SLOVENSKÁ TECHNICKÁ UNIVERZITA BRATISLAVA
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

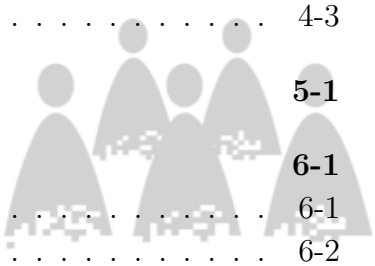
SIMULÁCIA DAVU

Dokumentácia k riadeniu projektu

Bc. Michal Fornádeľ
Bc. Adam Pomothy
Bc. Lukáš Pavlech
Bc. Marek Hlaváč
Bc. Daniel Petráš
Bc. Martin Košický

Vedúci tímového projektu: Ing. Peter Lacko, PhD.
Akademický rok: 2011/12

Obsah



Kapitola 1

Úvod

Tento dokument sa zameriava na riadenie tímového projektu v zložení Bc. Michal Fornádeľ, Bc. Adam Pomothy, Bc. Lukáš Pavlech, Bc. Marek Hlaváč, Bc. Martin Košický a Bc. Daniel Petráš. Dokument obsahuje náležitosti, ktoré boli potrebné z hľadiska vytvorenia koncepcie vnútorného fungovania tímu ako celku.

1.1 Účel a rozsah dokumentu

Účelom dokumentu je vytvoriť ucelený pohľad na spôsob vytvárania inžinierskeho diela a rozdelenie kompetencií. Hlavný dôraz je kladený na plánovanie a korektné manažovanie potrebných náležitostí. Spôsoby plánovania sú pritom prispôsobené agilnému spôsobu vývoja softvéru SCRUM, ktorým sa riadil aj tento timový projekt.

Dokument je výsledkom práce členov tímu v rámci predmetu Tímový projekt na Fakulte informatiky a informačných technológií Slovenskej Technickej Univerzity v Bratislave.

Dokument je určený pre pedagógov zodpovedných za kontrolu a kvalitu vypracovania dokumentu a hlavne správneho a vyváženého spôsobu fungovania tímu.

1.2 Prehľad dokumentu

Kapitola 2 sa zameriava na ponuky, ktoré boli prezentované pri uchádzaní sa o pridelenie projektov. Kapitola obsahuje stručnú charakteristiku jednotlivých členov tímu, znenie ponúk a opis technológií použitých v rámci riešenia. Okrem iného je priložený aj prioritne zoradený zoznam tém tímových projektov, o ktoré sa tím uchádzal.

Kapitola 3 predstavuje plán projeku. Jeho obsahom sú pravidelne revidované dlho-

dobé plány pre zimný a letný semester spolu s jednotlivými šprintami naplánovanými postupne do týždňov, ktoré zahŕňajú konkrétne úlohy spolu s menom zodpovedného člena tímu za ich plnenie a výhradeného času na ich splenenie.

Kapitola 4 prezentuje úlohy členov tíme. V úvode kapitoly sú spomenuté roly členov, ich dlhodobé aj krátkodobé úlohy ako aj to, kto bol autorom jednotlivých kapitol prípadne podkapitol v dokumentoch riadenia a projektu.

Prostriedky technickej podpory, ktoré predstavujú dôležitú úlohe v rámci komunikácie a synchronizácie práce členov tímu sú predmetom kapitoly 5.

Kapitola 6 obsahuje špecifický pohľad na rôzne aspekty projektu z pohľadu oblasti manažmentu, ktorej sa to týka. V tomto prípade sú to riziká, kvalita, rozvrh a plánovanie, podpora vývoja, komunikácia a monitorovanie projektu.

Metodiky potrebné pri vývoji sú rozobraté v kapitole 7. Jedná sa o súbor pravidiel dodržiavaných počas signifikantných procesov počas trvania projektu. Kapitola definuje spôsob písania dokumentácie, zápisníc, zdrojového kódu, testovania a zadávania úloh v systéme Redmine. Príliš rozsiahle metodiky sú presunuté do častí prílohy.

V kapitole 8 sú priložené záznamy zo stretnutí počas jednotlivých týždňov vývoja aplikácie. Zápisnice obsahujú potrebné informácie o stave plnenia navrhovaných úloh a opise činností vykonávaných počas stretnutí.

Kapitola 2

Ponuka

2.1 O nás

Bc. Marek Hlaváč
Absolvent bakalárskeho programu Informatika na Fakulte Informatiky a Informačných Technológií. Vypracoval bakalársku prácu na tému Tréner mentálnych schopností, ktorá spočívala vo vytvorení webového portálu obsahujúceho sadu kognitívnych hier. V súčasnosti pracuje ako herný vývojár pre platformu Flash. Má skúsenosti s programovacími jazykmi ActionScript, Java, C/C++, PHP, JavaScript (jQuery) a databázovým systémom MySQL.

Bc. Michal Fornádeľ
Je absolventom bakalárskeho štúdia na FIIT STU v odbore Informatika a v inžinierskom štúdiu pokračuje v odbore Informačné systémy. Jeho bakalárska práca bola zameraná na vizualizáciu a zhlukovanie fotografických dát v priestore za použitia technológií JOGL a Java3D. V súčasnosti pracuje v spoločnosti Siemens ako softvérový vývojár pre platformu Windows za použitia knižnice MFC. Jeho hlavným zameraním sú programovacie jazyky Java a C++. Má skúsenosti aj s technológiou QT a databázovým systémom Microsoft SQL Server a MySQL.

