SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológii

Dokumentácia k riadeniu projektu

Textový editor obohatený o grafické prvky

Tímový projekt

Vypracoval: Innovators – tím č.10
Akademický rok: 2011/12

Autor:	Innovators – tím č.10
Téma projektu:	textový editor obohatený o grafické prvky (TrollEdit)
Vytvorený:	02.10. 2011
Stav:	predbežný
Vedúci projektu:	Ing. Peter Drahoš, PhD.
Vedúci tímu:	Bc. Lukáš Turský
Členovia tímu:	Bc. Marek Brath Bc. Adrián Feješ Bc. Maroš Jendrej Bc. Jozef Krajčovič Bc. Ľuboš Staráček
Kontakt:	tp-team-10@googlegroups.com

Obsah

Zoznam príloh

Príloha A: zápisnice zo stretnutí

Príloha B: preberacie protokoly

Príloha C: pravidlá dokumentácie

1 Úvod

Účelom tohto dokumentu je zdokumentovať riadenie tímu v rámci projektu textový editor obohatený o grafické prvky na predmete Tímový projekt. Projekt je riešený tímom č.10 s názvom „Innovators“ počas dvoch semestrov v akademickom roku 2011/2012.

1.1 Prehľad dokumentu

Na začiatku sa nachádza ponuka, ktorú sme vypracovali pri výbere témy projektu. Podarilo sa nám získať jednu z troch nami preferovaných tém. V tejto časti sú zároveň krátko predstavený členovia tímu. Nasleduje prerozdelenie rolí v rámci tímu a krátkodobé úlohy, ktoré sme doteraz riešili. Ďalšou kapitolou je plán projektu na zimný semester. Nasledujúca kapitola sa zaoberá firemnou kultúrou a nami používanými podpornými prostriedkami. Poslednou kapitolou sú kópie zápisníc zo stretnutí.

2 Ponuka

Nasleduje ponuka tak, ako sme ju odovzdali okrem titulnej strany:

2.1 Predstavenie členov tímu

Bc. Jozef Krajčovič
Absolvent odboru Informatika na FPV UCM v Trnave. Vypracoval bakalársku prácu na tému „Návrh lekárskeho informačného systému ambulancie“. Má skúsenosti s vývojom webových ako aj desktopových aplikácií. Používa väčšinu technológie a nástroje z dielne Microsoft. Zaujíma sa o tvorbu a vývoj používateľských rozhraní ako aj riadenie a motivovanie ľudí v tíme. Ovláda technológie: HTML/XHTML, PHP, JavaScript, C#, Visual Basic, C/C++, Java, Mysql, MSSQL a Oracle, .Net Framework (WPF, XAML, WCF), WindowsPowerShell, XML.

Bc. Adrián Feješ
Je absolventom študijného odboru Informatika na FIIT STU. Vo svojej bakalárskej práci sa venoval procesu refaktorizácie zdrojových kódov a jej nástrojovej podpore. Výsledkom práce bol nástroj vo forme Eclipse plug-inu, podporujúci rozpoznávanie a označovanie antivzorov v kóde. Má skúsenosti s vývojom aplikácií hlavne v programovacom jazyku Java. Svoje vedomosti ďalej rozvíja aj v praxi, kde pracuje ako Java programátor a zaoberá sa vývojom podnikových aplikácií. Ovláda technológie: C/C++, Java SE/EE, XML, XMLSchema, XPath, SQL, JavaScript,

Bc. Lukáš Turský
Vyštudoval obor Informatika na FIIT STU. Počas štúdia sa zameral najmä na vývoj aplikácií pre platformu Java SE a FX. V rámci bakalárskej práce analyzoval využitie Modelom riadenej architektúry pri tvorbe softvéru, pričom výstupom bolo úplné namodelovanie web aplikácie a jej následne implementovanie pre platformu Java EE (využitie Spring, Struts, Hibernate). Popri štúdiu získal skúsenosti v oblasti analýzy rizík a administrácie bezpečnosti bankových aplikácií. V dohľadnej dobe by sa chcel ďalej zamerať na vývoj webových aplikácií a rozšíriť znalosť databáz v rámci predmetu Pokročilé Databázové technológie.

Bc. Luboš Staraček
Absolvent študijného odboru Informatika na STU FIIT v Bratislave, vypracoval bakalársku prácu na tému „Štúdia realizácie zmien aspektovo-orientovaným spôsobom na úrovni modelu“. Za najpodstatnejšie získané zručnosti považuje osvojenie si objektovo a aspektovo

orientovaného vývoja softvéru, metódy paralelného programovania a princípy umelej inteligencie. V rámci mimoškolskej činnosti vytvoril funkčnú web aplikáciu v jazyku JavaFX.

Bc. Maroš Jendrej
Absolvent študijného odboru Informatika na STU FIIT v Bratislave, vypracoval bakalársku prácu na tému „Manažovanie dokumentov“. Má skúsenosti s vývojom desktopových aplikácií pre platformu JAVA SE. Počas bakalárskeho štúdia si osvojil základy programovania v rôznych programovacích jazykoch a tiež získal znalosti o tvorbe softvérových systémoch. Po ukončení bakalárskeho štúdia sa zamestnal na pozícií QA/Tester v spoločnosti zaoberajúcim sa vývojom počítačových hier. V dohľadnej dobe by sa chcel hlbšie oboznámiť s počítačovou grafikou a dizajnom používateľského rozhrania. Ovláda technológie: HTML/XHTML, XML, JAVA SE, C/C++, Assembler, UML, CUDA, MPI, OMP

Bc. Marek Brath
Absolvent študijného odboru Informatika na FPV UCM v Trnave, vypracoval bakalársku prácu na tému „Programovanie v Jave“. Používa hlavne prostredie Eclipse na vytváranie desktopových aplikácií. Ovláda technológie: Java, C++, C#, PHP, HTML, XHTML, CSS, PHP, SQL

2.2 Znalosti a zručnosti študentov (Znalosti)

2.2.1 Motivácia

V dnešnom svete plnom informácií je nájdenie a zostavenie tímu ľudí, obzvlášť takých ktorí sa takmer nepoznajú, často veľmi ťažko riešiteľný problém. Vidíme to aj teraz na nás, študentoch, že problémom je nedostatok a roztrúsenosť informácií o našich kolegoch. Veríme tomu, že my samí si možno časom začneme hovoriť, že zadelenie v rámci daného tímu nie je najideálnejšie.

Práve preto nás nadchla myšlienka vytvorenia centrálnej databázy schopností a znalostí jednotlivých študentov, ktorej plné využitie by mohlo siahať aj ďaleko do komerčnej sféry. Veď pokiaľ by bol takýto systém dostatočne používateľsky prívetivý a interaktívny, mohol by uľahčiť prácu nielen učiteľom, ale určite aj neskôr študentom napr. pri hľadaní zamestnania.

Veľkú výhodu vidíme najmä v tom, že sami by sme boli motivovaný zlepšovať sa a týmto spôsobom ovplyvňovať svoje ohodnotenie v rámci systému.

Na druhej strany nielen pre profesorov ale aj vedúcich prác je to spôsob ako efektívne zostaviť tým podľa jeho preferencií a teda si môže rozhodnúť aké kvality by mal takýto tým, prípadne aj jednotlivec spĺňať. Taktiež je to informačný spôsob ako efektívne využiť potenciál každého jednotlivca v rámci vytváraného týmu a touto cestou aj zvýšenie miery na jeho budúci úspech.

