazník spokojný, či boli splnené všetky úlohy, ktoré sú pre zákazníka dôležité, a ktoré úlohy a vlastnosti softvéru ešte treba urobiť.

4.10.2 Účastníci

Role zúčastnené v tejto metodike sú:

- líder tímu – zodpovedný za komunikáciu so zákazníkom, za postup v riešení projektu a za splnenie požiadaviek zákazníka
- manažér monitorovania projektu – zodpovedný za kontrolu nad stavom projektu, jeho monitorovanie, musí mať vedomosť o tom, aký je stav projektu a zabezpečovať jeho transparentnosť
- ostatní členovia tímu – zodpovední za vhodné vykazovanie práce na projekte, postupu na riešení úloh

4.10.3 Nástroje

JIRA – nástroj na manažment projektu

Confluence – nástroj na evidovanie výsledkov práce, resp. výstupov čiastkových i úplných

Slack – komunikačný kanál na podporu komunikácie v tíme, je oddelený od ostatných sociálnych kanálov a určený len pre potreby internej komunikácie v tíme.

4.10.4 Rozhrania s inými metodikami

- metodika rozdeľovania úloh
- metodika zadávania úloh do JIRY (aj odhady trvania úloh)
- metodika evidovania čiastkových výsledkov práce do Confluence
- metodika evidencie práce v JIRE

4.10.5 Príprava na stretnutie

Vedúci tímu

- upovedomiť členov tímu, aby si doplnili do JIRY to, čo majú urobené
- vyzvať členov tímu a zozbierať od nich otázky na zákazníka
- spísať splnené a nesplnené úlohy

Povinnosťou vedúceho tímu je, aby bolo na stretnutie všetko pripravené, pretože zodpovedá za celý tím a jeho výsledky. Nasleduje preto tieto kroky:

1. Deň pred stretnutím so zákazníkom v rozmedzí 16. a 20. hodiny pošle všetkým členom tímu správu cez spoločný komunikačný nástroj Slack. V tomto nástroji sú viaceré kanály, z ktorých vyberie #general, aby túto správu dostal každý. Odoslaná správa musí obsahovať pokyn k skontrolovaniu a prípadnému aktualizovaniu záznamu práce, pokyn k zadaniu odrobených hodín práce. Tieto pokyny musia byť v rozkazovacom tvare, aby členovia tímu chápali dôležitosť tejto úlohy. Členovia tímu majú tendenciu na túto činnosť zabúdať, preto nemôže byť vynechaná.
2. Na splnenie týchto úloh nechá členom tímu dostatok času, čo je približne 1 – 2 hodiny.
3. Potom vyzve členov tímu, aby vedúcemu tímu poslali zoznam otázok, ktoré majú na zákazníka a ktoré sa nezodpovedali počas týždňa. Takýmito otázkami sú také, ktoré vznikli maximálne 2 dni pred stretnutím alebo také, ktoré potrebujú dôkladné prerokovanie, pretože sa týkajú zásadného vývoja projektu. Správu formátuje ako vyzvanie, aby nebol na členov tímu vyvíjaný nátlak. Podstatné je opakovane pripomínať, že sa nemusia hanbiť za svoje otázky a zároveň nesmú podceňovať svoje úlohy, aby projekt nezlyhal kvôli nedostatočnej komunikácii.
4. Následne si spíše v bodoch najdôležitejšie pokroky v projekte, ktoré zahŕňajú opravené chyby, vyriešené úlohy označené ako „Blockers“, novo pridané vlastnosti a funkcie. Tieto body bude prezentovať zákazníkovi, aby videl, ako napreduje vývoj projektu.
5. Po bodoch pokroku si spíše vedúci tímu na záchytný zoznam tých vecí, ktoré bude prezentovať zákazníkovi aj úlohy, ktoré neboli splnené. Zákazníkovi prezentujeme aj to, čo nie je tak, ako bolo

dohodnuté, aby sme zachovali transparentnosť projektu a korektnosť prístupu k zákazníkovi. Podklady pre tento krok vytvára manažér monitorovania projektu.

6. Po bodoch pokroku i nesplnených bodoch si pridá do záchytného zoznamu otázky zozbierané od členov tímu.

Výstupom vedúceho tímu v rámci tejto metodiky je záchytný zoznam obsahujúci nasledovné veci rozpísané v bodoch:

- zoznam splnených dôležitých úloh
- zoznam nesplnených úloh
- otázky na zákazníka

Manažér monitorovania projektu

- vyrobiť grafy dokumentujúce stav projektu
- pripraviť filtre na prezentáciu stavu úloh

Manažér monitorovania projektu je v rámci tejto metodiky zodpovedný za prípravu dokumentov potrebných na stretnutí a dokumentov tvoriacich podklady pre tohto manažéra, vedúceho projektu a v konečnom dôsledku i pre každého člena tímu.

Manažér monitorovania projektu musí pripraviť grafy znázorňujúce pokrok na projekte. Najdôležitejší je koláčový graf rozdelenia vyriešených úloh podľa priority. Tento graf nájde manažér v sekcii „Reports“. Najkratšia cesta vedie cez nasledovný link:

- http://arl6.library.sk:8080/browse/TP#selectedTab=com.atlassian.jira.plugin.system.project%3Areports-panel (Obrázok 1)

Tu má na výber mnoho typov grafov a sumárov, z nich vyberie „Pie Chart Report“, následne sa mu zobrazí ponuka, podľa čoho majú byť úlohy projektu rozdelené. V tejto sekcii vyberie voľbu „Priority“. Ďalej sa mu zobrazí samotný graf (Obrázok 2), ktorý je potrebné vytlačiť, pretože nie je možné obmedziť časový rozsah úloh, ktoré majú byť do grafu zahrnuté. Ďalej na vyššie uvedenom linku nájde taktiež sumár úloh zoskupený taktiež podľa priority jednotlivých úloh. K tomuto sumáru sa dostane kliknutím na „Single Level Group By Report“. Následne sú požadované dve veci:

- Filter – kde treba vybrať filter projektu, ktorý sa má zobraziť
- Statistic Type – kde treba vybrať kategóriu, podľa ktorej budú úlohy zoskupené

V časti Filter vyberie manažér filter pre aktuálny projekt a v druhej časti vyberie opäť kategóriu „Priority“, pretože táto kategória najviac zaujíma zákazníka. Ak ešte nie je žiadny filter pre projekt vytvorený, musí ho manažér vytvoriť nasledovným spôsobom:

1. Najprv klikne na horizontálnom menu na záložku „Projects“.
2. Potom vyberie aktuálny projekt v zozname (môže vybrať aj viac, ale takýto scenár nepredpokladáme v rámci Tímového projektu).
3. Takto vytvorený filter uloží pod názvom „Filter_monitorovanie_projektu“.

Vo vytvorenom sumári (Obrázok 3) sú teda zoskupené úlohy podľa priority, pričom pri každej úlohe je uvedený stav úlohy. Pre každú kategóriu je vo vrchnej časti taktiež zobrazené, koľko percent vytvorených úloh tvoria splnené úlohy. To pomáha zákazníkovi, ale aj iným členom tímu urobiť si predstavu o stave projektu, v akom sa nachádza (koľko úloh zostalo nesplnených, koľko úloh sa nezačalo ešte riešiť, koľko úloh už bolo vyriešených...). Ďalej tento sumár slúži vedúcemu tímu ako podklad pre tvorbu svojho záchytného zoznamu.

Tento sumár si pripraví manažér monitorovania projektu pred stretnutím na svojom počítači, aby ho potom mohol ukázať zákazníkovi. Takisto si pripraví aj daný graf, ktorý však musí vytlačiť, pretože na stretnutie musí priniesť tlačenú podobu grafu z minulého stretnutia. Aby sme však predišli komplikáciám s technikou a problémom s tým spojenými, musí manažér mať pripravené v zálohe aj tlačenú podobu týchto podkladov. Následne je na zákazníkovi, či preferuje tlačenú alebo elektronickú verziu. Vedúcemu projektu posiela tieto grafy vyexportované pomocou vstavanej funkcie „Export“ v pravom rohu obrazovky sumárov a grafov.

Každý člen

- skontrolovať aktuálnosť svojich informácií na Confluence
- pozrieť aktuálny stav ostatných úloh
- prichystať otázky na zákazníka a poslať ich vedúcemu

Pretože je každý člen súčasťou tímu, je v rámci zásad Scrum-u potrebné, aby mal prehľad o projekte, aby poznal jeho ciele a víziu projektu. V súlade s tým je povinný priebežne kontrolovať priebeh projektu a činnosti ostatných členov tímu. Vykonáva teda nasledovné činnosti:

- priebežne kontroluje obsah stránok na Confluence, o zmene ktorých dostáva aj notifikácie
- priebežne kontroluje stav úloh priradených k „epic“ úlohe, v rámci ktorej vykonáva prácu na projekte
- venuje sa otázkam, ktoré majú ostatní členovia tímu v snahe pomôcť im

Deň pred stretnutím nezávisle od pokynu vedúceho tímu ku skontrolovaniu obsahu stránok v Confluence skontroluje aktuálnosť obsahu stránky, v rámci ktorej vykonáva prácu na projekte. Drží sa nasledovného postupu:

1. Skontroluje, či stránka obsahuje všetky doposiaľ dosiahnuté výsledky práce.
2. Ak rieši v čase prípravy na stretnutie nejaký problém, ktorý doposiaľ nevyriešil, pridá čiastkové výsledky práce na tomto probléme.

3. Skontroluje, či stránka obsahuje odôvodnenia rozhodnutí učinených v rámci časti projektu, na ktorom pracuje.
4. Skontroluje formát stránky, aby bola prehľadná, čitateľná a informácie podávala stručne, jasne a transparentne.
5. Po ukončení všetkých kontrol, vyzve náhodného člena tímu, aby si prečítal túto stránku a zhodnotil jej vlastnosti, najmä prehľadnosť, zrozumiteľnosť a úplnosť.

Po vyzvaní od vedúceho tímu, aby predložil svoje otázky na zákazníka tak urobí, iba ak by nemal žiadne nejasnosti ohľadne postupu ďalšej práce. V rámci tvorby otázok si kladie tieto otázky, aby nič nepodcenil, ani neprecenil:

- Je moja nejasnosť spôsobená nepochopením zadania alebo nedeterministickým vyjadrením zadania?
    - Túto otázky vyrieši jednoduchá konzultácia s kolegom.
- Je táto otázka zodpovedateľná zákazníkom alebo je skôr technického rázu?
- Je táto otázka riešiteľná kreatívnym prístupom alebo je to otázka potrebujúca zásah zákazníka?