Bc. Martin Košický
Absolvoval bakalárske štúdium na FIIT STU v odbore Informatika a pokračuje softvérovým inžinierstvom v inžinierskom štúdiu. Ovláda jazyky C/C++, Java, PHP, Flash AS, Python, HTML/HTML5, CSS, SQL na pokročilej úrovni. Súčasne pracuje ako developer vo firme Eset, pričom náplňou jeho práce je implementovať, navrhovať a tiež analyzovať riešenia na rôzne problémy. C++ a Java sú jeho hlavnou oblasťou s tým, že sa aktívne zaujíma o prácu na projektoch s 3D grafikou s OpenGL / Direct3D. Na stránke http://assassin2150.webpark.cz/ je galéria niektorých projektov ktoré robil len pre seba.

Bc. Adam Pomothy

Absolvent bakalárskeho programu Informatika na Fakulte Informatiky a Informačných Technológií. Zameriava sa na výyoj JavaEE aplikácií. Má skúsenosti s vývojom na databázovej, biznis aj front-end vrstve enterprise projektov - konkrétne pracuje s technológiami HTML, CSS, XML, JavaScript, JQuery, Hibernate, IceFaces a PostgreSQL. V súčasnosti pracuje ako softvérový vývojar. Medzi jeho kompetencie patrí aj komunikácia so zákazníkom.

Bc. Lukáš Pavlech

Absolvolval bakalárske štúdium na FIIT STU v odbore Informatika a v inžinierskom štúdiu pokračuje v odbore Informačné systémy. V bakalárskej práci vypracoval tému webovej aplikácie pre hudobný notový zápis. Túto aplikáciu naimplementoval pomocou Java Frameworku Google Web Toolkit na strane klienta a Java Servletov na strane servera. V súčastnej dobe pracuje ako softvérový vývojár vo firme Siemens, kde programuje v jazykoch C++ a JAVA. Súčasne má skúsenosti aj s technológiami ActionScript, PHP, JavaScript, CSS, SQL.

Bc. Daniel Petráš

Absolvent bakalárskeho študijného programu na FIIT STU. Bakalársku prácu vypracoval na tému Systém pre správu programátorských maratónov. Na pokročilej úrovni ovláda jazyky C/C++, C#, Java, PHP, XML, HTML/CSS, (My,MS)SQL, Android (java). Má skúsenosti s prácou s .NET, WinAPI, Windows Driver Device Kit (DDK) a OpenGL. Pracoval/pracuje na projektoch ako XML Printer, MKMB, GeoEmpires. Zastáva pozíciu kde navrhuje a implementuje riešenia problémov a podieľa sa aj na návrhu architektúry systému.

2.2 Znenie jednotlivých ponúk

2.2.1 Znalosti a zručnosti študentov

Motivácia

V dnešnej dobe si mnohé spoločnosti nechcú dávať inzeráty na portáloch ako je napr. profesia.sk. Často sa im z nich hlásia ľudia, ktorí v skutočnosti nespĺňajú ich požiadavky aj keď to o sebe tvrdia. Riešením tejto situácie by mohol byť systém na správu znalostí a zručností študentov. Vďaka nemu by škola mohla odporučiť firmám tých pravých ľudí (nás) a jednoducho by našla správnych ľudí na rôzne projekty, stáže atď. Systém by pomohol aj učiteľom pri hľadaní študentov na rôzne projekty. Väčšia spoľahlivosť by bola zaručená faktom, že by každý študent mal v profile nie len údaje, ktoré vyplnil on sám, ale aj údaje zadané vyučujúcim. Tým sa radikálne zvýši dô-

veryhodnosť zadaných informácií a zároveň atraktivita databázy pre firmy hľadajúce nových zamestnancov.

Toto zadanie sa od ostatných líši najmä svojou praktickosťou. Nie je zamerané na výskum a nepatrí do vedeckej sféry. Implementácia projektu zahŕňa problematiku, ktorá je bežná v komerčnej informatike a projekt teda znamená značný prínos pre fakultu ako aj pre budúci profesionálny život riešiteľov.

Koncepcia riešenia
Riešenie tohto problému si predstavujeme ako webovú aplikáciu ktorá by ponúkala rôzne možnosti práce pre jednotlivých používateľov. Táto aplikácia by ponúkala tieto rozhrania:

- rozhranie pre učiteľov / administratívnych pracovníkov (napr. študijné oddelenie) - bude slúžiť na zadávanie, vyhľadávanie a filtrovanie údajov v systéme. Zadávanie údajov bude možné buď:
  - jednotlivo - zadanie zručnosti len pre jedného študenta
  - hromadne
    - vybranie niekoľkých študentov a priradenie/úprava určitej zručnosti
    - nahranie súboru študentov, v ktorom sa priraďuje určitá zručnosť

  Učitelia budú mat možnosť dotvárať systém - dopĺňať jednotlivé zručnosti, zaraďovať ich do skupín a podobne.
- rozhranie pre študentov - bude slúžiť pre študentov ako možnosť zverejnenia svojich schopností za účelom "zverejnenia sa".
- rozhranie pre správcu - bude slúžiť na úpravu "statického" obsahu aplikácie (novinky, vysvetľujúce texty, návody) a najmä na sledovanie audit trailu.

Základ tohto projektu vidíme v dobrom návrhu databázovej vrstvy, nakoľko je aplikácia založená najmä na ukladaní a následnom spracovaní dát. Hlavnú časť biznis/aplikačnej logiky (pod čím sa myslí správne roztriedenie, zaradenie a spracovanie dát) by sme chceli implementovať na databázovej úrovni. Vďaka tomu by sa zvýšila robustnosť aj efektívnosť aplikácie. Veľký dôraz je potrebné klásť najmä na reprezentáciu samotných zručnosti študentov. Každá zručnosť by bola popísaná niekoľkými druhmi "zaradenia" kvôli neskoršej filtrácií. Aby bola aplikácia flexibilná, učiteľom by bolo umožnené pridávať nové typy zručností a ich klasifikáciu. Tie by následne mohli aj používať pri prideľovaní.