Rozhodne by sme chceli stáť u zrodu takéhoto systému, lebo veríme tomu, že na to máme ako tým všetky predpoklady a bolo by veľmi zaujímavé pokiaľ by sa takýto systém podarilo reálne nasadiť do prevádzky.

2.2.2 Koncept riešenia

Vzhľadom na to, že väčšina nášho týmu má väčšie či menšie skúsenosti s konceptom a využívaním Java EE technológií, tak by sme chceli práve túto platformu využiť pre vytvorenie požadovanej client-server webovej aplikácie, ku ktorej budú môcť používatelia voľne pristupovať.

V rámci riešenia vidíme viacero hľadísk na ktoré bude potrebné sa zamerať. V rámci prezentačnej vrstvy je to nutnosť využiť interaktívne zaujímavý framework, ktorý by sa použil pre vytvorenie používateľského prostredia, a ktorý bude na použitie dostatočne prívetivý. Dobre vieme, že je to jediná časť s ktorou pracuje používateľ priamo a mnohokrát rozhoduje o zániku či úspechu systému. V tomto smere ešte nemáme jasno, o aký framework by šlo a teda by bolo nutné spraviť krátku analýzu.

Pre jednoduchšiu orientáciu, by mohlo vyhľadávanie a najmä pridávanie znalostí mať v hlavnej časti u každého študenta len nejaké zhrnutie jeho znalostí – správne zvolené väčšie celky, ktoré by zoskupovali podobné znalosti, napr. aká je miera technických znalosti, spoločenských schopností, využívanie určitých typov nástrojov, takisto by bolo zaujímavé mať aj indikátor ako sa študentovi darí v škole.

Keďže bolo spomenuté, že na takomto projekte sa už v minulosti pracovalo, tak by sme chceli využiť niektoré časti tohto riešenia, ktoré už sú dostatočne vyriešené a sústrediť sa na podstatnejšie veci, ktoré sú spomenuté nižšie.

Z pohľadu aplikačnej logiky chceme venovať úsilie vytvoreniu mechanizmu, ktorý by vedel na základe daných schopností používateľa ďalej odvodiť bázu znalostí, ktorá by určite zlepšila šance pri filtrovaní a výbere. Reprezentácia znalostí by mala spĺňať požiadavky ako

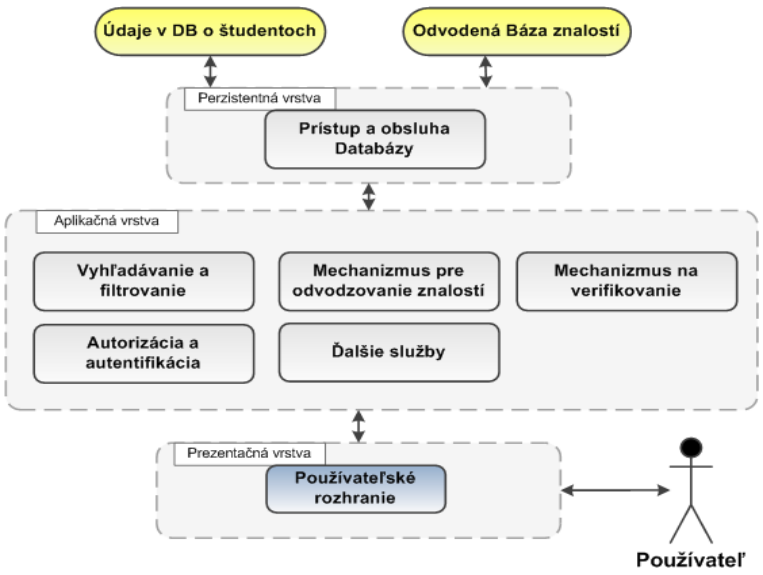
je rýchle vyhľadávanie a porovnávanie údajov, a preto spôsob reprezentácie musí byť jednoduchý a jednotný pre rôzne typy poznatkov a zručností.

Chceli by sme implementovať schopnosť automatizovane zadávať jednotlivé znalosti študentov, napr. ich hromadné pridávanie. V tomto smere by sa pre ich získavanie vo veľkej miere dalo využiť aj bodové hodnotenie v AIS pri jednotlivých zadaniach v rámci predmetov.

Vzhľadom na to, že pôjde o systém do ktorého bude mať prístup viacero skupín ľudí, navrhujeme vytvorenie viacerých úrovní prístupov (rolí) a k ním priradiť možné akcie, prípadne časti systému na ktoré by im tieto role dávali prístup. Teda logicky podľa toho do akej kategórie daný používateľ patrí, také akcie budú môcť vykonávať v systéme. Tu by bolo vhodné implementovať jednoduché pridávanie právomocí v rámci jednotlivých rolí, napr. odklikávanie akcií, alebo výber z listu.

Plánujeme implementovať spôsob overovania a kontroly študentmi zadávaných schopností, tak čo sa týka určitej miery verifikovateľnosti zadávaných schopností študentov, jeden spôsob vidíme v možnosti nechať zaslať požiadavku oprávnenej osobe na overenie.

Rozhodnutie koho potrebujeme zohnať a aké by mali byť požiadavky na študenta/tým by mali byť ponechané čisto na zadávateľa.



2.3 Digitálne divadlo (Divadlo)

2.3.1 Motivácia

Ovládanie softvéru pomocou ľudských pohybov a gest, bez nutnosti použitia klávesnice alebo myši, je samo o sebe veľmi zaujímavá a aktuálna téma. Obzvlášť, keď je tento projekt zameraný na tvorbu umeleckého diela, kde je zároveň výsledok tejto tvorby premietaný na plátno v reálnom čase. Myslíme si, že práca na takomto projekte bude pre nás zaujímavá, bude nás baviť, a tiež, v neposlednom rade, získame množstvo užitočných skúseností v zaujímavej oblasti IT.

Najmä kvôli týmto dôvodom náš tím zaujala táto téma, a chceli by sme sa podrobnejšie oboznámiť s možnosťami, ako využiť senzor Kinect na rozpoznávanie obrazovej informácie, pohybov a gest. Pokúsili by sme sa o vytvorenie originálneho riešenia, v ktorom by umelec pred plátnom pomocou svojho vlastného tela vytváral obraz. Ponúkli by sme mu na tvorbu diela nástroje, ktoré sú bežne v štandardných programoch na PC (Skicár, Adobe Photoshop, MyPaint...). Išlo by o vnorenie umelca do počítačovej reality, kde by aj bez tableta, či myšky mohol maľovať obraz.

2.3.2 Koncept riešenia

Standalone aplikácia pre platformu Windows XP, Vista a 7. Využívali by sme existujúce knižnice na detekciu pohybov a gest človeka. V rámci prípravy na vytvorenie tejto ponuky sme tiež vykonali analýzu niekoľkých video prezentácií umiestnených na portáli youtube, napríklad o používaní senzora Kinect na ovládanie konzoly X-Box 360 a podobne, čo nám môže poskytnúť veľa inšpirácie pri navrhovaní riešenia pre potreby tohto projektu.