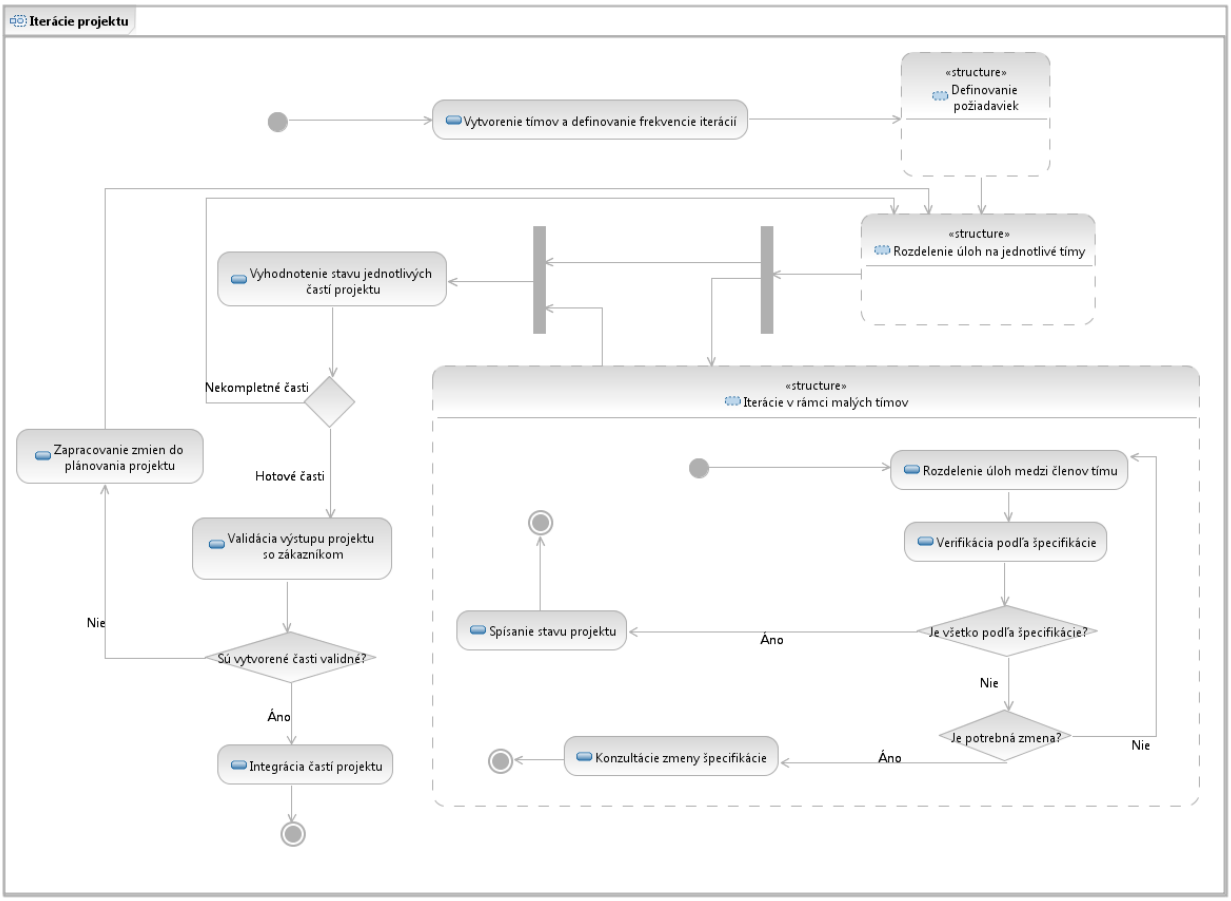
Výstupom prípravy každého člena tímu je kompletná stránka v Confluence, otázky pripravené pre zákazníka a komplexný obraz o stave projektu.

4.11Metodika iterácií projektu

Autor: Tomáš Jánošík

Táto metodika je určená pre veľké firmy na manažovanie procesov spojených s iteráciami projektov. Zahŕňa proces určenia nastavení tímu, ako je počet členov v tíme a počet tímov, opisuje procesy, ktoré by mali prebiehať v tíme, aby bola zabezpečený úspešný vývoj produktu.

Na úvod prikladáme procesný model, ktorý zobrazuje tok riadenia medzi procesmi a následnosť jednotlivých procesov. Opis tohto modelu je uvedený v nasledujúcej časti.



4.11.1 Opis procesného modelu metodiky

Metodika iterácií projektu má ako vstup zazmluvnený projekt, ktorý bol firme pridelený na riešenie. Na začiatku je potrebné definovať veľkosť a zloženie tímov a zároveň aj frekvenciu iterácií, v ktorých sa budú zástupcovia (vedúci) týchto tímov stretávať, aby prerokovali stav projektu a nasledujúci postup v ňom. Následne určení vedúci malých tímov určia pre svoj tím frekvenciu stretávania sa v rámci malého tímu.

Potom nasleduje získavanie a špecifikovanie požiadaviek od zákazníka. Túto časť pokrýva metodika plánovania úloh a metodika zberu požiadaviek. Ďalej nasleduje Rozdelenie úloh medzi jednotlivé tímy, túto časť pokrýva metodika plánovania úloh.

Ďalej nasleduje samotné riešenie projektu. Je rozdelené na iterácie v rámci menších tímov a iterácie celého tímu, ktorý rieši daný projekt.

Iterácie v rámci menších tímov začínajú vždy rozdelením úloh definovaných pre daný tím na jednotlivých členov tímu. Tento proces patrí pod manažment plánovania. Po uplynutí stanoveného času nasleduje verifikácia hotových úloh, či sú výsledky také, ako sa špecifikovalo. Ak výsledky práce zodpovedajú špecifikácii, členovia tímu spíšu svoje výsledky a pripravia ich na prezentáciu projektovému manažérovi a zákazníkovi.

Ak však výsledky špecifikácie nezodpovedajú špecifikácii, musí vedúci tímu rozhodnúť, či je potrebné prepracovať výsledky práce, alebo prepracovať špecifikáciu. V prípade, že je potrebné prepracovať špecifikáciu projektu, túto skutočnosť konzultuje vedúci tímu s manažérom rizík a s projektovým manažérom.

Po uplynutí stanoveného času na iteráciu celého projektu sa stretnú zástupcovia jednotlivých tímov (najlepšie vedúci tímov) a prejdú si stav projektu. V tomto procese zohráva dôležitú úlohu aj manažér monitorovania projektu, ktorý musí mať prehľad o stave projektu, čiže o postupe jednotlivých tímov v projekte a hotových, resp. nekompletných častiach projektu.

Nekompletné úlohy znova prejdú procesom rozdeľovania úloh na tímy, pretože niektoré tímy mohli ukončiť svoju časť práce a mohli by sa zapojiť do riešenia ostatných častí projektu. Tu zohráva dôležitú úlohu manažér plánovania projektu, ktorý musí vedieť, ktoré časti projektu sú dôležité a musí do procesu rozdeľovania zahrnúť aj nové požiadavky od zákazníka.

Hotové úlohy sú následne prezentované zákazníkovi. Tento proces zahŕňa v sebe aj otestovanie výsledkov práce podľa plánu testovania a akceptačných testov špecializovanými testermi, ktorí môžu byť zamestnancami zákazníka. Testovanie je popísané v metodike plánovania úloh. Zákazník po vzhliadnutí a otestovaní hotových častí projektu môže mať pripomienky k týmto častiam, prípadne môže zadefinovať nové požiadavky na zmenu.

Nové a bližšie špecifikované požiadavky na projekt sú zapracované do plánovania úloh a následne opätovne rozplánované na jednotlivé tímy.

Hotové časti projektu akceptované zákazníkom sú integrované do existujúceho systému, prípadne do už vytvorených častí systému (v prípade novo vytváraného systému alebo podsystému). Následne sú tieto integrované časti otestované v testovacej prevádzke a neskôr v ostrej prevádzke.

Rola	Zodpovednosť	Proces

Projektový manažér	Určenie počtu tímov, frekvencie iterácii	Vytvorenie tímov a definovanie frekvencie iterácií projektu
Člen tímu	Skontrolovanie, či sú splnené všetky body požiadaviek k danej úlohe	Verifikácia podľa špecifikácie
Vedúci tímu	Urobenie ešte jednej kontroly požiadaviek, najmä nefunkcionálnych	Verifikácia podľa špecifikácie
Člen tímu	Spísať výsledky kompletne tak, ako sú implementované, pripraviť zdrojové kódy na integráciu	Spísanie stavu projektu
Vedúci tímu	Spísanie akceptačného protokolu, kontrola úplnosti spísaných vecí	Spísanie stavu projektu
Vedúci tímu	Odprezentovanie výsledkov riešenia úloh	Vyhodnotenie stavu jednotlivých častí projektu
Manažér monitorovania projektu	Vytvorenie komplexnej správy zahŕňajúcej všetky splnené a nesplnené úlohy	Vyhodnotenie stavu jednotlivých častí projektu
Projektový manažér	Zhodnotenie stavu projektu a vyvodenie dôsledkov zaň	Vyhodnotenie stavu jednotlivých častí projektu
Projektový manažér	Odprezentovanie výsledkov riešenia úloh zákazníkovi	Validácia výstupu projektu so zákazníkom
Analytik	Získať doplňujúce požiadavky, ktoré má zákazník na výsledky riešenia úloh	Validácia výstupu projektu so zákazníkom

Administrátor	Pripraviť prostredie na integráciu, naplánovať integráciu	Integrácia častí projektu
Členovia tímu	Kontrola integrovaných častí s ohľadom na ich funkčnosť	Integrácia častí projektu

4.11.2 Projektové role

- Manažér projektu – je zodpovedný za celý projekt, za jeho smerovanie a výsledky, vyhodnocuje stav celého projektu, rozhoduje o dôležitých otázkach projektu, ako aj o počte tímov potrebných na splnenie projektu
- Manažér monitorovania projektu – je zodpovedný za monitorovanie stavu projektu, sumarizuje hotové a nevyriešené úlohy do jednotného výpisu, pripravuje podklady pre stretnutie so zákazníkom
- Vedúci tímu – je zodpovedný za tím ľudí, za výsledky úloh im pridelených a motiváciu tímu, kontroluje stav úloh, pokrok v ich riešení, komunikuje s manažérom projektu s ohľadom úloh
- Člen tímu – je zodpovedný za úlohy jemu pridelené, za ich vyriešenie a pripravenie na integráciu
- Analytik – je zodpovedný za získavanie požiadaviek a analýzu systému
- Administrátor – je zodpovedný za integráciu, vykonáva činnosti spojené s integráciou

4.11.3 Opis procesov

Nasleduje opis jednotlivých procesov spolu s vysvetlením ich dôležitosti, vstupmi, výstupmi a scenárom.

4.11.4 Vytvorenie tímov a definovanie frekvencií iterácií

Tento proces je dôležitý pre nastavenie parametrov riešenia projektu. Vytvorenie tímov je základom pre úspešné riešenie projektu, pretože tímy by mali byť zohrané a fungujúce. Ak zostavujeme nový tím, musí prejsť viacerými fázami, počas ktorých sa spoznáva, kým začne fungovať naplno. Ďalej je dôležité rozhodnúť o veľkosti tímu, ktorá by mala byť primeraná. Potom je potrebné zadefinovať ako často sa majú tímy stretávať a ako často sa bude vyhodnocovať riešenie projektu zástupcami všetkých tímov spolu s projektovým manažérom.

Vstupy	Projekt vybraný na riešenie, zadefinovaná ideálna veľkosť tímu
Výstupy	Zadefinované tímy a iterácie v rámci tímu ako aj v rámci všetkých tímov spolu
Role	Projektový manažér

Scenár:

1. Projektový manažér určí náročnosť daného projektu.

2. Podľa zadefinovanej ideálnej veľkosti tímu projektový manažér určí, koľko tímov je potrebných na riešenie projektu, a to nasledovne.

3. Podľa zazmluvnených MD a podľa veľkosti tímu určí, koľko tímov je potrebných a to podľa priemerného počtu MD, ktorý tím s danou veľkosťou dokáže urobiť.

4. Následne určí, či je stanovený počet tímov primeraný. V niektorých prípadoch môže byť žiaduce mať radšej väčšie tímy, napríklad preto, že máme časť projektu, ktorá vyžaduje viac práce ako ostatné a činnosti v rámci tejto časti úzko súvisia.