Databáza študentov by sa dala rozširovať importovaním dát z xls alebo xml súboru.

Pre zvýšenie použiteľnosti je potrebné poskytnúť aj exportovanie dát a to do viacerých formátov. Pre školské účely by bolo vhodné implementovať aj rôzne štatistiky (aké percento študentov má skúsenosti s určitou technológiou atď.), ktoré by sa následne podľa potreby dali aj vizualizovať (graf). Vďaka tomu by škola mohla pružnejšie reagovať pri tvorbe nových predmetov, ktoré by boli obľúbené v radoch študentov.

Technológie
Na tvorbu dátovej časti by sa mohol použiť veľmi populárny databázový systém PostgreSQL.

Front-end aplikácie by mohol byt zrealizovaný napr. pomocou frameworku GWT. Ten dovoľuje pomerne rýchlo vytvoriť veľmi atraktívnu aplikáciu. Ďalšou výhodou tohto JAVA frameworku je možnosť debugovania klientskej časti aplikácie, z ktorej sa po rekompilácií stáva JavaScript spolupracujúci s HTML5. Alternatívou sú frameworky ako IceFaces, Spring alebo JSF.

2.2.2 Štatistický preklad textu

Motivácia
Hlavný dôvod záujmu o tému spočíva v jej komplexnosti (morfológia, syntax a takisto aj sémantika sú často veľkou prekážkou pri preklade dvoch štruktúrou pomerne odlišných jazykov). Štatistický preklad voľného textu pokrýva viaceré oblasti softvérového vývoja – webové spektrum, dolovanie dát (data mining), efektívnu algoritmizáciu, štatistiku a prepája ich do jedného celku. Jednotliví členovia tímu už majú skúsenosti v týchto oblastiach, takže je pre nás téma veľmi atraktívna a v konečnom dôsledku sme všetci schopní prispieť určitou časťou a celkové úsilie tak rozdeliť homogénne do všetkých oblastí projektu.

Ďalším dôvodom pre výber témy je značná voľnosť pri výbere konkrétnych technológií použitých pri realizácií projektu a teda je možnosť prispôsobiť tento výber preferenciám členov tímu. Na druhej strane sme otvorený rozšíreniu si prehľadu o nové technológie, ktoré majú do budúcna určitý potenciál a tým pádom sa ich aspoň čiastočná znalosť stáva pre nás cennou.

V neposlednom rade je nutné podotknúť, že prekladače textov sú pre členov nášho tímu pri čítaní cudzojazyčnej literatúry neoddeliteľnou súčasťou a teda by sme boli nadšení posunúť vývoj v tejto oblasti o krok ďalej. I keď sa môže zdať, že vývoj prekladačov textu ubehol už dlhý kus cesty, tak v slovenských vodách nám podobný nástroj ešte stále chýba a v prípade výberu nášho tímu pre tento projekt sme ochotní ho v plnej miere vyplniť.

Koncepcia riešenia

Základom pri (pomerne) úspešnom preklade je vzájomná príbuznosť dvojice jazykov. Angličtina, nemčina, francúzština alebo španielčina patria do tejto skupiny, pretože sa charakterovo zhodujú napríklad v absencii ohýbania alebo pevnom slovoslede. Situácia sa stáva zaujímavou pri skupine jazykov, ktoré sú odlišné. V takomto prípade je vhodné navrhnúť prístupy, ktoré by do úvahy brali spôsob štruktúrovania jazyka a zameriavali sa na jeho špecifiká. Navrhované riešenie by spočívalo v návrhu webovej aplikácie, ktorá by jednoduchým a hlavne intuitívnym spôsobom ponúkala používateľovi priestor pre preklad vybranej webovej stránky. Ideálnym spôsobom by bol import funkcionality prekladu do rôznych prehliadačov napríklad v podobe pluginu, ktorý by presmerovával používateľa na odkaz preloženej kópie stránky, prípadne prekladal stránku "priamo na mieste". Ponúkané riešenie sa rozdeľuje do troch hlavných častí:

- Návrhu vizuálneho prevedenia (front-endu) aplikácie
- Návrhu prekladača
- Optimalizácie riešenia pre "real-time" prekladanie

Návrh vizuálneho prevedenia by obsahoval graficky atraktívne rozhranie pre používateľa. Po zadaní adresy prekladanej stránky (alebo automatickom presmerovaní priamo zo stránky požadovanej na preklad) alebo "iba" bloku textu by zaujímavým riešením mohlo byť aj napríklad dvoj-panelové zobrazenie (známe napr. z programov Total Commander, WinMerge), kde by v prvom paneli bola zobrazená originálna stránka (príp. text) a napravo by bol jej preklad. Webová aplikácia by sa mohla takisto zameriavať aj na grafické zvýrazňovanie textu podľa toho či napríklad vznikli určité problémy pri preklade poprípade by mohla ponúknuť rôzne alternatívne výrazy pri označení príslušného textu.

Okrem vizuálneho prevedenia sa hlavnou úlohou stáva návrh prekladača. V súčasnosti existujú viaceré prístupy spôsobu prekladu. Okrem využívania pravdepodobnosti pri jednotlivých dvojiciach alebo n-ticiach slov je zaujímavým riešením aj implementovanie trénovacích metód.