Naše navrhované riešenie by mohlo byť akýmsi wraperom na ľudské telo, pomocou ktorého sa bude vytvárať obraz, v prípade požiadavky na stereo projekciu sme pripravení pokúsiť sa o vytvorenie výstupného obrazu v troch dimenziách. Využili by sme pri tom rozpoznávanie hĺbky obrazu, ktorá nám je týmto senzorom ponúknutá. Prípadne, ak senzor Kinect umožňuje aj rozoznávanie hlasu, mohlo by stáť za zváženie umožniť aj ovládanie kombináciou ľudských gest a hlasu. Tu by ale bolo dôležité zabezpečiť, aby bolo možné nastaviť ovládanie hlasom tak, že by príkazy hlasom mohla dávať iba oprávnená osoba, a nie ktokoľvek. Inak by mohli vznikať komplikácie, kde by počas používania tohto softvéru napríklad na prezentáciu mohol do tejto prezentácie vstupovať ktokoľvek z publika, čo je nežiaduce.

Jednou z alternatív pre overenia riešenia by mohlo isť o vytvorenie prívetivého ovládania pre existujúcu open source aplikáciu MyPaint, slúžiacu na tvorbu obrázkov. Jej výhodou je jednoduché a minimalistické používateľské rozhranie, neobmedzený canvas bez nutnosti zmeny jeho rozmerov a schopnosť využívania grafického tabletu. Rovnako ako pri kreslení keď využívame grafický tablet by sme mohli využiť aj senzor Kinect, ktorý by za pomoci hĺbky obrazu dokázal určiť kedy umelec naťahuje ruku a teda snaží sa v obraze kresliť. Intenzitu kreslenia by sme určovali ako hlboko umelec ponorí svoju ruku do obrazu, je to podobne ako sa na grafickom pere určuje stupeň prítlaku. Rozlišovali by sme tiež pravú a ľavú ruku, jedna by bola ako štetec a za pomoci druhej ruky by umelec vytváral gestá takto by prepínal medzi typmi štetcov, nastavoval farbu alebo inými funkciami. Trup umelca bude dynamickým stredom a maximálne natiahnutá ruka dopredu bude zaznamenané ako maximálna intenzita prítlaku štetca, nemôže sa tu stať niečo také, že štetec bude reagovať neprimerane.

Taktiež by mohlo byť zaujímavé implementovať ovládanie gestami do softvéru na tvorbu, respektíve spúšťanie prezentácií. Napríklad do open source programu OpenLP, ktorý okrem spúšťania prezentácií umožňuje aj prehrávanie videí, vytváranie a zobrazovanie galérií obrázkov a ďalšie. http://openlp.org/en/features.

2.4 Textový editor obohatený o grafické prvky (TextEdit)

2.4.1 Motivácia

Tato téma nás predovšetkým zaujala svojou myšlienkou vytvoriť akýsi multiplatformový graficky editor, ktorý využije grafické prvky na zvýraznenie štruktúr textu pomocou grafických blokov a tým podporili myšlienku „literate programming, čo v súčasnosti veľa podobných riešení dosiaľ neexistuje a taktiež fakt, že práca na editore je z 50% už hotová. Ďalšou motiváciu pre nás je, že sa pri tomto projekte môžeme rozšíriť svoje znalosti a zručnosti o nové technológie a postupy v danej doméne, ktorá je pre nás zaujímavá. Uvedomujeme si, že s danou doménou nemáme veľa praktických skúsenosti čo sa môže zdať ako nevýhoda, ale opak je však pravdou a o to viac to bude pre nás väčšia výzva, aby sme vytvorili kvalitný produkt, ktorý bude úspešný a mohol by presadiť aj v praxi.

2.4.2 Koncept riešenia

Cieľom tohto projektu bude pokračovať vo vývoji existujúceho multiplatformového editora (TrollEdit), ktorý bol vytvorení predchádzajúcim tímov „*UFOPAK*“. Našim zameraním pre

editor bude rozšírenie stavajúcej funkcionality pre reálne nasadenie editora do praxe. Najväčšiu zmenou bude vylepšenie používateľského rozhrania, ktoré v súčasnom editore nie je tak ako u podobných editor čo sa týka dizajnu nezaujímaví t.j. klasický dizajn „*ala notepad*“.

Pri implementácii budeme predovšetkým vychádzať z už použitých technológii ako knižnica Qt, skriptovací jazyk Lua a podobne plus niektoré nami zvolené technológie, ktoré sa rozhodneme použiť po podrobnej analýze súčasného editora.

Čo sa tyká rozšírenia funkcionality plánujeme implementovať tieto vylepšenia:

- Možnosti „undo“/ „redo“.
- Detekcia pachov kódu.
- Možnosť rozšírených nastavení priamo v editore
- Určitý druh fulltextového vyhľadávania s prípadnou optimalizáciou na najčastejšie vyhľadávané výrazy.
- Možnosť exportovania súboru do iných formátov (XML, WORD)
- Schopnosť detegovať určite ukazovatele v zdrojovo kóde ako index udržovatelnosti, cyklomatická zložitosť, hodnoty fan in a fan aut, ktoré by boli zobrazene v určitej tabuľke.

Taktiež plánujeme čo najvhodnejšie použiť známe návrhové vzory, aby sme zabezpečili vysokú modularitu systému a tým umožnili neskoršie pridávanie a modifikovanie funkcionality.

Ohľadom spomínaného dizajnu používateľského rozhrania plánujeme vďaka podpore Qt modulu pre vývojové prostredie Visual Studio použiť najmodernejšie technológie ako WPF (Windows presentation fundations), XML.