5. Ku každému tímu priradí vedúceho tímu, ktorý bude zodpovedať za prácu v tíme.

6. Projektový manažér určí, ako dlho budú trvať iterácie v rámci všetkých tímov spoločne. Túto hodnotu nastaví podľa požiadaviek zákazníka ako často chce mať dodávaný prototyp. Každopádne by nemal prekročiť dĺžku jeden mesiac, pretože chceme prototyp dodávať čo najčastejšie. Pri prekročení limitu musí tento krok projektový manažér zdôvodniť v dokumentácii.

7. Vedúci tímu určí, ako často budú prebiehať iterácie v rámci tímu jemu prideleného. V rámci malého tímu by sa mali stretnúť aspoň 2-krát za jednu iteráciu projektu.

4.11.5 Verifikácia podľa špecifikácie

Tento proces sa odohráva po uplynutí „malej iterácie“ v rámci tímu. Začína tesne pred stretnutím, keď je každý člen tímu povinný skontrolovať svoju prácu. Je to preto, aby v každom okamihu vývoja produktu bolo jasné, že jeho vývoj prebieha podľa špecifikácie. A keďže vedúci tímu je zodpovedný za svoj tím a jeho prácu, musí skontrolovať, či práca na projekte prebieha podľa špecifikácie.

Vstupy	Úlohy rozdelené na jednotlivých členov tímu, výsledky práce na úlohách
Výstupy	Dokument popisujúci, či sú úlohy vyriešené podľa špecifikácie
Role	Vedúci tímu, členovia tímu

Scenár:

1. Každý člen tímu sa pozrie znovu na svoje výsledky riešenia a skontroluje ich podľa špecifikácie úloh, ktoré mu boli dodané. Kontroluje hlavne, či sú splnené všetky funkcionálne požiadavky a či je jeho riešenie funkčné. Vypracuje aj dokument, v ktorom sa vyjadrí ku každej funkcionálnej požiadavke či a ako je splnená.

2. Celý tím sa stretne so svojim vedúcim, aby vedúci tímu ešte raz prešiel dokumenty vypracované svojim tímom. Skontroluje postupne s každým členom jednotlivé body, ktoré prekonzultujú.

3. Ak nastane možnosť, že funkcionálna požiadavka nebola splnená, konzultuje vedúci s daným členom tímu dôvod, pre ktorý nebola splnená. Môžu nastať dve možnosti:

    a. nebola splnená kvôli tomu, že ju nebolo možné v daných podmienkach splniť a potrebuje zmenu špecifikácie alebo inú zmenu týkajúcu sa nastavenia projektu a technológii použitých na jeho riešenie; táto zmena je konzultovaná s manažérom rizík a s projektovým manažérom

    b. nebola splnená z iného dôvodu, potom sa pokračuje v riešení danej úlohy dokým nebude zodpovedať špecifikácii

4.11.6 Spísanie stavu projektu

Tento proces nasleduje po verifikácii výsledkov riešenia projektu a má za úlohu pripraviť podklady pre kontrolu v rámci všetkých tímov a pripraviť riešenia úloh na integráciu. Je dôležitý, aby boli zachytené výsledky kontroly vedúceho tímu a aby tieto výsledky kontroly boli spísané jednotne a pospolu. Ďalej pripravuje hotové úlohy na integráciu a testovanie. Na tieto činnosti je potrebné, aby súbory so zdrojovými kódmi boli v určitom formáte a aby testeri mali jasno v tom, ako funguje daná časť projektu.

Vstupy	Dokument popisujúci, či sú úlohy vyriešené podľa špecifikácie, zdrojové kódy riešenia úloh
Výstupy	Dokument zhrňujúci hotové a nedokončené úlohy s obsahujúci stručný popis ich riešenia, zdrojové kódy pripravené na integráciu a testovanie
Role	Vedúci tímu, členovia tímu

Scenár:

1. Každý člen tímu pripraví svoje zdrojové kódy podľa konvencie tak, aby neobsahovali debugové výpisy a nepotrebné časti kódov.

2. Zdrojové kódy umiestni na určené miesto, kde ich administrátor vykonávajúci integráciu ľahko nájde.

3. Ďalej zadefinuje v akom poradí sa majú zdrojové kódy integrovať, pretože niektoré časti kódu budú závisieť od iných (napríklad pri databázach).

4. Člen tímu zadefinuje aj možné vstupy pre svoje riešenie, implementovanú funkcionalitu a očakávané výstupy, aby tester vedel otestovať funkčnosť integrovaných častí.

5. Vedúci tímu zosumarizuje hotové a nedokončené úlohy do zoznamu, z ktorého bude jasné, ktoré časti sú hotové a ktoré nie, a pošle ich manažérovi monitorovania projektu

6. Ku každej riešenej úlohe s pripraví do zoznamu stručný opis riešenia.

7. Taktiež si k zoznamu nedokončených úloh napíše dôvod, pre ktorý neboli dokončené v prípade, že už mali byť hotové.

4.11.7 Vyhodnotenie stavu jednotlivých častí projektu

Tento proces sa odohráva po uplynutí času stanoveného pre „veľkú iteráciu“ platnú pre všetky projekty. Je dôležitý, aby vrcholový manažment mal predstavu o riešení a stave projektu, aby tento stav mohli prezentovať zákazníkovi a dohadovať termíny, prípadne ich posunutie.

Taktiež je to dôležité, aby sa celý tím riešiaci daný projekt dohodol na integrácii určitých častí projektu a na prezentácii hotových častí zákazníkovi, pretože niektoré časti spolu súvisia, a tým pádom by mali byť prezentované spolu.

Vstupy	Dokument popisujúci, či sú úlohy vyriešené podľa špecifikácie, rozdelenie úloh na jednotlivé tímy
Výstupy	Scenár prezentovania prototypu zákazníkovi, rozhodnutia urobené projektovým manažérom, zoznam úloh pripravených na testovanie
Role	Projektový manažér, vedúci tímov, manažér monitorovania projektu

Scenár:

1. Projektový manažér si prejde pripravené dokumenty od každého tímu a s vedúcim tímu (prípadne s povereným členom tímu).

2. Pri nekompletných úlohách si vypočuje zdôvodnenie, prečo nie sú hotové.

3. Na základe vysvetlenia, prečo úlohy nie sú hotové vyvodí dôsledky pre projekt

    a. Poverí manažéra plánovania úloh, aby prepracoval plán riešenia úloh tak, aby tímy neboli zahltené prácou a zároveň bolo vyvinuté maximálne úsilie splniť projekt

    b. Udelí tresty za nesplnené úlohy

    c. Prekonzultuje so zákazníkom posunutie termínu kvôli závažným okolnostiam brániacim vyriešeniu projektu načas.

4. Projektový manažér spíše výsledky kontroly a rozhodnutia, ktoré boli uskutočnené po kontrole.

5. Manažér monitorovania projektu určí, ktoré hotové úlohy sú pripravené na prezentovanie zákazníkovi vzhľadom na nadväznosti jednotlivých úloh na seba.

6. Manažér monitorovania projektu alebo ním poverený človek pripraví scenár prezentovania prototypu zákazníkovi. Tento scenár popisuje funkcionalitu, ktorá bola implementovaná, spôsob vyriešenia úloh spojených s funkcionalitou a postupnosť prezentovania funkcionalít.

7. Vedúci tímov spíšu hotové úlohy pripravené na integráciu a odovzdajú príslušný zoznam administrátorovi a testovaciemu tímu.

4.11.8 Validácia výstupu projektu so zákazníkom

Tento proces sa odohráva u zákazníka a zahŕňa prezentovanie vyriešených častí projektu zákazníkovi, aby stanovil, či je riešenie použiteľné a či zodpovedá jeho predstavám o výsledku. Je dôležité, aby pri vývoji produktu boli časti samotného produktu validované zákazníkom, teda tým, ktorý bude pravdepodobne tento produkt využívať.

Vstupy	Scenár prezentovania prototypu zákazníkovi, otestované hotové časti projektu
Výstupy	Dokumenty obsahujúce poznámky zákazníka k hotovým častiam projektu
Role	Projektový manažér, vedúci tímov, manažér monitorovania projektu

Scenár:

1. Zákazník si pozrie zoznam vyriešených úloh, teda hotových častí projektu spolu s ich otestovanou verziou.

2. Zákazník dostane možnosť otestovať si sám implementovanú funkcionalitu v testovacom prostredí.

3. Projektový manažér asistuje pri testovaní zákazníkom a poskytuje vysvetľujúce informácie na základe dokumentu popisujúceho implementovanú funkcionalitu. Túto asistenciu môže vykonávať pri zložitejších funkcionalitách aj iný určený člen tímu.

4. Po skončení testovania zákazníkom položí projektový manažér (alebo človek ním určený) zákazníkovi doplňujúce otázky, ktorými zistí, ktoré časti projektu neboli dostatočne pochopiteľne spracované alebo ktoré by potrebovali dlhý čas na adaptovanie sa.

5. Projektový manažér (alebo človek ním určený) zapíše všetky pripomienky zákazníka.

6. Projektový manažér spolu so zákazníkom zhodnotia a prerokujú jednotlivé poznámky k riešeniu a implementácii prezentovaných častí projektu.

7. Projektový manažér rozdelí úlohy na tie, ktoré potrebujú upraviť a nie sú v súlade s predstavami zákazníka, a na tie, ktoré sú pripravené na integráciu do existujúceho systému, prípadne do iných hotových častí projektu.

4.11.9 Integrácia častí projektu

Tento proces sa odohráva po tom, ako sú vyriešené úlohy schválené projektovým manažérom, ako aj zákazníkom. Nachádza sa v metodike iterácií projektu preto, že ukončuje iteráciu daných úloh a vytvára ucelený výsledný produkt. Z tohto pohľadu je dôležité zašpecifikovať procesy spojené s integráciou riešení úloh. Tohto procesu sa zúčastňujú aj členovia tímu zodpovední za dané úlohy, aby kontrolovali funkčnosť integrovaných častí.

Vstupy	Zoznam úloh pripravených na testovanie
Výstupy	Protokol o integrovaných úlohách
Role	Administrátor, členovia tímu

Scenár:

1. Administrátor pripraví dokument integrácie, ktorý špecifikuje poradie integrácie riešenia úloh. Zároveň určí čas, kedy dôjde k tejto integrácii.

2. Administrátor pripraví prostredie na integráciu.

3. Postupne integruje riešenia úloh, ktoré mu boli dodané v zozname spolu s ich zdrojovými kódmi.

4. Administrátor zapisuje chyby a výpisy, ktoré nevie sám odstrániť.

5. Po úspešnej integrácii skontrolujú funkčnosť integrovaných úloh a taktiež funkčnosť úlohy v rámci integrovaného prostredia a funkčnosť rozhraní.