Pri prekladaní by bolo vhodné použiť postupne kombináciu používaných prekladacích metód:

- Doslovný preklad - slová sú prekladané do jedného alebo viacerých slov
- Preklad založený na frázach - preklad sekvencií slov alebo blokov slov
- Syntaktický preklad - preklad syntaktických jednotiek vety pomocou stromov so syntaktickou štruktúrou (derivačné stromy)

Medzi problémy, ktoré by potrebovali pri prekladaní špeciálnu pozornosť patrí:

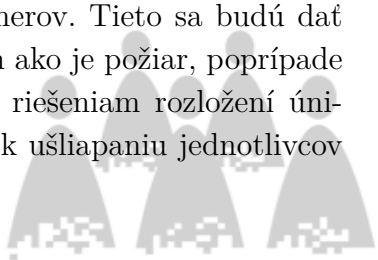
- Problém priradenia viet - viaceré vety môžu byť preložené na viacej spôsobov
- Zložené slová
- Frázy - slovné frázy sa dajú preložiť viacerými spôsobmi vzhľadom na kontext použitia
- Pomiešaný slovosled
- Neznáme slová - slová existujúce a používané mimo slovník
- Syntax

2.2.3 Simulovanie davu

Motivácia
Uvedomenie si potreby vedieť dopredu odhadnúť ako sa bude správať dav ľudí je dôležité napríklad kvôli návrhu únikových trás, veľkosti únikových východov, naplánovaniu prechodu davu cez mesto alebo inú zastavanú oblasť. Je zaujímavé sledovať, ako sa dav správa v neštandardných situáciách, pretože reálna simulácia s takým počtom ľudí nie je vo väčšine prípadov možná. Na téme je takisto zaujímavé využitie GPU na urýchlenie výpočtov, tak aby mohla simulácia prebiehať v reálnom čase, alebo aby sa k tomu aspoň priblížila.

Koncepcia riešenia
Riešenie je predstavované ako aplikácia, ktorá bude schopná realisticky simulovať správanie sa masy ľudí v ľubovoľnom prostredí. Aplikácia bude schopná v reálnom čase odsimulovať správanie až niekoľko tisíc ľudí v prostredí, ktoré sa bude dať jednoducho nadefinovať. Tieto prostredia sa rozlišujú na uzavreté a otvorené.

Uzatvorené prostredia budú poväčšinou budovy, poprípade arény ako napríklad futbalové ihriská, alebo to budú interiéry budov väčších rozmerov. Tieto sa budú dať použiť na simulovanie správania ľudí v rizikových situáciách ako je požiar, poprípade iný poplach. Výsledok týchto simulácii môže dopomôcť k riešeniam rozložení únikových východov a minimalizovať situácie, kedy dochádza k ušliapaniu jednotlivcov davom.



Otvorené prostredia budú spravidla ulice v mestách. Tu má zmysel simulovať väčšie počty ľudí. Simulácia môže dopomôcť policajným zložkám potlačiť rozbúrený dav, rabovanie a iné situácie. Simulácia sa bude približovať reálnemu času a na konci simulácie sa bude dať získať kompletná štatistika o počte zachránených ľudí, ušliapaných

ľudí, o počte zranených policajtov. Súčasťou tohto výpisu budú tiež nákresy slabých miest.

Hlavnou myšlienkou je vytvoriť tento systém modulárnym spôsobom a to tak, že správanie ľudí bude implementované agentom a bude sa dať pridávať v podobe zásuvných modulov. Použitie by sa dalo rozšíriť napríklad o simulovanie protiteroristického zásahu obkľúčených, poprípade iné situácie ktoré si vyžadujú plánovanie pohybu ľudí v určených priestoroch.

Riešenie bude spustiteľné na platformách Windows ako aj na Unix a bude využívať do najväčšej miery hardvérové možnosti stroja, na ktorom bude aplikácia bežať. Pokiaľ to bude stroj podporovať, budú výpočty presmerované na GPU pre najvyšší možný výkon. V opačnom prípade sa presmerovanie neudeje. Používateľovi to preto nekladie prekážky pri použití rôznych druhov hardvéru. Aplikácia bude z tohto titulu praktickejšia a jednoduchšie použiteľná.

Použité technológie Pri riešení tímového projektu boli použité nasledovné technológie:

- Vývojové prostredie Visual Studio 2010
- Programovací jazyk C++ rozšírený o MPI a OpenMP na distribuované počítanie
- Project managment system Redmine
- Verziovací systém SVN
- AnkhSVN ako integračný modul do Visual Studia pre potreby manažovania súborov v ramci SVN
- Redakčný systém Joomla 1.5

2.3 Zoradenie tém podľa priority

Pri uchádzaní sa o jednotlivé témy tímových projektov prebehlo vytvorenie zoznamu zoradeného podľa priority. Témy boli zoradené na spoločnom stretnutí, kde jednotliví členovia postupne vyjadrovali svoj názor na dostupné témy. Po vzájomnej diskusii bol vytvorený spoločný zoznam, ktorý je uvedený v tabuľke 2.1.

Poradie	**Názov témy**
1	Znalosti a zručnosti študentov

2	Štatistický preklad voľného textu
3	Simulácia davu
4	Webový editor pre TeX
5	3D UML
6	Personalizované odporúčanie
7	Plagiáty na webe
8	Textový editor obohatený o grafické prvky
9	Inteligentná hra pre mobilné zariadenia
10	Tvorba "ľahko"sémantického obsahu pre adaptívny webový (výučbový) portál
11	Digitálne divadlo
12	Rozvrhový systém novej FIIT
13	Editovanie viacrozmerného grafu prepojenia informácií v dokumentoch
14	RoboCup - tretí rozmer
15	Virtuálna FIIT
16	Osobný manažment fyzickej aktivity pomocou mobilných zariadení
17	Imagine Cup 2012: Game Design

Tabuľka 2.1: Zoradenie všetkých tém podľa preferencie

2.4 Spoločný rozvrh

Táto podkapitola obsahuje spoločný rozvrh odzrkadľujúci vyťaženie členov tímu v zimnom semestri. Podľa spoločného rozvrhu bol upresnený termín stretnutia realizovaného každý týždeň. V tabuľke 2.2 sú uvedené skratky predmetov spolu s ich oficiálnymi menami.