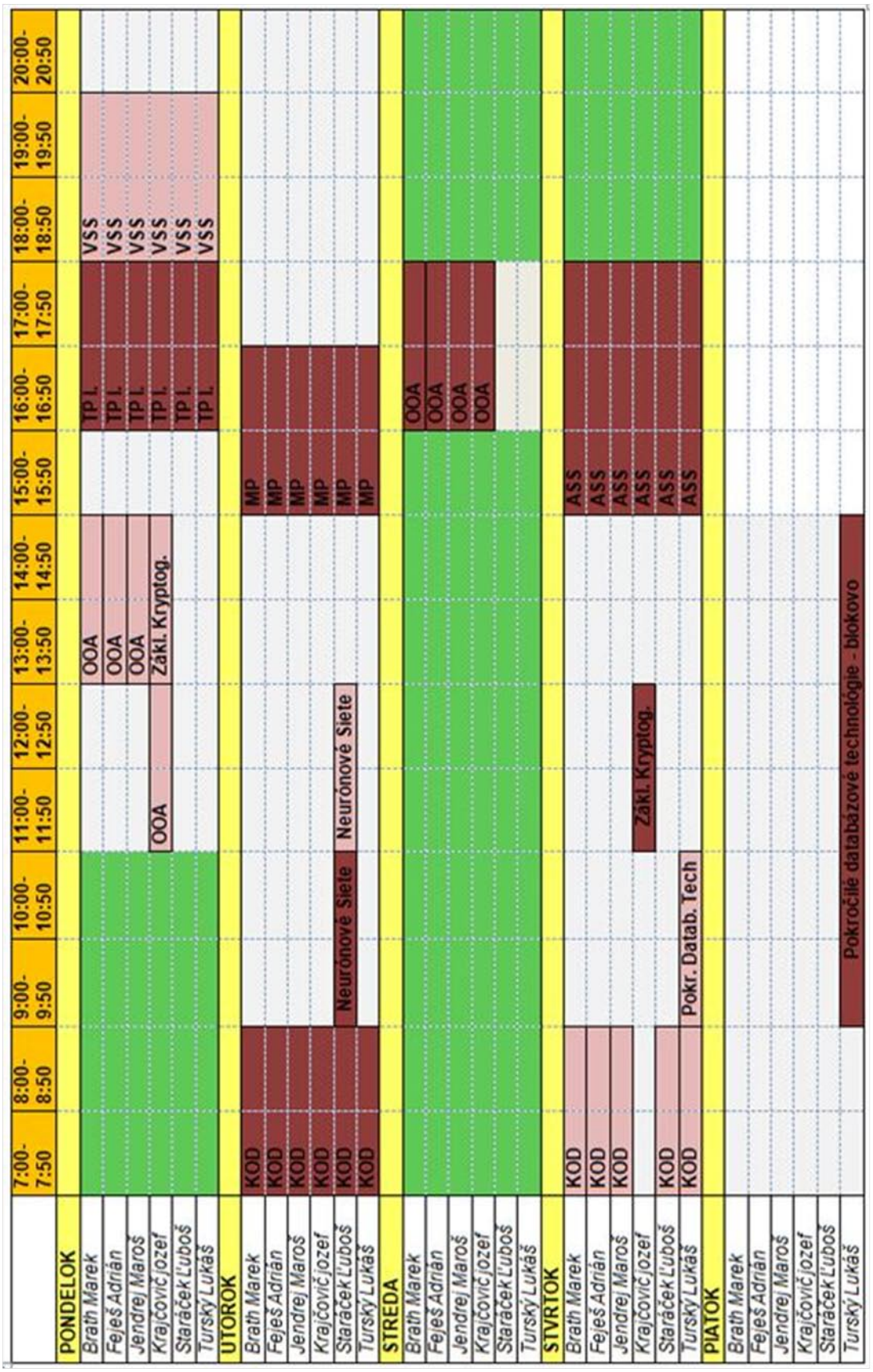
Tieto technológie nám umožnia navrhnúť si dizajn podľa vlastnej fantázie bez zdĺhavého programovania pri ktorom by sme museli použiť rôzne grafické knižnice čo v tomto prípade odpadá. Plánujme návrh dizajn používateľského rozhrania v štýle „Office“ t.j. použiť dobre známi „*Ribbon*“, ktorý je stále častejšie používaní v desktopových aplikáciách.

Veríme, že nami navrhnute riešenie vo finálnej verzii bude kvalitný produkt, ktorý nájde uplatnenie v praxi.

3 Zoradenie všetkých tém podľa priority

Priorita	Názov témy	Číslo témy
1.	Znalosti a zručnosti študentov (Znalosti)	13
2.	Digitálne divadlo (Divadlo)	3
3.	Textový editor obohatený o grafické prvky (TrollEdit)	11
4.	Štatistický preklad voľného textu (Preklad)	9
5.	Inteligentná hra pre mobilné zariadenia (MobHra)	8
6.	Rozvrhový systém novej FIIT (Rozvrhy)	12
7.	Plagiáty na webe (Plagiáty)	4
8.	Simulácia davu (Dav)	15
9.	Personalizované odporúčanie (Odporúčanie)	5
10.	Osobný manažment fyzickej aktivity pomocou mobilných zariadení (Aktivita)	2
11.	Editovanie viacrozmerného grafu prepojenia informácií v dokumentoch (Dokumenty)	16a
12.	Virtuálna FIIT (VirtFIIT)	14
13.	RoboCup - tretí rozmer (RoboCup	7
14.	Webový editor pre TeX (WebEdit)	10
15.	Tvorba "ľahko" sémantického obsahu pre adaptívny webový (výučbový) portál (ALEF)	6
16.	Imagine Cup 2012: Game Design (ICup2012) - pridelená	1
17.	3D UML (3D UML)	16b

4 Rozvrh členov tímu pre zimný semester



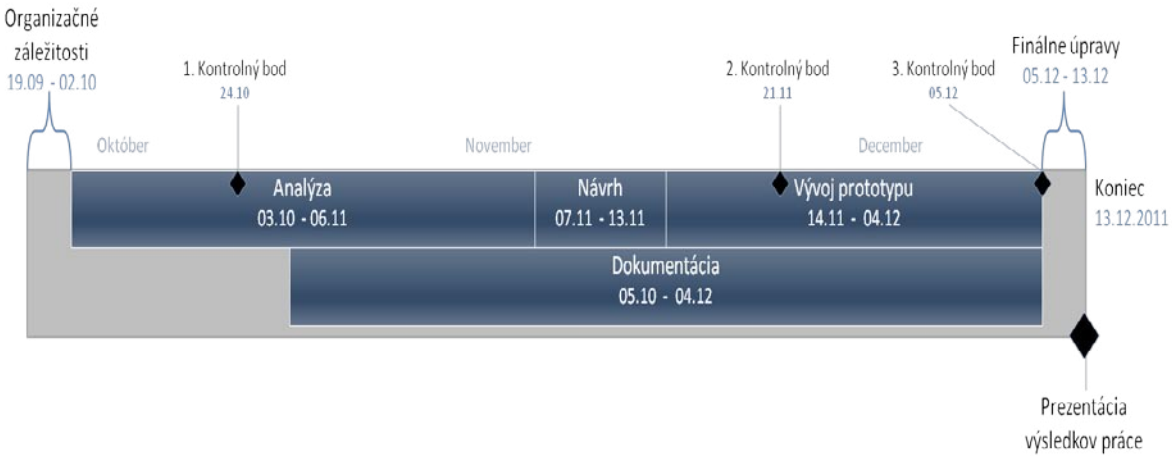
5 Plán

Po dôkladnej analýze dostupných metodík sme sa rozhodli, že budeme vyvíjať inkrementálnym a iteratívnym spôsobom. Naše rozhodnutie ovplyvnili najmä výhody takéhoto prístupu k vývoju. Plán projektu samozrejme musíme prispôsobiť vlastnostiam inkrementálneho a iteratívneho vývoja. Celý projekt sa rozloží na dobre definované a použiteľné časti (inkrementy), ktoré postupne integrujeme do celku. Získame tak prehľadnejší a ľahšie manažovatelný vývojový proces. Jednotlivé časti budeme iteratívne vyvíjať, čo môže vo veľkej miere zvýšiť kvalitu výsledkov.