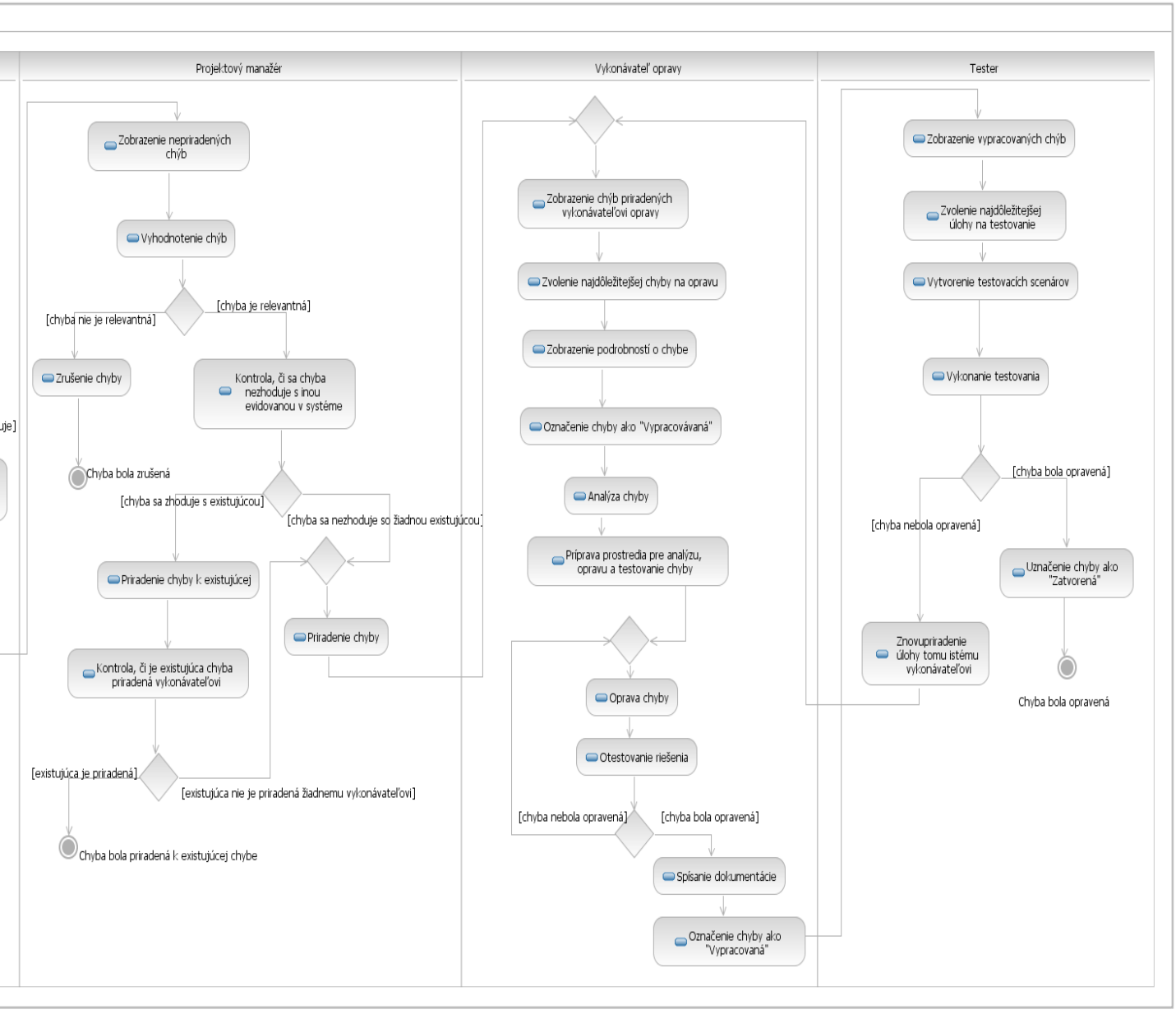
6. Pri zistení chyby alebo narušenej funkcionality zapíšu túto skutočnosť do záznamu integrácie.

7. Členovia tímu okamžite začnú pracovať na odstránení chýb.

4.12 Metodika manažmentu chýb

Autor: Stanislav Kubica

Nasledujúci diagram aktivít znázorňuje priebeh procesov, ktoré menia stav chyby. Znázorňuje tiež role, ktoré dané procesy vykonávajú (alebo sú inak do procesu zahrnuté).


Projektový manažér
Vykonávateľ opravy
Tester
Zobrazenie nepriradených chýb
Vyhodnotenie chýb
[chyba nie je relevantná]
[chyba je relevantná]
Zrušenie chyby
Kontrola, či sa chyba nezhoduje s inou evidovanou v systéme
Chyba bola zrušená
[chyba sa zhoduje s existujúcou]
[chyba sa nezhoduje so žiadnou existujúcou]
Priradenie chyby k existujúcej
Priradenie chyby
Kontrola, či je existujúca chyba priradená vykonávateľovi
[existujúca je priradená]
[existujúca nie je priradená žiadnemu vykonávateľovi]
Chyba bola priradená k existujúcej chybe
Zobrazenie chýb priradených vykonávateľovi opravy
Zvolenie najdôležitejšej chyby na opravu
Zobrazenie podrobností o chybe
Označenie chyby ako "Vypracovávaná"
Analýza chyby
Príprava prostredia pre analýzu, opravu a testovanie chyby
Oprava chyby
Otestovanie riešenia
[chyba nebola opravená]
[chyba bola opravená]
Spísanie dokumentácie
Označenie chyby ako "Vypracovaná"
Zobrazenie vypracovaných chýb
Zvolenie najdôležitejšej úlohy na testovanie
Vytvorenie testovacích scenárov
Vykonanie testovania
[chyba bola opravená]
[chyba nebola opravená]
Uznačenie chyby ako "Zatvorená"
Znovupriradenie úlohy tomu istému vykonávateľovi
Chyba bola opravená

4.12.1 Proces 1: Vytvorenie záznamu o chybe

Proces predstavuje kroky pracovníka technickej podpory vedúce od nájdenia chyby vo vytváranom/udržiavanom softvérovom systéme po jej evidenciu v systéme pre evidenciu chýb.

Stav chyby po vykonaní procesu	Nová
Zodpovedná osoba	Pracovník technickej podpory

Postup pridania chyby:

1. Identifikovanie chyby autorom záznamu o novej chybe – autor záznamu zistí čo najviac doplňujúcich informácií, ktoré budú neskôr užitočné pre realizátora opravy.
2. Vyhľadanie chyby v systéme pre evidenciu chýb – autor zistí, či sa identifikovaná chyba už nachádza v systéme .

Ak bola chyba už zaevidovaná:

a. Vytvorenie nového záznamu o chybe – autor vytvorí nový záznam o chybe v systéme pre evidenciu chýb.

Inak

b. Vloženie doplňujúcich informácií k existujúcej chybe – autor vloží doplňujúce informácie k už existujúcej správe o chybe.

Bližšie podrobnosti o vykonaní tohoto procesu sú zahrnuté v Metodike pre vytvorenie záznamu o chybe.

4.12.2 Proces 2: Vyhodnotenie chyby

Popisuje kroky projektového manažéra, ktoré je potrebné vykonať pred priradením chyby realizátorovi opravy alebo jej zrušením, či priradením k existujúcej chybe.

Stav chyby pred vykonaním procesu	Nová
Stav chyby po vykonaní procesu	Zrušená Duplikovaná Pridelená
Zodpovedná osoba	Projektový manažér

Postup vyhodnotenia chyby

1. Zobrazenie nepriradených chýb – projektový manažér si v systéme pre evidenciu chýb zobrazí všetky nové chyby.
2. Vyhodnotenie chýb – na základe kritérií popísaných v dolnej úrovni metodiky uvedeného procesu vyhodnotenia určí, či je daná chyba relevantná a či je potrebné ju opravovať.

Ak je relevantná:

a. Kontrola, či sa chyba nezhoduje s inou už evidovanou v systéme – projektový manažér skontroluje v systéme pre evidenciu chýb, či sa daná chyba zhoduje s niektorou z existujúcich chýb.

- **Ak sa chyba zhoduje s existujúcou:**

i. Priradenie chyby k existujúcej – projektový manažér priradí chybu k už existujúcej chybe. Ak existujúca nie je priradená, pokračuje krokom Priradenie chyby.

- **Inak:**

ii. Priradenie chyby – projektový manažér na základe kritérií popísaných v dolnej úrovni metodiky uvedeného procesu vyhodnotenia určí vhodného kandidáta na vykonávateľa opravy a priradí mu ju.

- **Inak:**

b. Zrušenie chyby – projektový manažér označí chybu ako zrušenú.

Bližšie podrobnosti o vykonaní tohoto procesu sú zahrnuté v Metodike vyhodnotenie chyby.

4.12.3 Proces 3: Zahájenie práce na oprave chyby

Popisuje kroky vykonávateľa opravy, ktoré sa musia vykonať pred zahájením a počas samotnej opravy uvedenej chyby.

Stav chyby pred vykonaním procesu	Pridelená
Stav chyby po vykonaní procesu	Vypracovávaná
Zodpovedná osoba	Analytik/Návrhár/Vývojár

Postup zahájenia práce na úlohe:

1. Zobrazenie priradených chýb vykonávateľa opravy – vykonávateľ opravy si v systéme pre evidenciu chýb zobrazí zoznam všetkých jemu priradených chýb (v podobe úloh).
2. Zvolenie najdôležitejšej chyby na opravu – zvolenie najdôležitejšej chyby zo zoznamu priradených chýb (tie sú zoradené podľa dôležitosti)
3. Zobrazenie podrobností o chybe – otvorením chyby si zobrazí jej popis a pokračuje ďalším krokom.
4. Označenie chyby ako „Vypracovávaná“ - po otvorení chyby ju označí ako „Vypracovávanú“. V tomto stave sa bude chyba nachádzať až do jej opravenia.
5. Analýza chyby - zobrazí si podrobnosti a doplnkové informácie uvedenej chyby, zadané autorom záznamu o chybe. Na základe týchto informácií vypracuje analýzu pre ďalší postup.
6. Príprava prostredia pre analýzu, opravu a testovanie chyby – na základe predchádzajúcej analýzy záznamu o chybe vytvorí rovnaké podmienky, aké boli v čase výskytu chyby. Tento proces je bližšie opísaný v metodike dolnej úrovne uvedeného procesu.

Bližšie podrobnosti o vykonaní tohoto procesu sú zahrnuté v Metodike zahájenia práce na oprave chyby.

4.12.4 Proces 4: Ukončenie práce na oprave chyby

Popisuje kroky vykonávateľa opravy vedúce k vypracovaniu priradenej chyby zodpovednou osobou.

Stav chyby pred vykonaním procesu	Vypracovávaná
Stav chyby po vykonaní procesu	Vypracovaná
Zodpovedná osoba	Analytik/Návrhár/Vývojár

Postup vyriešenia chyby:

1. Opravenie chyby – vykonávateľ opravy opraví chybu na základe analýzy v procese Zahájenie práce na úlohe
2. Otestovanie riešenia vo vlastnej réžii – na základe scenárov opísaných v podrobnostiach chyby (popis, za akých podmienok chyba vznikla) otestuje opravenú chybu.

- **Ak testovanie preukáže že chyba bola opravená:**

a. Spísanie dokumentácie – vytvorenie dokumentu s popisom chyby, jej príčinou a spôsobom opravy. Ak je to potrebné, upravenie existujúcej projektovej dokumentácie (bližšie popísané v metodike dolnej úrovne daného procesu).
b. Označenie chyby ako „Vypracovaná“ - označenie chyby v systéme pre evidenciu chýb ako „Vypracovaná“ a zapísanie počtu hodín strávených na jej oprave.

- **Inak:**

c. Návrat ku kroku 1. - chyba nie je opravená, je potrebné ju opraviť.

Bližšie podrobnosti o vykonaní tohoto procesu sú zahrnuté v Metodike ukončenia práce na oprave chyby.

4.12.5 Proces 5: Testovanie vypracovanej chyby

Popisuje kroky, ktoré musí vykonať tester po opravení chyby jej vykonávateľom.

Stav chyby pred vykonaním procesu	Vypracovaná
Stav chyby po vykonaní procesu	Zatvorená Pridelená
Zodpovedná osoba	Tester

Postup otestovania opravenej chyby:

1. Zobrazenie vypracovaných chýb v systéme pre evidenciu chýb – tester si v systéme pre evidenciu chýb zobrazí všetky vypracované chyby.
2. Zvolenie najdôležitejšej úlohy na testovanie – na testovanie zvolí najdôležitejšiu úlohu zo zoznamu vypracovaných úloh (tie sú zoradené podľa dôležitosti)
3. Vytvorenie testovacích scenárov – otvorením chyby si zobrazí jej podrobnosti a vypracuje testovacie scenáre (ich tvorba je bližšie popísaná v Metodike tvorby testovacích scenárov).
4. Vykonanie testovania – na základe vytvorených testovacích scenárov vykoná testovanie.