Skratka predmetu	**Názov predmetu**
AIS	Architektúry inf. systemov
AOP	Aspektovo orientované programovanie
ASS	Architektúry soft. systémov
DD	Dejiny dizajnu
MPSIS	Manažment projektov soft. a inf. systémov
NS	Neurónové siete
OOANS	Objektovo orientovaná analýza a návrh systému
PDT	Pokročilé databázové technológie
SU	Strojové učenie

TP	Tímový projekt
VI	Vyhľadávanie informácií
VIS	Výskum inf. systémov
VSS	Výskum soft. systémov

Tabuľka 2.2: Skratky predmetov spolu s ich menami

		7.00-7.50	8.00-8.50	9.00-9.50	10.00-10.50	11.00-11.50	12.00-12.50	13.00-13.50	14.00-14.50	15.00-15.50	16.00-16.50	17.00-17.50	18.00-18.50	19.00-19.50
Po	Michal Fornádeľ								SU		TP		VIS	
	Marek Hlaváč							OOANS	SU				VSS	
	Martin Košický												VSS	
	Lukáš Pavlech								SU				VIS	
	Daniel Petráš		VI			VI							VSS	
	Adam Pomothy								SU				VIS	
Ut	Michal Fornádeľ									MPSIS		MPSIS		
	Marek Hlaváč													
	Martin Košický	Kódovanie												
	Lukáš Pavlech													
	Daniel Petráš			NS		NS								
	Adam Pomothy													
St	Michal Fornádeľ		TP											
	Marek Hlaváč										OOANS			
	Martin Košický					AOP								
	Lukáš Pavlech													
	Daniel Petráš										DD		DD	
	Adam Pomothy													
Št	Michal Fornádeľ								PDT		AIS			
	Marek Hlaváč									ASS				
	Martin Košický	Kódovanie								ASS			AOP	
	Lukáš Pavlech					SU			PDT		AIS			
	Daniel Petráš									ASS				
	Adam Pomothy			PDT		SU					AIS			
Pi	Michal Fornádeľ			PDT										
	Marek Hlaváč													
	Martin Košický													
	Lukáš Pavlech			PDT										
	Daniel Petráš													
	Adam Pomothy			PDT										

Kapitola 3

Plán projektu

Kľúčovou úlohou pre zabezpečenie progresívnej a správne sa uberajúcej práce počas zimného a aj nadchádzajúceho letného semestra bolo vypracovanie dlhodobých plánov. V rámci tímu bol plán pre zimný semester raz za mesiac aktualizovaný, aby odrážal aktuálny stav projektu. Kapitola okrem toho obsahuje aj plán jednotlivých šprintov spolu s úvodným rozbehom počas dvanástich týždňov zimného semestra. V šprintoch sú zobrazené iba úlohy, pričom podrobnosti o jednotlivých úlohách ako dobra trvania splnenia je uvedená v zápisoch zo stretnutí a systéme Redmine.

Dlhodobý plán - zimný semester

Tabuľka 3.1: Plán projektu (28.9.2011)

2. - 4. týždeň	Oboznámenie sa s problematikou simulovania davu, stanovenie používaných technológií, definovanie vývojového prostredia, systému pre plánovanie, spôsobu verziovania zdrojových súborov, analyzovanie dostupných riešení a možností pre integráciu s budúcim vyhotoveným riešením
5. - 6. týždeň	Vytvorenie základného prototypu aplikácie, spracovanie metodiky pre písanie zdrojových kódov, vytvorenie prvotnej vizualizácie
7. - 8. týždeň	Implementácia pohybujúcich sa agentov, načítavanie mapy zo vstupného súboru
9. - 10. týždeň	Vytvorenie základnej fyzikálnej simulácie, riešenie kolízií agenta so stenami a s inými agentami
11. - 12. týždeň	Vytvorenie prototypu s agentami, ktorí majú základnú inteligenciu

Tabuľka 3.2: Plán projektu (28.10.2011

6. týždeň	Vytvorenie základného prototypu aplikácie, definovanie spôsobu komunikácie agentov s prostredím, definovanie štruktúry prenášaného protokolu, návrh základnej funkcionality agenta a prostredia, spracovanie metodiky pre písanie zdrojových kódov, vytvorenie prvotnej vizualizácie
7. - 8. týždeň	Prepracovanie projektu do koncepcie samostatných DLL modulov, vytvorenie štruktúry použitej mapy, riešenie spôsobu hľadania dverí a plánovania pohybu zo strany agenta, detekcia kolizií agentov a pretínajúcich sa stien, simulovanie pohľadu agenta
9. - 10. týždeň	Vytvorenie základnej fyzikálnej simulácie, riešenie kolízií agenta so stenami a s inými agentami
11. - 12. týždeň	Vytvorenie prototypu s agentami, ktorí majú základnú inteligenciu

Tabuľka 3.3: Plán projektu (28.11.2011

9. - 10. týždeň	Vytvorenie základnej fyzikálnej simulácie, vytvorenie navigácie cez navigačnú geometriu
11. - 12. týždeň	Vytvorenie implementácie vnútornej mapy na základe navigačnej geometrie, implementovanie FlowField algoritmu pre pohyb agentov

Dlhodobý plán - letný semester

Tabuľka 3.4: Plán projektu (28.9.2011)