5.1 Hrubý plán pre zimný semester

1. týždeň	Vytvorenie tímu Rozdelenie rolí v tíme
2. týždeň	Výber preferovaných tém Vypracovanie a odovzdanie ponúk
3. týždeň	Vytvorenie webovej stránky, plagátu a loga tímu Analýza a výber podporných prostriedkov Analýza stavu predošlého projektu (preštudovanie technickej dokumentácie a dokumentácie riadenia)
4. týždeň	Špecifikácia požiadaviek Analýza použitých technológií a nástrojov
5. týždeň	Analýza zdrojových kódov aplikácie TrollEdit Analýza použitých technológií Vytvorenie predbežnej verzie technickej dokumentácie a dokumentácie riadenia
6. týždeň – kontrolný bod	Prepracovanie špecifikácie požiadaviek Určenie priority jednotlivých požiadaviek Diskusia o možnostiach implementácie jednotlivých funkcionalít
7. týždeň	Analýza implementácie určených funkcionalít
8. týždeň	Návrh implementácie funkcionality Návrh GUI Odovzdanie dokumentácie analýzy, špecifikácie a návrhu riešenia
9. týždeň	Implementácia prototypu fáza I.
10. týždeň – kontrolný bod	Testovanie a oprava chýb fázy I Implementácia prototypu fáza II Kontrola stavu technickej dokumentácie a dokumentácie riadenia
11. týždeň	Implementácia prototypu fáza III Testovanie a oprava chýb fázy II, III Vypracovanie finálnej verzie technickej dokumentácie a dokumentácie riadenia

12. týždeň – 3.kontrolný bod	Odovzdanie prototypu spolu s dokumentáciou
13. týždeň	Prezentácia výsledkov práce Vypracovanie priebežnej správy pre TP Cup



Obr. 1 Časová os hrubého plánu

6 Úlohy členov tímu

Táto kapitola obsahuje informácie o rolách jednotlivých členov týmu a krátkodobých úlohách, ktoré sme riešili pri tvorbe projektu v zimnom semestri.

6.1 Dlhodobé úlohy

Jednotliví členovia tímu zastávajú nasledujúce dlhodobé úlohy na projekte

Bc. Lukáš Turský	Manažér tímu Manažér komunikácie Správca webového sídla
Bc. Jozef Krajčovič	Zástupca vedúceho tímu Manažér podpory vývoja Manažér tvorby dokumentácie
Bc. Adrián Feješ	Manažér rozvrhu a plánovania
Bc. Maroš Jendrej	Manažér kvality a testovania
Bc. Ľuboš Staráček	Manažér rizík
Bc. Marek Brath	Manažér monitorovania

6.2 Krátkodobé úlohy

Rozpis krátkodobých úloh, ktoré boli riešenie v zimnom semestri sú popísané v nasledujúcej tabuľke.

ID	Popis úlohy	Zodpov. osoba	Dátum vzniku	Dátum ukončenia	Stav
01.	Vytvorenie webovej stránky pre prezentáciu tímu	Lukáš	28.09.2011	29.09.2011	dokončená
02.	Vybrať podporné nástroje pre vývoj	Všetci	28.09.2011	30.09.2011	dokončená
03.	Preskúmanie nástroja TrollEdit a porovnanie ho s ďalšími nástrojmi a navrhnutie zmeny funkcionality	Všetci	28.09.2011	30.09.2011	dokončená
04.	Vytvorenie loga a plagátu tímu	Jozef	28.09.2011	29.09.2011	dokončená
05.	Spojazdnenie virtuálneho stroja umiestneného	Lukáš	29.09.2011	30.09.2011	dokončená
06.	Podrobná analýza možnosti, ktoré ponúkajú technológie QT,Lua	Všetci	12.10.2011	18.10.2011	dokončená
07.	Vytvorenie dokumentácie štýlu programovania	Jozef	12.10.2011	16.10.2011	dokončená
08.	Analýza možnosti ako implementovať funkcionalitu (2 módy, ShortCuts, Undo/Redo)	Všetci	12.10.2011	-	rozpracovaná

09.	Urobiť redesign používateľského rozhrania nástroja TrollEdit	Jozef	12.10.2011	-	odložená
10.	Vytvoriť logo pre TrollEdit	Jozef	12.10.2011	22.10.2011	dokončená
11.	Uppdate webovej stránky (doplnenie technológii, fotky vedúceho)	Lukáš	19.10.2011	23.10.2011	dokončená
13.	Prepojenie GitHub z Redmine	Marek	19.10.2011	23.10.2011	dokončená
14.	Analýza doxygen	Lukáš	19.10.2011	25.10.2011	dokončená
15.	Vytvorenie prihlášky na TP CUP	Jozef	19.10.2011	25.10.2011	dokončená
16.	Vytvorenie predbežnej verzie technickej dokumentácie	Jozef	19.10.2011	25.10.2011	dokončená
17.	Vytvorenie predbežnej dokumentácie riadenia	Jozef	19.10.2011	25.10.2011	dokončená
18.	Prenesenie súborov z SVN na GitHub	Jozef	19.10.2011	23.10.2011	dokončená
19.	Vytvorenie statickej web stránky pre TrollEdit	Adrián	19.10.2011	25.10.2011	dokončená
20.	Podrobná analýza možnosti, ktoré ponúkajú technológie QT	Jozef, Lukáš, Adrián	19.10.2011	25.10.2011	dokončená
21.	Podrobná analýza možnosti, ktoré ponúkajú technológie Lua	Marek, Ľuboš, Maroš	19.10.2011	25.10.2011	dokončená
22.	Zrušenie starého repozitára a presun taskov	Marek	26.10.2011	26.10.2011	dokončená
23.	Refaktorizácia zdrojového kódu podľa nami zadefinovaného štýlu programovania	Adrián	26.10.2011	31.10.2011	dokončená
24.	Podrobná analýza súčasného stavu TrollEditu	Lukáš	26.10.2011	2.11.2011	dokončená
25.	Špecifikácia požiadaviek	Jozef	26.10.2011	28.10.2011	dokončená
26.	Návrh funkcionality pre UNDO/REDO	Adrián	26.10.2011	01.11.2011	dokončená
27.	Analýza a návrh paralelného spracovania syntaxu v Qt	Lukáš		-	rozpracovaná
28.	Analýza spracovania syntaktického stromu v jazyku LUA	Ľuboš, Maroš	26.10.2011	02.11.2011	dokončená
29.	Analýza vytvárania shortcutov	Jozef	26.10.2011	01.11.2011	dokončená
30.	Doplnenie štýlu programovania o syntax Doxygen a jeho použitie	Lukáš	26.10.2011	29.10.2011	dokončená
31.	Analýza a návrh mapovania objektov z C++ do jazyka LUA	Ľuboš, Maroš	26.10.2011	29.10.2011	dokončená
32.	Napísať návod buildovania TrollEditu	Marek	26.10.2011	27.10.2011	dokončená
33.	Analýza a návrh paralelného spracovania syntaxe v QT	Lukáš	02.11.2011	09.11.2011	dokončená
34.	Návrh funkcionality pre UNDO/REDO	Adrián	02.11.2011	09.11.2011	dokončená

35.	Návrh spracovania syntaktického stromu v jazyku LUA	Ľuboš, Maroš	02.11.2011	09.11.2011	dokončená
36.	Návrh funkcionality pre shortcuts	Marek	02.11.2011	09.11.2011	dokončená
37.	Vytvorenie Use Case diagramov	Jozef	02.11.2011	05.11.2011	dokončená
38.	Analýza jazyka QML pre integráciu používateľského rozhrania	Jozef	02.11.2011	05.11.2011	dokončená

6.2.1 Autori jednotlivých častí dokumentácie riadenia

Nasledujúca tabuľka zobrazuje príspevky jednotlivých členov tímu k dokumentácii riadenia v zimnom semestri.