- **Ak testovanie preukáže že chyba bola opravená:**

a. Zatvorenie vyriešenej chyby – označí vypracovanú chybu ako zatvorenú.

- **Inak:**

b. Znovupriradenie neopravenej chyby jej pôvodnému vykonávateľovi – chybu je potrebné opraviť. Tester ju priradí jej pôvodnému vykonávateľovi. Bližšie podrobnosti o vykonaní tohoto procesu sú zahrnuté v Metodike testovania vypracovanej chyby.

4.13 Metodika plánovania úloh

4.13.1 Plánovanie úloh

Pre plánovanie úloh pri vývoji softvérového produktu je dôležitý manažment a definované procesy, ktorými sa manažment riadi. V rámci manažmentu existuje delenie, ktoré je v procesoch plánovania úloh jasne definované.

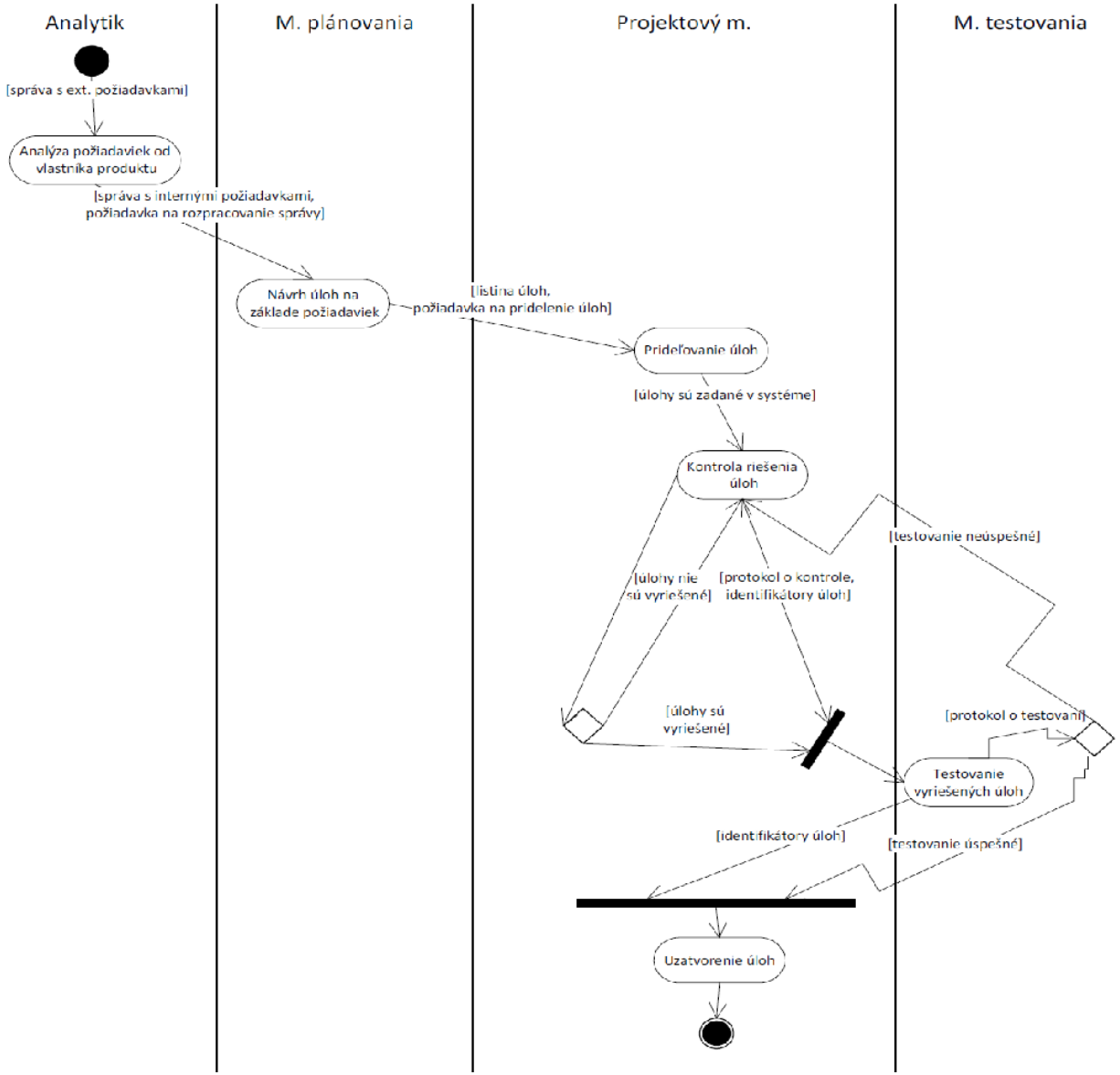
4.13.1.1 Deklarácia procesov

Nasledujúca tabuľka deklaruje procesy v manažmente plánovania, vystupujúce role a zodpovednú rolu, ktoré sú zahrnuté v rámci každého z deklarovaných procesov.

Proces	Zodpovedná rola	Zahrnuté role
Analýza požiadaviek od vlastníka produktu	Analytik	Analytik, manažér komunikácie, projektový manažér, manažér rizík, manažér plánovania
Návrh úloh na základe požiadaviek	Manažér plánovania	Manažér plánovania, analytik,

		manažér rizík
Prideľovanie úloh	Projektový manažér	Projektový manažér, manažér plánovania tím, analytik
Kontrola riešenia úlohy	Projektový manažér	Projektový manažér
Testovanie vyriešenej úlohy	Manažér testovania	Manažér testovania, tester
Uzatvorenie úlohy	Projektový manažér	Projektový manažér

Deklarované procesy so medzi sebou úzko súvisia. Na obrázku nižšie je diagram aktivít, kde sú znázornené všetky relácie medzi procesmi, sekvencia v akej po sebe nasledujú, podmienky, ktoré musia byť v toku aktivít splnené a plavecké dráhy, ktoré definujú zodpovednú rolu pre procesy, ktoré každá z plaveckých dráh obsahuje. Diagram je vytvorený notáciou UML.


Analytik
M. plánovania
Projektový m.
M. testovania
[správa s ext. požiadavkami]
Analýza požiadaviek od vlastníka produktu
[správa s internými požiadavkami, požiadavka na rozpracovanie správy]
Návrh úloh na základe požiadaviek
[listina úloh, požiadavka na pridelenie úloh]
Prideľovanie úloh
[úlohy sú zadané v systéme]
Kontrola riešenia úloh
[testovanie neúspešné]
[úlohy nie sú vyriešené]
[protokol o kontrole, identifikátory úloh]
[úlohy sú vyriešené]
[protokol o testovaní]
Testovanie vyriešených úloh
[identifikátory úloh]
[testovanie úspešné]
Uzatvorenie úloh

4.13.2 Popis procesov

V tejto časti metodiky sú rozpísané všetky procesy deklarované z tabuľky č.1. Každý z procesov má popísaný vstup a výstup. Potom nasleduje stručný popis, scenár a zoznam rolí, ktoré v scenári vystupujú.

4.13.2.1 Analýza požiadaviek od vlastníka produktu

Vstup:

- správa s externými požiadavkami (tzv. „user stories“)

Výstup:

- správa s internými požiadavkami
- požiadavka na rozpracovanie správy

Popis:

Pred tým, ako sa súbor požiadaviek od vlastníka produktu rozdelí na úlohy, ktoré sa zadajú do systému a ktoré bude potrebné implementovať, je potrebné tento súbor požiadaviek skontrolovať na základe istých kritérií (v scenári: bod 2 a bod 3). Ďalej je potrebné súboru požiadaviek identifikovať typ, či ide o hlásenie chýb, vylepšenie funkcionality alebo o novú funkcionalitu. Nakoniec sa overí dĺžka trvania práce na požiadavkách, či je v súlade rámca šprintu, v ktorom požiadavky majú byť implementované a spíšu sa možné riziká, ktoré môžu ovplyvniť cestu k dosiahnutiu splnenia požiadaviek.

Scenár:

1. Analytik, ktorý je zodpovedný za vykonanie tohto procesu, príjme správu s externými požiadavkami od manažéra komunikácie.
2. Vyberie sa zo zoznamu požiadaviek jedna požiadavka a pokračuje bodom 3, ak zoznam požiadaviek je už prázdny pokračuje na bod 7.
3. Projektový manažér skontroluje, či podobná požiadavka už existuje. Ak áno, pokračuje bodom 2.
4. Analytik overí, či je požiadavka splniteľná.
5. Analytik definuje typ požiadavky.
6. Požiadavka sa vloží do správy s internými požiadavkami. Prechádza sa na bod 2.
7. Manažér rizík identifikuje riziká pre splnenie požiadaviek.

8. Manažér plánovania overí dĺžku trvania práce na splnení požiadaviek.
9. Projektový manažér vytvorí požiadavku na spracovanie správy s internými požiadavkami.

Zahrnuté role:

- analytik
- manažér komunikácie
- projektový manažér
- manažér rizík
- manažér plánovania

4.13.2.2 Návrh úloh na základe požiadaviek

Vstup:

- správa s internými požiadavkami
- požiadavka na rozpracovanie správy

Výstup:

- listina úloh
- požiadavka na pridelenie úloh

Popis:

V tomto procese sa správa s internými požiadavkami rozpracuje na úlohy, na základe možného blokovania jednotlivých úloh sa im priradí poradové číslo. Takmer celý tento proces prebieha kolaboratívne.

Scenár:

1. Manažér plánovania, ktorý je zodpovedný za vykonanie tohto procesu, príjme správu s internými požiadavkami.
2. Analytik roztriedi do skupín požiadavky, ktoré so sebou úzko súvisia takým spôsobom, že sa dajú zahrnúť do rámca čiastkovej implementácie.

3. Analytik a manažér plánovania kolaboratívne zoradia úlohy sekvenčne za sebou na základe identifikovaní možných blokovaní počas paralelnej práce na viacerých úlohách.
4. Manažér plánovania spíše úlohy do listiny úloh a vytvorí požiadavku na pridelenie úloh.

Zahrnuté role:

- analytik
- manažér plánovania

4.13.2.3 Prideľovanie úloh

Vstup:

- listina úloh
- požiadavka na pridelenie úloh

Výstup:

- úlohy sú zadané v systéme

Popis:

V tomto procese sa kolaboratívne v tíme pre každú úlohu z listiny úloh určí dĺžka trvania plnenia úlohy, určí sa riešiteľ a zadá sa do systému.

Scenár:

1. Projektový manažér, ktorý je zodpovedný za vykonanie tohto procesu, na základe požiadavky na pridelenie úloh príjme listina úloh.
2. Vyberie sa z listiny úloh jedna z nich a pokračuje bodom 3, ak listina požiadaviek je už prázdna scenár sa ukončí.
3. Analytik, manažér plánovania a tím kolaboratívne identifikujú dĺžku trvania plnenia vybranej úlohy na základe metódy Planning poker.
4. Tím kolaboratívne zvolí medzi sebou zodpovedného riešiteľa za danú úlohu.
5. Projektový manažér danú úlohu vloží do systému.
6. Pokračuje sa bodom 2.