1. - 2. týždeň	Vytvorenie novej sady máp, analýza steering behaviours, pridanie ďalšej úrovne do rozhodovacieho automatu agenta
2. - 4. týždeň	Integrácia flowfield do hlavného projektu, analýza možností paralelizácie aplikácie a jej vykonanie, doplnenie náhodnosti do rozhodovania agentov, pridať tretí rozmer, zlepšiť vizuálnu stránku aplikácie
4. - 6. týždeň	Preskúmať možnosti používania mapy z projektu Virtuálnej FIIT, návrh optimalizácie, záťažové testy aplikácie (veľký počet agentov)
6. - 8. týždeň	Analýza možností simulácie v mapách vonkajšieho prostredia, vykonať navrhnutú optimalizáciu

9. - 10. týždeň	Vytvorenie zaujímavých máp a situácií
11. - 12. týždeň	Experimentovanie s parametrami aplikácie pre dosiahnutie čo najreálnejšej simulácie, posledné úpravy aplikácie, úprava dokumentácie

Rozbeh

Názov úlohy	**Zodpovedný**	**Interval vykonávania**
Naštudovanie problematiky simulácie davu	Všetci	28.9. - 12.10.
Analýza podobných riešení	Marek Hlaváč, Adam Pomothy, Lukáš Pavlech, Martin Košický	28.9. - 12.10.
Rozbehanie RedMine systému	Michal Fornádeľ	1.10. - 3.10.
Rozbehanie SVN	Adam Pomothy	4.10.
Rozbehanie servera a web stránky	Lukáš Pavlech	29.9.
Vytvorenie šablóny pre dokumentáciu	Michal Fornádeľ	8.10.
Návrh architektúry prvého prototypu	Všetci	28.9. - 12.10.
Prvotná analýza mapových formátov a ich možnosti použitia	Marek Hlaváč	10.10.

Šprint#1

Názov úlohy	**Zodpovedný**	**Interval vykonávania**
Základná vizualizácia	Martin Košický	12.10. - 26.10.
Protokol agentových akcií a akcií prostredia	Daniel Petráš	12.10. - 26.10.
Komunikácia prostredia a agentov	Daniel Petráš	12.10. - 26.10.
Základná funkcionalita agenta	Michal Fornádeľ	12.10. - 26.10.
Code Guidelines	Adam Pomothy	12.10. - 26.10.
Zapracovanie analyzovaných riešení do šablóny dokumentu riadenia	Michal Fornádeľ	12.10. - 26.10.
Integrovanie SVNka s vývojovým prostredím	Adam Pomothy	12.10. - 26.10.
Zistenie formátu mapy v projekte VirtFIIT	Lukáš Pavlech	12.10. - 26.10.
Nájdenie nástroja pre konverziu OSM -> CAD	Marek Hlaváč	12.10. - 26.10.
Základná implementácia mapy	Marek Hlaváč	12.10. - 26.10.

Analýza formátov máp	Marek Hlaváč	12.10. - 26.10.
Základná funkcionalita prostredia	Lukáš Pavlech	12.10. - 26.10.

Šprint#2

Názov úlohy	**Zodpovedný**	**Interval vykonávania**
Simulovanie pohľadu agenta	Martin Košický, Michal Fornádeľ	26.10. - 9.11.
Hľadanie dverí zo strany agenta	Adam Pomothy	26.10. - 9.11.
Plánovanie pohybu agenta	Marek Hlaváč	26.10. - 9.11.
Integrácia vyhotovenej dokumentácie do šablóny dokumentu riadenia	Michal Fornádeľ	26.10. - 9.11.
Vytvorenie koncepcie oddelených modulov (DLL knižnice)	Martin Košický	26.10. - 9.11.
Detekcia pretínajúcich sa stien	Daniel Petráš	26.10. - 9.11.
Riešenie kolízií agentov	Lukáš Pavlech	26.10. - 9.11.
Generovanie agentov do mapy	Lukáš Pavlech	26.10. - 9.11.
Vytvorenie definovanej mapy a doplnenie vrstvy pre reprezentáciu dverí	Marek Hlaváčh	26.10. - 9.11.

Šprint#3

Názov úlohy	**Zodpovedný**	**Interval vykonávania**
Prenos generovania agentov do nového projektu	Lukáš Pavlech	9.11. - 23.11.
Prenos máp do nového projektu	Marek Hlaváč	9.11. - 23.11.
Prenos základnej funkcionality agnetov	Lukáš Pavlech	9.11. - 23.11.
Návrh plánovania rozhraní pohybu agentov	Martin Košický	9.11. - 23.11.
Detekcia agenta prechádzajúceho cez stenu	Daniel Petráš	9.11. - 23.11.
Monitorovanie a integrácia implementačných riešení	Michal Fornádeľ	9.11. - 23.11.
Analýza a návrh Flowfield	Michal Fornádeľ, Adam Pomothy	9.11. - 23.11.
Kontrola dosiahnutia výstupného stavu agenta	Adam Pomothy	9.11. - 23.11.
Riešenie kolízií agentov	Lukáš Pavlech	9.11. - 23.11.
Implementácia časovania prostredia	Daniel Petráš	9.11. - 23.11.

Vykreslenie načítanej mapy	Martin Košický	9.11. - 23.11.
Zdokumentovanie DLL	Martin Košický	9.11. - 23.11.
Distribúcia mapy modulom	Martin Košický	9.11. - 23.11.
Prihláška do TP cupu	Marek Hlaváč	9.11. - 23.11.