Kapitola	Autor
1 Úvod	Jozef Krajčovič
2 Ponúka 2.2 Znalosti a zručnosti študentov 2.3 Digitálne divadlo 2.4 Textový editor obohatený o grafické prvky	 Lukáš Turský & Adrián Feješ Maroš Jendrej & Ľuboš Staráček Jozef Krajčovič
3 Plán	Adrián Feješ
4 Role a úlohy členov tímu	Jozef Krajčovič
5 Firemná kultúra 5.1 Manažment rozvrhu 5.2 Manažment rizík 5.3 Manažment komunikácie 5.4 Manažment podpory vývoja 5.5 Manažment kvality 5.6 Manažment monitorovania 5.7 Manažment tvorby dokumentácie 5.8 Štýl programovania	 Adrián Feješ Ľuboš Staráček Lukáš Turský Jozef Krajčovič Maroš Jendrej Marek Brath Jozef Krajčovič Jozef Krajčovič & Lukáš Turský

6.2.2 Autori jednotlivých častí technickej dokumentácie

Kapitola	Autor
1 Úvod	Jozef Krajčovič
2 Analýza 2.1 Existujúce riešenia editorov 2.2 Analýza predchádzajúceho riešenia nástroja TrollEdit 2.3 Analýza použitých technológií	 Marek Brath Lukáš Turský Jozef Krajčovič
3 Špecifikácia požiadaviek	Jozef Krajčovič
4 Návrh riešenia 4.1 Diagram prípadov použitia 4.2 Architektúra programu 4.3 Návrh GUI	 Jozef Krajčovič Jozef Krajčovič Jozef Krajčovič

5 Implementácia prototypu	

6.2.3 Prehľad času stráveného na projekte

V nasledujúcej tabuľke sú zaznamenané časy, ktoré sme venovali prácou na tímovom projekte v priebehu jednotlivých týždňov.

6.2.3.1 Zimný semester

	Marek Brath	**Adrián Feješ**	**Maroš Jendrej**	**Jozef Krajčovič**	**Ľuboš Staráček**	**Lukáš Turský**	**Spolu**	**Priemer**
1.týždeň								
2.týždeň								
3.týždeň								
4.týždeň								
5.týždeň								
6.týždeň								
7.týždeň								
8.týždeň								
9.týždeň								
10.týždeň								
11.týždeň								
12.týždeň								
Spolu								

// tu bude nejaký čiarový graf vygenerovaný z vyššej verzie Office lebo 2003 je ...

7 Firemná kultúra

Tato kapitola opisuje firemnú kultúru ktorú sme si definovali v rámci tímového projektu. Definovane sú metodiky, ktoré musí každý člen tímu dodržiavať pri práci na projekte.

7.1 Manažment rozvrhu

Vytvorí Adrián

7.2 Manažment rizík

Manažment rizík sa musel v tomto prípade prispôsobiť špecifikám práce na projekte v malom tíme v školskom prostredí. To ovplyvňuje napríklad nemožnosť nastania rizika nedodržania rozpočtu, a podobne. Najprv sú v kapitole 1.1 identifikované riziká, ktoré môžu nastať. Potom sú v kapitole 1.2 tieto riziká (subjektívne) ohodnotené, podľa toho s akou pravdepodobnosťou môžu nastať a aké veľké dopady na úspech projektu budú mať, ak nastanú. Miera dopadu je dôležitejšia, preto viac ovplyvňuje celkové ohodnotenie rizika. Tu sú riziká zoradené zostupne podľa ich celkového ohodnotenia.

7.2.1 Identifikácia rizík

Číslo	Identifikované riziko	Možný spúšťač	Ošetrenie rizika
1	Odchod člena tímu	Prílišné nároky na niektorého člena tímu, nezvládnutie tlaku, strata motivácie	Rovnomerné rozkladanie úloh na všetkých členov tímu (zamedzenie rizika), prerozdelenie úloh na zvyšných členov (prijatie rizika)
2	Nedodržanie termínov	Nedisciplinovanosť členov tímu	Stanovenie dostatočne skorých termínov, prísna kontrola splnenia úloh vedúcim tímu (zamedzenie rizika)
3	Nedodržanie požiadaviek	Nesprávne pochopenie alebo neúplnosť požiadaviek zákazníka	Častá komunikácia so zákazníkom (vedúcim tímu), skoré prototypovanie (zamedzenie rizika)
4	Nezhody medzi členmi tímu	Členovia tímu budú mať rozdielne názory na určitú časť projektu, spôsob implementácie alebo riadenia	Častá komunikácia v rámci tímu, dodržiavanie firemnej kultúry a vypracovaných metodík (zamedzenie rizika)
5	Nezvládnutie novej technológie	V projekte bude nasadená technológia, s ktorou žiadny člen tímu nemá skúsenosti	Dôkladná analýza danej technológie, experimentovanie s technológiou (zamedzenie rizika), konzultácia s expertom na danú technológiu (prenos rizika)

7.2.2 Klasifikácia rizík

Ako vzorec na výpočet celkového ohodnotenia rizika bol použitý:

*celkove_ohodnotenie = pravdepodobnosť*3 + miera_dopadu*7*

Číslo	Identifikované riziko	Pravdepodobnosť nastania [0-10]	Miera dopadu [0-10]	Celkové ohodnotenie [0-100]
1	Odchod člena tímu	2,5	9	70,5
2	Nedodržanie požiadaviek	3	8,5	68,5
3	Nezvládnutie novej technológie	2	8	62
4	Nedodržanie termínov	4	5	47
5	Nezhody medzi členmi tímu	9	1,5	28,5

7.3 Manažment komunikácie

Vytvorí Lukáš

7.4 Manažment podpory vývoja

Vytvorí Jozef

7.5 Manažment kvality

Manažment kvality...

7.5.1 Manažment testovania

Manažment testovania zabezpečuje kontrolu vytváraného softvéru, jeho cieľom je minimalizovanie šance aby obsahoval nejaké chyby. Je to neustále sa opakujúci proces a začína už po implementácii prvej iterácie až po ukončenie vývoja a nasadenie softvéru pre zákazníka. Neznamená len spúšťanie testov, ale je to predovšetkým plánovanie a riadenie procesov pred a po vykonaní týchto testov.

V ďalších častiach dokumentu sú procesy manažmentu testovania nastavené pre potreby veľkosti menšieho tímu (6-7 ľudí) a iteratívny spôsob vývoja.

Manažment testovania projektu treba začať po vytvorení prvej iterácií projektu. Po tejto prvej iterácií treba nastaviť testovacie prostredie. Ďalej treba vytvoriť testovacie scenáre a uložiť ich k ostatným do nástroja na správu testovacích scenárov. Pre všetky testovacie scenáre treba vytvoriť unit testy. Následne treba testovať všetky vytvorené unit testy od prvého až po posledný. Ak po testovaní v projekte nachádzame chyby, treba ich opraviť

a skontrolovať. Ak projekt po testovaní neobsahuje žiadne chyby tak treba zhodnotiť výsledky testovania a zistiť či daná iterácia projektu bola posledná, ak áno, nepridávajú sa už ďalšie testovacie scenáre a nastane koniec testovania projektu. Na obrázku č. 1. sú znázornené procesy manažmentu testovania a v tabuľke č. 1. je určené ich poradie a kapitola, v ktorej sa nachádzajú.