Zahrnuté role:

- projektový manažér
- manažér plánovania
- tím
- analytik

4.13.2.4 Kontrola riešenia úlohy

Vstup:

- úlohy sú zadané v systéme

Výstup:

- protokol o kontrole
- identifikátory úloh

Popis:

Tento proces je určený iba pre projektového manažéra. Ten v ňom kontroluje, či niektoré úlohy sú vypracované. Ak áno, v protokole o kontrole definuje požiadavku na otestovanie čiastkových implementácií, ktorým zodpovedajú určité identifikátory úloh zo systému.

Scenár:

1. Projektový manažér, ktorý je zodpovedný za vykonanie tohto procesu, prezerá stav úloh zadaných v systéme.
2. Projektový manažér vyberie úlohu zadanú v systéme a pokračuje bodom 3, ak vybral už všetky úlohy, pokračuje sa bodom 6.
3. Projektový manažér skontroluje stav úlohy. Ak je v stave, že je vyriešená, pokračuje bodom 4, ináč pokračuje bodom 2.
4. Projektový manažér priradí identifikátor vybranej úlohy medzi identifikátory úloh, ktoré sú výstupom tohto procesu.
5. Projektový manažér pokračuje bodom 2.
6. Projektový manažér vypracuje protokol o kontrole na základe kontroly úloh.

Zahrnuté role:

- projektový manažér

4.13.2.5 Testovanie vyriešenej úlohy

Vstup:

- protokol o kontrole
- identifikátory úloh

Výstup:

- protokol o testovaní
- identifikátory úloh

Popis:

V tomto procese sa otestujú všetky čiastkové implementácie zodpovedajúce identifikátorom úloh zo vstupu.

Scenár:

1. Manažér testovania, ktorý je zodpovedný za vykonanie tohto procesu, príjme protokol o kontrole a identifikátory úloh.
2. Manažér testovania vytvorí protokol testovania, ktorý bude doplnený výsledkami z testov.
3. Manažér testovania vyberie z pomedzi identifikátorov jeden z nich a pokračuje bodom 4, ak vybral už všetky identifikátory, scenár pokračuje bodom 8.
4. Manažér testovania na základe vybraného identifikátora nájde úlohu v systéme.
5. Na základe nájdenej úlohy vytvorí testovací scenár.
6. Tester príjme testovací scenár a podľa pokynov v ňom testuje čiastkovú implementáciu softvérového produktu a testy dokumentuje do protokolu.
7. Pokračuje sa bodom 3.

8. Manažér testovania potvrdí protokol, vloží ho do systému.

Zahrnuté role:

- manažér testovania
- tester

4.13.2.6 Uzatvorenie úlohy

Vstup:

- identifikátory úloh

Výstup:

- uzatvorenie úlohy v systéme

Popis:

V systéme je obmedzenie, ktoré spočíva v tom, že úlohu môže uzatvoriť iba projektový manažér. V tomto procese je popísaný postup pri uzatváraní úlohy v systéme na základe vstupu.

Scenár:

1. Projektový manažér, ktorý je zodpovedný za vykonanie tohto procesu, príjme identifikátory úloh, ktoré sa majú uzatvoriť.
2. Projektový manažér vyberie z pomedzi identifikátorov jeden z nich a pokračuje bodom 3, ak vybral už všetky identifikátory, scenár končí.
3. Projektový manažér na základe vybraného identifikátora úlohy nájde v systéme úlohu.
4. Projektový manažér nájdenú úlohu označí ako uzatvorenú.
5. Pokračuje sa bodom 2.

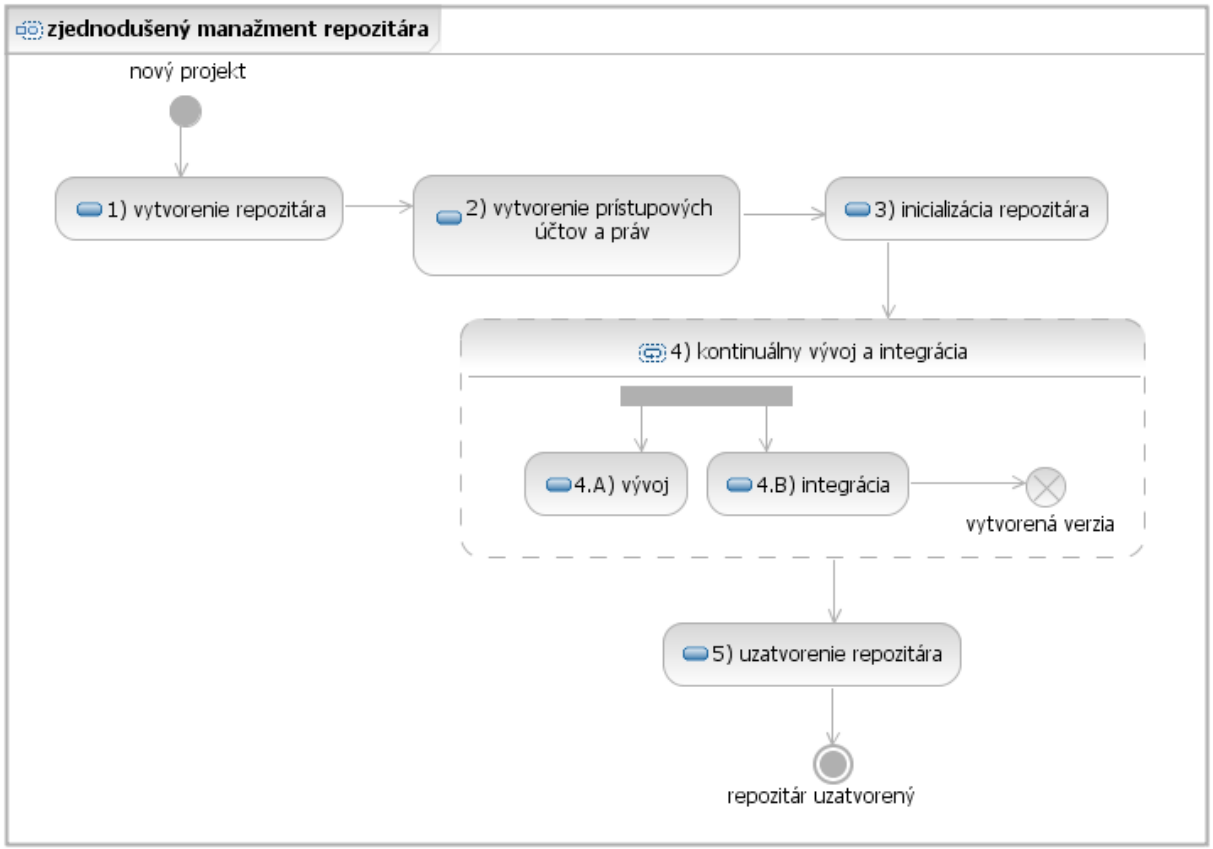
Zahrnuté role:

- projektový manažér

4.14 Metodika manažmentu projektového repozitára a verzií

Metodika je určená pre korporátne projekty na zjednotenie procesov súvisiacich s projektovými repozitármi, verziovaním, integráciou a nasadzovaním systému.

4.14.1 Zjednodušený procesný model



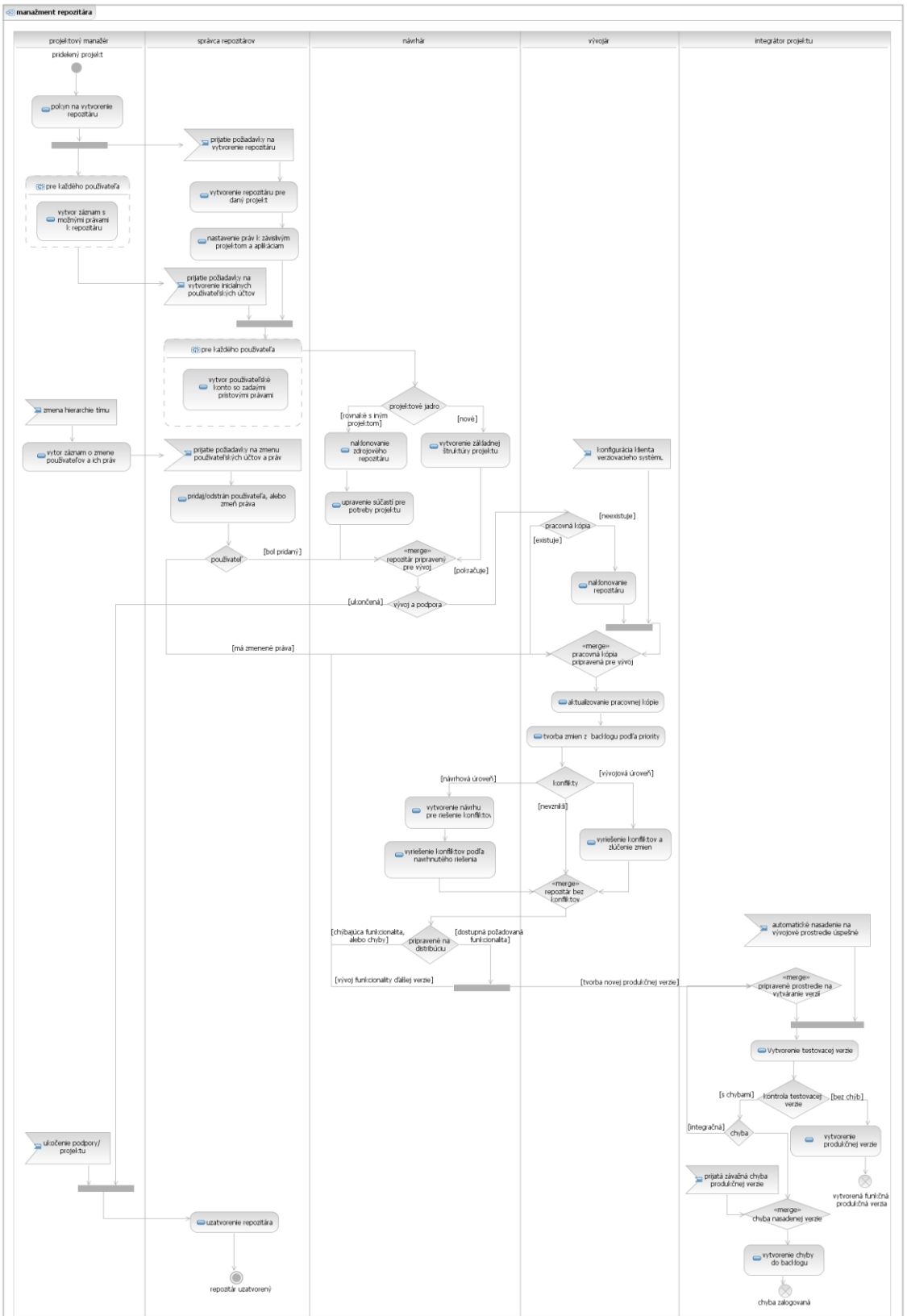
4.14.2 Role a ich zodpovednosti v jednotlivých procesoch

Rola	Proces	Zodpovednosť
Projektový manažér	1) Vytvorenie repozitára	Vydáva pokyn na vytvorenie repozitára
Projektový manažér	2) Vytvorenie prístupových účtov a práv	Vytvára zoznam používateľov a ich práv k repozitára
Projektový manažér	3) Inicializácia repozitára	Vydáva pokyn na inicializovanie základnej štruktúry repozitára
Projektový manažér	4.A) Vývoj	Po dohode s návrhárom iniciuje začiatok vývojovej fázy
Projektový manažér	4.B) Integrácia	Prijíma stav aktuálnej integrácie Konzultuje nový možný termín nasadenia
Projektový manažér	5) Uzatvorenie repozitára	Vydáva pokyn na uzatvorenie repozitára
Správca repozitárov	1) Vytvorenie repozitára	Fyzicky vytvára repozitár
Správca repozitárov	2) Vytvorenie prístupových účtov a práv	Spravuje prístupové účty a práva jednotlivých účtov k repozitáru