Šprint#4

Názov úlohy	**Zodpovedný**	**Interval vykonávania**
Návrh koncepcie ovplyvňovania pohybu agentov na základe hustoty výskytu ostatných agentov	Michal Fornádeľ, Adam Pomothy	23.11. - 7.12.
Vytvorenie vrstvy pre NavigationMesh do mapy	Marek Hlaváč	23.11. - 7.12.
Programová reprezentácia NavigationMesh	Marek Hlaváč	23.11. - 7.12.
Implementácia návrhu stavového automatu pre agenta	Martin Košický	23.11. - 7.12.
Vytvorenie kompletnej cesty agenta do cieľa na zaklade NavigationMesh	Lukáš Pavlech	23.11. - 7.12.
Generovanie pohybov agenta na základe naplánovanej cesty	Daniel Petráš	23.11. - 7.12.
Integrácia existujúcej dokumentacie	Michal Fornádeľ, Adam Pomothy	23.11. - 7.12.
Zosúladenie zápisnic zo stretnutí a doplnenie ID-čiek z Redminu	Michal Fornádeľ	23.11. - 7.12.
Úprava a zosúladenie zápisníc zo stretnutí	Adam Pomothy	23.11. - 7.12.
Vykresľovanie vnútorných stien	Martin Košický	23.11. - 7.12.
Pridanie polygónov reprezentujúcich východy do NavigationMesh	Marek Hlaváč	23.11. - 7.12.

Kapitola 4

Úlohy

4.1 Identifikované roly členov

V úvode semestra prebehlo rozdeľovanie rolí medzi jednotlivých členov tímu. Rozdelenie prebehlo počas ústnej dohody na základe preferencií a požadovanej práce. Členovia tímu si úlohy vyberali podľa svojich doterajších skúseností a zručností. Rozdelenie úloh v zimnom semestri je uvedené v tabuľke 4.1.

Člen tímu	**Rola v tíme**
Marek Hlaváč	Manažér monitorovania projektu
Martin Košický	Manažér plánovania
Michal Fornádeľ	Manažér rizík, zástupca vedúceho tímu
Adam Pomothy	Manažér kvality
Lukáš Pavlech	Vedúci tímu
Daniel Petráš	Manažér podpory vývoja

Tabuľka 4.1: Rozdelenie úloh počas zimného semestra

4.2 Krátkodobé úlohy

V tabuľke 4.2 sú uvedené krátkodobé jednorázové úlohy, ktoré boli identifikované počas stretnutí alebo vyplynuli z dôvodu riešenia inej úlohy.

Úloha	**Pomothy**	**Petráš**	**Košický**	**Pavlech**	**Fornádeľ**	**Hlaváč**
Vytvorenie webovej stránky (aj vzhľad)	-	-	-	•	•	-
Vytvorenie google skupiny	-	-	-	•	-	-

Vytvorenie plagátu tímu	-	-	-	-	●	-
Vytvorenie šablóny pre zápisy	●	-	-	●	●	-
Vytvorenie šablóny pre dokumentácie	-	-	-	-	●	-
Vytvorenie návodu Visual Assist	-	-	-	●	-	-
Vytvorenie návodu pre Boost	-	●	●	-	-	-
Vytvorenie návodov pre Google Test	-	-	●	-	-	-
Vytvorenie návodu na používanie SVN	●	-	-	-	-	-
Vytvorenie návodu na používanie pluginu AnkhSVN	●	-	-	-	-	-

Tabuľka 4.2: Zoznam krátkodobých úloh

4.3 Dlhodobé úlohy

V tabuľke 3.4 sú uvedené dlhodobé úlohy v time spolu s percentuálny vyjadrením podielu práce konkrétnych členov na danej činnosti.

Úloha	**Pomothy**	**Petráš**	**Košický**	**Pavlech**	**Fornádeľ**	**Hlaváč**
Webová stránka	-	-	-	●	-	-
Systém Redmine	-	-	-	-	●	-
Prepis do Latex-u	●	-	-	-	●	-
Implementácia	●	●	●	●	●	●

4.4 Autorstvo

Táto podkapitola sumarizuje autorov jednotlivých kapitol požadovanej dokumentácie ako aj zápisníc zo stretnutí tímového projektu.

4.4.1 Zápisnice

Názov	Autor
Šablóna zápisnice	Lukáš Pavlech
Zápisnica č.1	Lukáš Pavlech
Zápisnica č.2	Adam Pomothy
Zápisnica č.3	Marek Hlaváč
Zápisnica č.4	Daniel Petráš
Zápisnica č.5	Martin Košický
Zápisnica č.6	Michal Fornádeľ

4.4.2 Dokumentácia riadenia

Kapitola	Názov	Autor
	Štylistické a gramatické korektúry	Michal Fornádeľ, Adam Pomothy
1	Úvod	Michal Fornádeľ
2.1	O nás	Všetci
2.2.1	Znalosti a zručnosti študentov	Lukáš Pavlech, Adam Pomothy
2.2.2	Štatistický preklad textu	Michal Fornádeľ, Marek Hlaváč
2.2.3	Simulovanie davu	Martin Košický, Daniel Petráš
2.3	Zoradenie tém podľa priority	Všetci
2.4	Spoločný rozvrh	Marek Hlaváč
3	Plán projektu	Michal Fornádeľ,Adam Pomothy
4	Úlohy	Michal Fornádeľ
5	Prostriedky technickej podpory	Michal Fornádeľ
6	Metodiky potrebné pri vývoji	Michal Fornádeľ
6	Manažment projektu	Všetci
7	Metodiky potrebné pri vývoji	Michal Fornádeľ
7	Záznamy zo stretnutí	Všetci
A	Metodika vytvárania zdrojového kódu	Adam Pomothy
B	Metodika pre jednotkové testovanie	Adam Pomothy
C	Metodika pre zadávanie úlohy v systéme Redmine	Michal Fornádeľ