- **Manažér kvality** – navrhuje a vytvára testovacie scenáre, dohliada na tvorbu unit testov a priebeh testovania, okrem toho sa zaoberá správnosťou tvorby jednotlivých procesov súvisiacich s testovaním a zhodnocuje ich na konci každej iterácie

Tab. č. 2. Roly a zodpovednosti v manažmente testovania

Rola	**Zodpovednosť**
Programátor	vyvíja aplikáciu
	opravuje reportované chyby v aplikácii
Tester	vytvára unit testy podľa testovacích scenárov
	testuje aplikáciu pomocou unit testov
	reportuje a klasifikuje chyby
	kontroluje opravené chyby
Manažér kvality	navrhuje a vytvára testovacie scenáre
	pozoruje výskyt chýb
	kontroluje klasifikovanie chýb
	kontroluje výsledky testovania a zhodnocuje ich

7.6 Manažment monitorovania

Vytvorí Marek

7.7 Manažment tvorby dokumentácie

V tejto časti sú definovane pravidla riadenia písania dokumentácii, ktoré musí každý člen tímu dodržiavať v rámci firemnej kultúry.

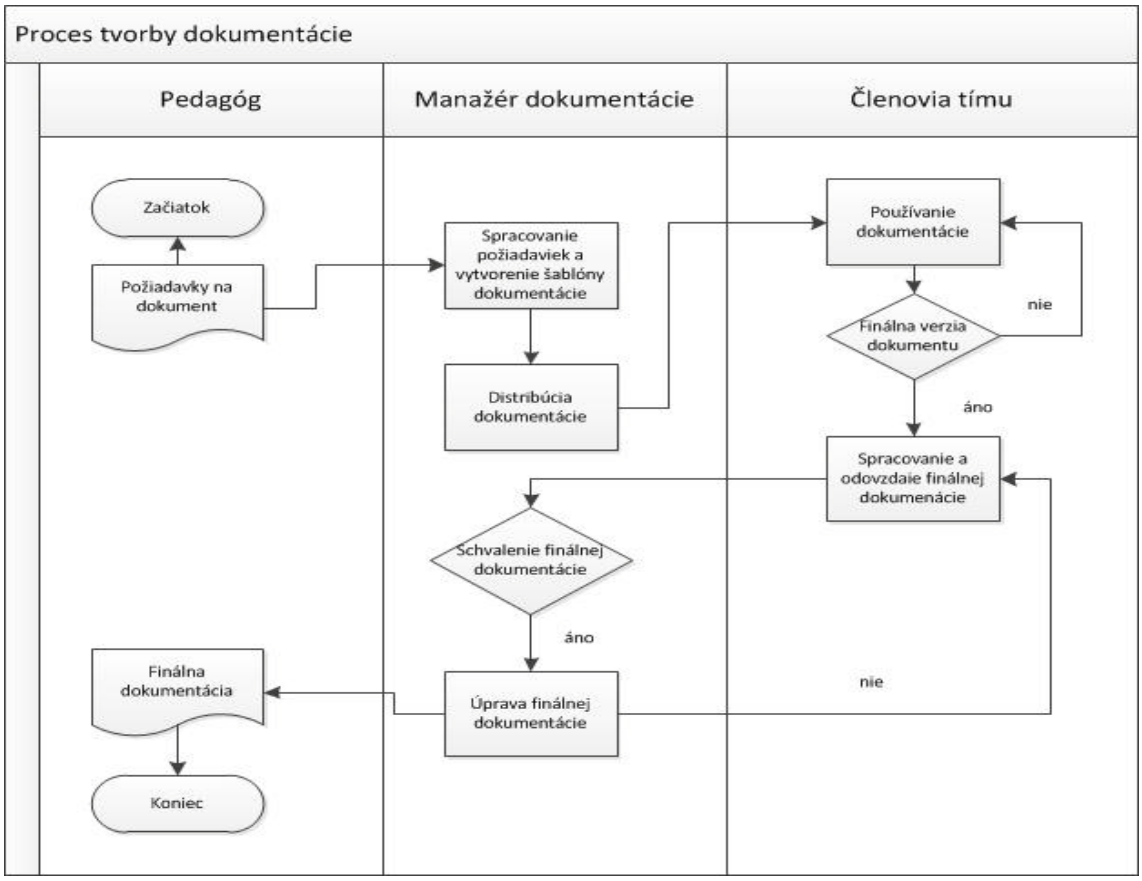
7.7.1 Role a zodpovednosti

Rola	**Zodpovednosť**
Manažér dokumentácie	- Vytvára šablóny pre dokumentácie - Zodpovedá za obsah, dodržiavanie formátovania a kontroluje chyby v dokumentácii - Vytvára a verifikuje finálnu dokumentáciu ktorú treba odovzdať
Zapisovateľ	- Pridáva obsah do dokumentácie - Môže meniť obsah dokumentácie po dohode z manažérom

7.7.2 Základne pravidla pri písaný dokumentácie

Na základe dodržania konzistencie pri písaní dokumentácie boli spísané všeobecné pravidlá pre písanie dokumentácie, ktorými by sa mali všetci členovia tímu riadiť. Vzhľadom na rozsah tohto dokumentu, sa pravidlá nachádzajú v **Prílohe C – Pravidlá dokumentácie**.

7.7.3 Postup tvorby dokumentácie



Obr.1 Proces tvorby dokumentácie

1) Na začiatku sú definované požiadavky od nášho vedúceho tímu (pedagóga) alebo od prof. Bielikovej.
2) Na základe týchto požiadaviek vytvorí manažér dokumentácie šablónu dokumentácie, uloží do repozitára a oznámi členom tímu účel dokumentácie.
3) Každý člen tímu používa dokumentáciu a pridáva svoj časť, ktorá mu bola pridelená manažérom tímu.
4) Ak každý člen tímu doplnil svoju časť do dokumentácie, manažér dokumentácie verifikuje obsah a v prípade výskytu chýb oznámi zodpovednému za časť obsahu v ktorej sú chyby aby ju prerobil. Ak je dokumentácia v poriadku tak ju manažér dokumentácie označí ako finálnu a znemožni jej úpravu. Takto vytvorenú finálnu dokumentáciu archivuje do repozitára pod platným číslom verzie.
5) Manažér dokumentácie odovzdá finálnu dokumentáciu nášmu vedúcemu tímu.

7.7.4 Vytváranie zápisníc zo stretnutí

Zápisnice sa vytvára podala šablóny ktorá je umiestnená v tímov repozitáre. Zápisnicu vytvára ten člen tímu, ktorý je nato určený podľa poradovníka. V zápise zo stretnutia by malo byť zachytené všetko, o čom sa na stretnutí diskutovalo. A taktiež určenie úloh, ktoré vyplynuli zo stretnutia a tiež vyhodnotenie plnenia úloh z predchádzajúceho stretnutia.

7.7.4.1 Pravidlá vytvárania zápisníc

1) Zápisnicu vytvára člen tímu podľa šablóny
2) Zápisnicu treba vytvoriť do 8 hodín od ukončenia tímového stretnutia
3) Zápisnicu uložiť do repozitára a oznámiť členom tímu o jej vytvorený
4) Členovia tímu verifikujú úplnosť zápisu v prípade nejakej nezrovnalosti oznámi zapisovateľovi zápisnice aby ju modifikoval
5) Manažér plánovania pridá úlohy, ktoré treba vykonať zo zápisnice do nástroja Redmine.