Správca repozitárov	3) Inicializácia repozitára	Povoľuje prístup návrhára ako aj samotného repozitára k existujúcim projektovým repozitárom z ktorých sa budú čerpať isté zdroje
Správca repozitárov	5) Uzatvorenie repozitára	Fyzicky uzatvára repozitár na zmeny
Návrhár	3) Inicializácia repozitára	Vytvára prvotný návrh ako aj samotnú štruktúru projektové repozitára a jeho komponentov
Návrhár	4.A) Vývoj	Umožňuje začiatok vývojovej fázy správnou inicializáciou repozitára Rieši návrhové konflikty
Návrhár	4.B) Integrácia	Predpripravuje repozitár na vytváranie novej verzie Umožňuje začiatok tvorby novej verzie
Vývojár	3) Inicializácia repozitára	Môže návrhárovi pomôcť riešiť základnú iniciálnu štruktúru repozitára
Vývojár	4.A) Vývoj	Vyvíja a testuje novú funkcionalitu spôsobom branch per feature, alebo branch per issue (závisí od potrieb projektu) Rieši chyby rôznych úrovní spôsobom branch per bug Rieši konflikty na vývojovej úrovni
Vývojár	4.B) Integrácia	S integrátorom konzultuje a rieši nedorobenie funkcionality do termínu

		S integrátorom konzultuje a rieši chyby, ktoré spôsobil na vývojovom prostredí
Projektový integrátor	4.A) Vývoj	Vytvára funkcionálne aj nefunkcionálne požiadavky na systém s ohľadom na splnenie zákazníkových potrieb
Projektový integrátor	4.B) Integrácia	Vytvára novú verziu na testovacom a produkčnom prostredí Loguje a rieši chyby vývojového, testovacieho a produkčného prostredia

4.


manažment repozitára
projektový manažér
správca repozitárov
návrhár
vývojár
integrátor projektu
pridelený projekt
pokyn na vytvorenie repozitára
prijatie požiadavky na vytvorenie repozitára
pre každého používateľa
vytvor záznam s možnými právami k repozitáru
vytvorenie repozitára pre daný projekt
nastavenie práv k závislým projektom a aplikáciam
prijatie požiadavky na vytvorenie iniciálnych používateľských účtov
pre každého používateľa
vytvor používateľské konto so zadanými prístupovými právami
projektové jadro
[rovnaké s iným projektom]
[nové]
naklonovanie zdrojového repozitára
vytvorenie základnej štruktúry projektu
konfigurácia klienta verzovacieho systému
zmena hierarchie tímu
vytor záznam o zmene používateľov a ich práv
prijatie požiadavky na zmenu používateľských účtov a práv
pridaj/odstráň používateľa, alebo zmeň práva
upravenie súčastí pre potreby projektu
pracovná kópia
[neexistuje]
[existuje]
používateľ
[bol pridaný]
«merge» repozitár pripravený pre vývoj
[pokračuje]
[ukončená]
vývoj a podpora
naklonovanie repozitára
[má zmenené práva]
«merge» pracovná kópia pripravená pre vývoj
aktualizovanie pracovnej kópie
tvorba zmien z backlogu podľa priority
konflikty
[návrhová úroveň]
[vývojová úroveň]
[nevznikli]
vytvorenie návrhu pre riešenie konfliktov
vyriešenie konfliktov podľa navrhnutého riešenia
vyriešenie konfliktov a zlúčenie zmien
«merge» repozitár bez konfliktov
[chýbajúca funkcionalita, alebo chyby]
pripravené na distribúciu
[dostupná požadovaná funkcionalita]
automatické nasadenie na vývojové prostredie úspešné
[vývoj funkcionality ďalšej verzie]
[tvorba novej produkčnej verzie]
«merge» pripravené prostredie na vytváranie verzií
Vytvorenie testovacej verzie
kontrola testovacej verzie
[s chybami]
[bez chýb]
[integračná]
chyba
vytvorenie produkčnej verzie
prijatá závažná chyba produkčnej verzie
vytvorená funkčná produkčná verzia
«merge» chyba nasadenej verzie
vytvorenie chyby do backlogu
chyba zalogovaná
ukončenie podpory/ projektu
uzatvorenie repozitára
repozitár uzatvorený

4.14.4 Proces 1: Vytvorenie repozitára

Časovanie	- Okamžite po získaní projektu - Najneskôr pred začatím vývojovej fázy
Vstup	Požiadavky na prostredia (vývojové, testovacie, produkčné)
Výstup	Vytvorený a nakonfigurovaný repozitár podľa požiadaviek
Iniciačný účastník	Projektový manažér
Zodpovedný účastník	Správca repozitárov
Participujúci účastníci	-
Popis	Pre jednoduchšiu prácu vývojárov a zjednodušenie plánovania a vykonávania integrácie sa vytvára repozitár pre každý projekt. Repozitár sa nakonfiguruje a vytvoria sa práva k prístupu k projektom od, ktorých závisí

4.14.4.1.1 Procesný tok:

1. Po získaní a pridelení projektu, projektový manažér pošle požiadavku správcovi repozitárov o vytvorení repozitára s požadovanou konfiguráciou.
2. Správca repozitárov prijme požiadavku na vytvorenie repozitára.
3. Správca repozitárov vytvorí nový repozitár.
4. Správca repozitárov nakonfiguruje repozitár a nastaví práva k projektom od ktorých je závislý.

4.14.5 Proces 2: Vytvorenie prístupových účtov a práv

Časovanie	- Po vytvorení repozitára - Pred začatím vývoja musí mať návrhár systému vytvorené konto - Prístupové účty vývojárov je najlepšie mať vytvorené tiež pred začiatkom vývojovej fázy - Prístupové účty nových pracovníkov sa vytvárajú počas behu projektu
Vstup	Zoznam pracovníkov s ich právami
Výstup	Vytvorené jednotlivé prístupové účty
Iniciačný účastník	Projektový manažér
Zodpovedný účastník	Správca repozitárov
Participujúci účastníci	Pracovník na projektu
Popis	Z dôvodu bezpečnosti sa nevytvárajú projekty otvorené. Preto treba pre každého pracovníka projektu vytvoriť samostatný účet a prideliť mu príslušné práva. Požiadavku na vytvorenie prístupového účtu zasiela projektový manažér. Vo výnimočných prípadoch môže požiadavku poslať aj samotný pracovník, ale táto požiadavka musí byť dodatočne schválená.

4.14.5.1.1 Procesný tok:

1. Správca repozitárov prijme zoznam pracovníkov s prístupom k repozitáru s ich možnými právami.
2. Správca repozitára vytvorí prístupové konto pre každého pracovníka zo zoznamu.
3. Správca repozitára pridelí požadované práva zo zoznamu jednotlivým prístupovým účtom.

4.14.6 Proces 3: Inicializácia repozitára

Časovanie	- Po vytvorení prístupových práv (aspoň pre návrhára) - Po vytvorení základného návrhu riešenia
Vstup	Požiadavka na inicializovanie základnej štruktúry repozitára
Výstup	Repozitár pripravený na kontinuálny vývoj
Iniciačný účastník	Projektový manažér
Zodpovedný účastník	Návrhár
Participujúci účastníci	Správca repozitára Vývojár
Popis	Projektový manažér pošle požiadavku na návrhára systému. Návrhár podľa analýzy uváži akým spôsobom inicializuje repozitár.

4.14.6.1.1 Procesný tok:

1. Návrhár prijme požiadavku na vytvorenie iniciálnej štruktúry repozitára od projektového manažéra.
2. Návrhár analyzuje iné projekty na možné prepoužitie zdrojov.

a. Ak je možné prepoužiť isté zdroje z iných projektov.

i. Návrhár naklonuje zdrojový repozitár do projektového repozitára.

ii. Návrhár upraví repozitár pre potreby projekta.

b. Ak nie je možné prepoužiť zdroje z iných projektov.

i. Návrhár vytvorí základnú štruktúru repozitára.

3. Návrhár oznámi projektovému manažérovi úspešné inicializovanie repozitára.

4.14.7 Proces 4: Kontinuálny vývoj a integrácia

Kontinuálny vývoj a integrácia je kontinuálny proces, ktorý je najdôležitejší pre fázy vývoja a testovania a asi aj najdôležitejší z pohľadu prínosu k ukončeniu projektu. V rámci týchto fáz by mal produkt nadobudnúť formu a funkcionalitu požadovanú zákazníkom, preto je potrebné, aby spomenuté podprocesy prebiehali paralelne a neblokujúco.

Pod vývojom sa v kontexte manažmentu repozitára a verzií rozumie akákoľvek aktivita, ktorá spôsobuje pridávanie, zmenu, alebo odstraňovanie súborov obsiahnutých v repozitári. Preto sa tu nebude rozlišovať oprava chýb a vývoj nových súčastí.

4.14.8 Podproces 4.A: Vývoj

Časovanie	- Začiatok fázy vývoja - Môže začať aj skôr ako istý vývoj v predstihu
Vstup	- Funkčný repozitár inicializovaný do požadovanej štruktúry - Vytvorené prístupové kontá pre vývojárov - Vytvorený backlog

Výstup	Kontinuálne funkčná jednotka možná nasadenia
Iniciačný účastník	Návrhár Projektový manažér
Zodpovedný účastník	Vývojár
Participujúci účastníci	Návrhár Projektový integrátor
Popis	Začiatok tohto procesu je možný po dohode projektového manažéra a návrhára, aby sa predišlo zapracovávaniu zmien do nesprávne iniciovaného repozitára, ale s ohľadom na dohodnutý termín začiatku. V tejto fáze sa vytvárajú všetky požiadavky na systém. Tento proces sa ukončuje rovnako ako aj samotný repozitár, teda po ukončení podpory, alebo samotného projektu. Reálny vývoj sa začína, keď si každý člen vývojového tímu naklonuje projektový repozitár a je schopný vykonávať zmeny nad ním.

4.14.8.1.1 Procesný tok:

4.14.8.1.1.1 Označené rímskymi číslicami sa vykonávajú len pred prvým použitím projektového repozitára.

I. Každý člen vývojového tímu si nakonfiguruje klienta pre používanie distribuovaného systému verzií.

II. Každý člen vývojového tímu si naklonuje aktuálnu verziu repozitára.

1. Každý člen si pre vykonávaním zmien aktualizuje pracovnú kópiu repozitára.
2. Pri tvorbe zmien vývojár skontroluje projektový backlog a zvolí si jednu z prioritných úloh.
3. Vývojár vykoná zmeny a priebežne ich odovzdáva do repozitára.

a. Ak v repozitári nevznikli konflikty.
b. Ak v repozitári vznikli konflikty na vývojárskej úrovni.
i. Vývojár vyrieši konflikty.
ii. Vývojár zlúči zmeny.
c. Ak v repozitári vznikli konflikty na návrhárskej.
i. Návrhár vytvorí návrh riešenia konfliktov.
ii. Návrhár vyrieši konflikty podľa navrhnutého riešenia.
iii. Návrhár zlúči zmeny.
4. Vývojár označí úlohu za dokončenú.
5. Vývojár pokračuje vo vývoji na ďalších úlohách z backlogu krokom 1.