7.7.4.2 Pravidlá vedenia zápisníc

1) Tvorca zápisnice ju prinesie na oficiálne stretnutie
2) Tvorca zápisnice oboznámi vedúceho tímu z stavom vykonania úloh zo zápisnice
3) Zapisovateľ, ktorý je určený podľa poradovníka pozorne počúva a zapisuje body o ktorých sa diskutuje na tímovom stretnutí.

7.8 Štýl programovania

V tejto kapitole definujeme štýl písania kódu, ktorý budeme dodržiavať v rámci firemnej kultúry tímového projektu. Definované sú pravidlá, ktoré musí každý člen tímu dodržiavať pre sprehľadnenie zdrojového kódu a tým zamedzeniu možných nedorozumený a konfliktov, ktoré môžu vzniknúť z nejednotného štýlu programovania.

7.8.1 Vytváranie názvov

Použitie správnych názvov je kľúčové k sprehľadneniu kódu, názvy treba zvoliť zmysluplne aby vystihovali podstatu riešenia.

Názvy budú písane po anglicky

Triedy

1. používať notáciu PascalCase
2. názvy by mali byť podstatnými menami

Metódy

1. používať notáciu camelCase
2. názvy by mali byť slovesného tvaru

Premene

1. názvy sú písane malými písmenami
2. voliť zmysluplne názvy nie nič nehovoriace skratky ako (napr. „v“)
3. v prípade dlhších názvov používať pre oddelenie slov podtrhovník (narp. „nazov_nazov“)
4. ak je možne tak premenu v tom istom riadku aj inicializovať

Konštanty

1. písane sú veľkými písmenami

Ovládacie prvky (GUI)

1. používať Maďarsku notáciu t.j. prefix, ktorý vystihuje o aký typ ovládacieho prvku ide (napr. „btnOK“ - btn pre tlačidlo a OK je názov tlačidla)

7.8.2 Odsadenia

Pre sprehľadnenie štruktúry blokov v zdrojovom kóde treba dodržiavať nasledujúce pravidla:

1. Každý vnorený riadok musí byť odsadený tabulátorom o jednu pozíciu do ľavá

Správne:

```
nazovMetody()
{
       if()
      {
            nejakyprikaz;
       }
}
```

7.8.3 Písanie zátvoriek

1. Pri písaní zložených zátvoriek nepoužívať štýl K&R!

Správne:

```
if()
{
  nejakyprikaz;
}
```

Nesprávne:

```
if() {
   nejakyprikaz;
}
```

2. Písanie okrúhlych zátvoriek za kľúčovým alebo nejakým príkazom bez použitia väčšieho počtu medzier.

Správne:

```
if (a == 2)
```

Nesprávne:

```
if      (a == 2)
```

7.8.4 Písanie komentárov pre potreby nástroja doxygen

Vo všeobecnosti sa písanie komentárov pre nástroj doxygen skoro vôbec nelíši od bežného komentovania. Pokiaľ niekto používal komentáre pre Javadoc, alebo iný dokumentačný prístup, tak je to v podstate to isté.

Pre Doxygen platí, že pokiaľ sa "blok komentáru" nachádza či už pred metódou, triedou, alebo nejakou štruktúrou, tak tento komentár sa priradí a bude tykať tejto danej časti kódu.

7.8.4.1 Všeobecné zásady pri písaní dokumentácie:

1. Komentáre písať čo možno stručne nevytvárať v žiadnom prípade literárne diela
2. Komentovať treba každú triedu, metódu (funkciu), zložitejší cyklus, prípadne aj premennú
3. Komentáre písať bez diakritiky a prvé slovo pri opise funkcie a metódy začínať s veľkým písmenom
4. Komentáre treba písať po anglicky pre budúce generácie

7.8.4.2 Komentovanie súborov

1. Rozlišujeme dva druhy súborov:
   - *.h* súbor obsahujúci definície (hlavičky metód, premenné, atď). Je reprezentovaný ako trieda *Class*
   - *.cpp* súbor obsahujúci implementáciu. Pozostáva z viacerých metód (funkcií).
2. Každý takýto súbor by mal obsahovať v hlavičke komentár so základnými informáciami.
3. Hlavička obsahuje popis, ktorý nás informuje čo daný súbor obsahuje a k čomu je určený. Akú časť aplikačnej logiky zastrešuje.

4. Komentár pre doxygen musí byť uzavretý v blokovom komentári. Oproti klasickému komentáru začína **/**** a končí klasickým ***/**
5. Riadky medzi začiatkom a koncom komentára môžu ale nemusia začínať znakom ***

Formát:

```
/**
*@Title nazov suboru
*---------------------------------------------
*@Description
*   [ popis suboru, na čo služi a čo sa v nom rieši ]
*
*@Category  o aky typ suboru ide
*@Author  ak ide o novy subor, tak je vhodne mat meno autora suboru.
*@Verzion
*/
```

7.8.4.3 Komentovanie metód

1. Okomentovať každú metódu. Sprehľadňuje nielen dokumentáciu ale aj zdrojový kód.
2. Stručný popis metódy, aké sú vstupy a čo vracia.

Formát:

```
Pred funkciou
/**
 * @Descripton Popis funkcie, v skratke čo ma robit
 *
 * @param meno_vstunej_premennej na čo služi
 *
 * @see metodaXY() - odvolavka na nejaku inu metodu, bude ako link vygenerovane, nie je nutne
 *
 * @return popis co vracia dana metoda
 */
```

7.8.4.4 Jednoriadkové komentáre

1. V prípade potreby je možné pre doxygen okomentovať aj jednoriadkový kód.
2. Pri definovaní typov môžeme použiť pre ich popis nasledujúci spôsob komentovania.
3. Použije sa komentárová značka **//!**

napr.

```
int var;            //! Stručny popis daneho riadku kodu, premennej, priradenia atd.
```

7.8.4.5 Komentovanie vetvení

1. *Zložitejšie* vetvenia je vhodné vždy komentovať za podmienkou v tom istom riadku

2. Využívať 2x stlačenie tabulátora pre odsadenia, prípadne mať vhodne zarovnanú odsadenú časť s komentárom v rámci celého vetvenia.

Správne:

```
if(podmienka)                 //! Komentar
{
}
else  if(podemienka)          //! Komentar
{
}
else                          //! Komentar
{
}
```

7.8.4.6 Používanie skratiek v komentároch

Už klasicky používame. Pre interné účely určené.

1. TODO: bude značiť niečo čo je potrebné v budúcnosti implementovať.

napr.
```
// TODO:  doplnit  funkcionalitu
```

2. BugID: bude signalizovať že na tomto mieste je známa chýba a ak je zaznačená v nástroji pre manažment zmien tak aj jej ID.

napr.
```
//Bug#12: chyba nespravneho vypisu hodnot
```

7.8.5 Písanie metód

Odporúčania:

1. Snažiť sa programovať krátke a jasné metódy
   - ak je problém rozložený na viacero menších problémov a každý z nich je riešený na samostatnom mieste, je jednoduchšie pochopiť celý problém
2. Odstránenie duplicity kódu
   - zvyšuje kvalitu kódu a prispieva k lepšej udržiavateľnosti
3. Vyhýbať sa tzv. „*mŕtvemu kódu*“
   - taký kód, ktorý je napr. v komentároch, už sa nepoužíva, len tam ostal ako história po predošlých verziách a zneprehľadňuje zdrojový kód