4.14.9 Podproces 4.B: Integrácia

Časovanie	- Pred nasadením systému do prostredia podľa dohodnutých, alebo interných míľnikov (opakujúca sa aktivita)
Vstup	- Repozitár obsahujúci požadovanú funkcionalitu - Úspešné automatické nasadenie na vývojové prostredie - Požiadavka na vytvorenie novej verzie (explicitná, alebo z harmonogramu)
Výstup	Funkčný systém nasadení do prostredia
Iniciačný účastník	Projektový integrátor
Zodpovedný účastník	Projektový integrátor

Participujúci účastníci	Vývojár
Popis	Každé prostredie má svoje špecifikácie a systémy s ktorými treba vyvíjaný systém integrovať, preto je potrebné mať viac prostredí (vývojové, testovacie, produkčné), ktorými sa zabezpečí, že k zákazníkovi sa vždy dostaví produkt na čas a bez chýb. Integrácia prebieha paralelne s vývojom a jej cieľ je nasadiť vyvíjaný systém do produkčného prostredia.

4.14.9.1.1 Procesný tok:

1. Integrátor prijme požiadavka, alebo má naplánovanú tvorbu novej produkčnej verzie.
2. Integrátor skontroluje, či sú všetky požadované súčasti verzie v stave na nasadenie.

a. Ak repozitár neobsahuje všetky požadované funkcionality, alebo automatické nasadenie a testovanie bolo neúspešné.

i. Integrátor konzultuje s vývojárom, ktorý spôsobil meškanie náročnosť opravy.

a. Ak chybovú časť je možné urýchlene opraviť.

i. Integrátor počká na ukončenie zmien vývojárom

b. Ak chybu nie je možné urýchlene opraviť.

i. Integrátor ohlási projektovému manažérovi meškanie integrácie.

ii. Integrátor po konzultácií s vývojármi naplánuje nový termín integrácie.

iii. Procesný tok končí.

b. Ak repozitár obsahuje všetky požadované funkcionality a automatické

i. Integrátor vytvorí a nasadí systém na testovacie prostredie.

a. Ak nasadenie prešlo bez chýb a testovacie prostredie je funkčné.

i. Integrátor nasadí systém na produkčné prostredie.

ii. Integrátor ohlási projektovému manažérovi meškanie integrácie.

b. Ak nasadenie na testovacie prostredie neprešlo, alebo bola prijatá závažná chyba na produkčnom prostredí.

i. Integrátor vloží do projektového backlogu nový chybový záznam s najvyššou prioritou riešenia.

ii. Integrátor ohlási stav projektovému manažérovi.

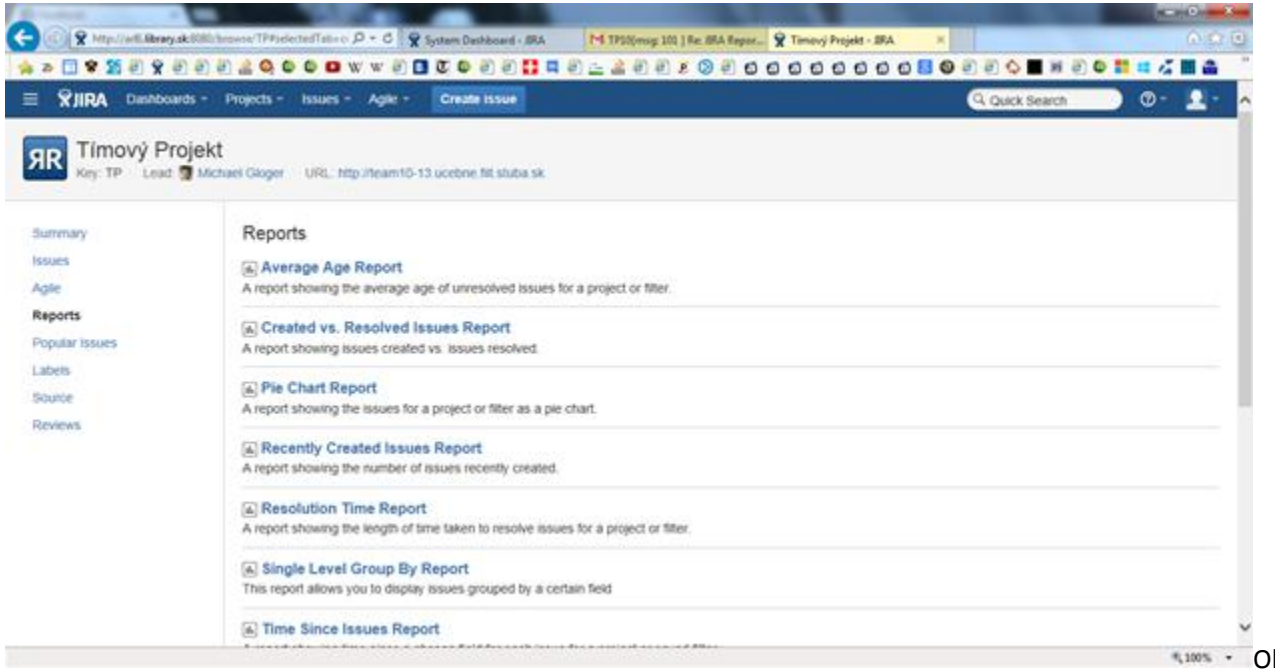
4.14.10 Proces 5: Uzatvorenie repozitára

Časovanie	Po skončení projektu, alebo ukončení podpory
Vstup	Požiadavka na uzatvorenie repozitára
Výstup	Repozitár uzatvorený na zmeny
Iniciačný účastník	Projektový manažér
Zodpovedný účastník	Správca repozitára
Participujúci účastníci	-
Popis	Po ukončení projektu bez dohodnutej podpory, alebo ukončení podpory v rámci zmluvy sa repozitár nevymazáva, lebo jeho súčasti, alebo celý projekt môže byť využiteľný. Preto sa repozitáre nikdy nevymazávajú, iba sa uzatvoria na operáciu zmeny.

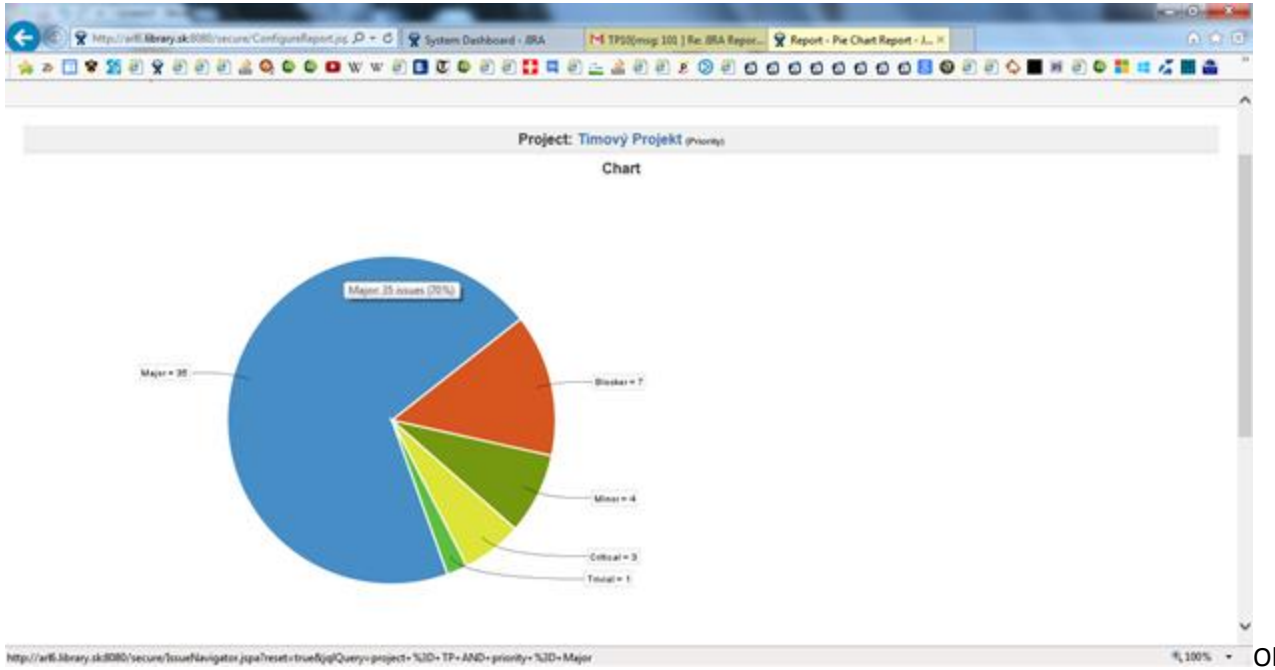
4.14.10.1.1 Procesný tok:

1. Projektový manažér pošle požiadavku na uzatvorenie repozitára.
2. Správca repozitára po prijatí požiadavky uzavrie repozitár na zmenu

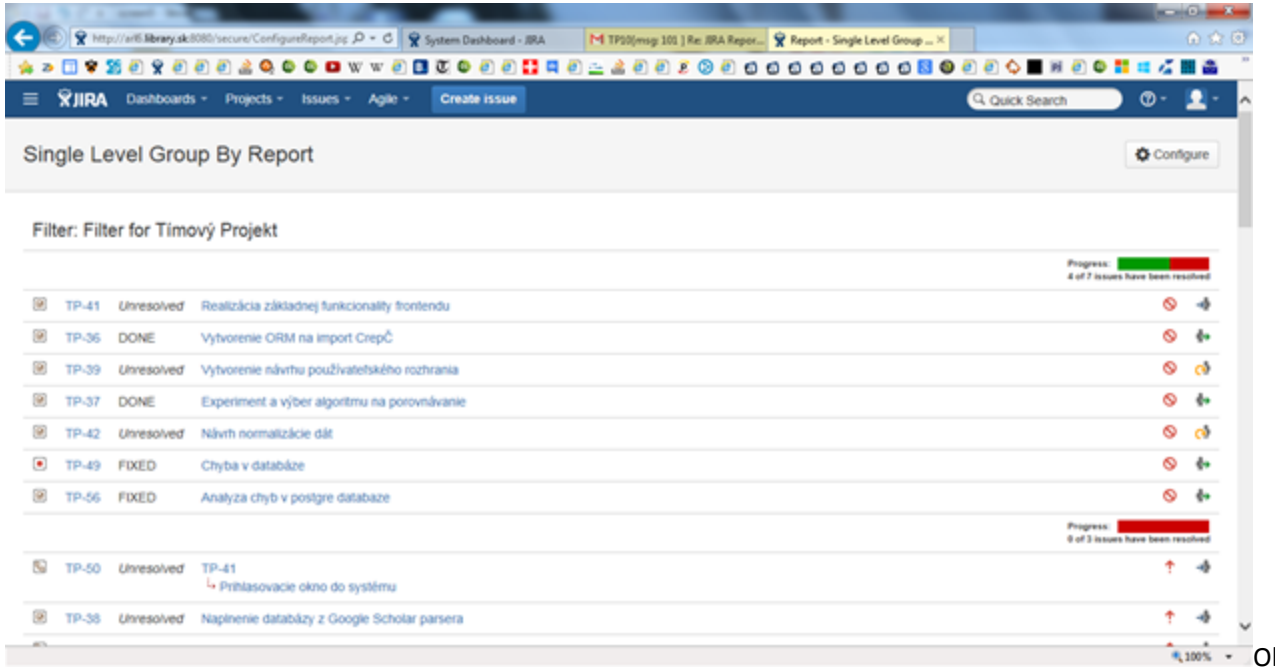
5 Prílohy



Obr. - Výber sumárov a grafov



Obr. - Koláčový graf



Obr. - Sumár úloh podľa priority