4.4.3 Projektová dokumentácia

Kapitola	Názov	Autor
	Štylistické a gramatické korektúry	Michal Fornádeľ, Adam Pomothy
1	Úvod	Michal Fornádeľ
2.1	PathFinder	Marek Hlaváč
2.2	PedGo	Adam Pomothy
2.3	Fire Dynamics Simulator and Smokeview	Lukáš Pavlech
2.4	SimWalk	Martin Košický
3.1	Základná vizualizácia	Martin Košický
3.2	Protokol agentových akcií a akcií prostredia	Daniel Petráš
3.3	Komunikácia prostredia a agentov	Daniel Petráš
3.4	Základná funkcionalita agenta	Michal Fornádeľ
3.5	Vypracovanie metodiky Code Guidelines	Adam Pomothy
3.6	Zapracovanie analyzovaných riešení do šablóny dokumentu riadenia	Michal Fornádeľ
3.7	Integrovanie systému SVN s vývojovým prostredím	Adam Pomothy
3.8	Zistenie formátu mapy v projekte VirtFIIT	Lukáš Pavlech
3.9	Nájdenie nástroja pre konverziu OSM -> CAD	Marek Hlaváč
3.10	Základná implementácia mapy	Marek Hlaváč
3.11	Analýza formátov máp	Marek Hlaváč
3.12	Základná funkcionalita prostredia	Lukáš Pavlech
4.1	Simulovanie pohladu agenta	Martin Košický
4.2	Hľadanie dverí zo strany agenta	Adam Pomothy
4.3	Plánovanie pohybu agenta	Marek Hlaváč
4.4	Integrácia vyhotovenej dokumentácie do šablóny dokumentu riadenia	Michal Fornádeľ
4.5	Vytvorenie koncepcie oddelených modulov (DLL knižnice)	Martin Košický
4.6	Detekcia pretínajúcich sa stien	Daniel Petráš
4.7	Riešenie kolízií agentov	Lukáš Pavlech
4.8	Generovanie agentov do mapy	Lukáš Pavlech
5.1	Kontrola dosiahnutia výstupného stavu agenta	Adam Pomothy
5.2	Riešenie kolízii agentov	Lukáš Pavlech
5.3	Prenos generovania agentov do nového projektu	Daniel Petráš

5.4	Prenos základnej funkcionality agentov do nového projektu	Lukáš Pavlech
5.5	Zdokumentovanie DLL	Martin Košický
5.6	Prenos máp do nového projektu	Marek Hlaváč
5.7	Distribúcia mapy modulom	Martin Košický
5.8	Návrh plánovania rozhraní pohybu agentov	Martin Košický
5.9	Implementácia časovania prostredia	Martin Košický
5.10	Detekcia agenta prechádzajúceho cez stenu	Daniel Petráš
6.1	Aplikovanie hustoty na model	Adam Pomothy
6.2	Vytvorenie kompletnej cesty agenta do ciela na základe NavigationMesh	Lukáš Pavlech
6.3	Vytvorenie vrstvy pre NavigationMesh do mapy	Marek Hlaváč
6.4	Programová reprezentácia NavigationMesh	Marek Hlaváč
6.5	Generovanie pohybov agenta na základe naplánovanej cesty	Daniel Petráš
7.1	Architektúra prototypu	Martin Košický
7.1.1	Komunikácia	Adam Pomothy
7.1.2	Prostredie	Adam Pomothy
7.1.3	Agent	Adam Pomothy
7.1.4	Mapa	Marek Hlaváč

Kapitola 5

Prostriedky technickej podpory

Pre vývoj aplikácie sa centrálnym miestom stal školský server bežiaci pod operačným systémom Ubuntu. Do neho boli následne nainštalované a nakonfigurované prostriedky pre podporu kolaborácie a komunikácie. Medzi základné prostriedky patria:

- **Redmine** - systém pre plánovanie sa stal základom pri manažovaní úloh v projekte . Je schopný jednoduchým a hlavne prehľadným spôsobom prinášať ucelený prehľad o stave plnenia jednotlivých požiadavok okrem iných aj pre agilný spôsob vývoja softvéru SCRUM (Ganttove diagramy, plánovanie šprintov). Podporuje integráciu SVN repozitárov, vďaka čomu je možné v jednom systéme mať možnosť okrem podporných plánovacích nástrojov aj možnosť nahliadnutia do zdrojových kódov aplikácie. Pri akýchkoľvek vykonaných zmenách sú zainteresovaní členovia okamžite notifikovaní prostredníctvom e-mailu.

- **Subversion a TortoiseSVN** - Subversion je verziovací systém a TortoiseSVN je grafický klient pre tento systém, ktorý je používaný každým členom tímu na pracovnom stroji, ktorý používa. TortoiseSVN predstavuje jednoduchú aplikáciu umožňujúcu prostredníctvom grafického používateľského rozhrania manažovať verzie zdrojových súborov umiestnených na serveri, kde je nakonfigurovaná SVN podpora. V rámci prostredia Visual studio 2010, ktoré je odporúčané pre používanie v rámci tohto tímu je možné doinštalovať plugin AnkhSVN, vďaka čomu je možné súbory verziovať priamo.

- **Google Groups** - podporuje používateľské skupiny. Jej hlavným prínosom je spoločný mailový priestor pre všetkých členov tímu a takisto aj podpora fóra pre prípadné príspevky.

- **Google Docs** - centrálne úložisko dokumentov potrebných pre tim, ktoré obsahuju najmä potrebné informácie pre prihlasovanie sa do systému, spôsoby a návody pre konfigurácie a v neposlednom rade aj odkazy na zaujímavú odbornú literatúru pre potreby simulácie davu.

- **Google Talk** - hlavným prostriedkom pre hlasovú komunikáciu. Vďaka tomuto nástroju sú v prípade potreby členovia okamžite schopní vytvoriť hlasové konferencie.

Kapitola 6

Manažment projektu

Na základe role každého člena tímu identifikovanej na začiatku semestra bol projekt analyzovaný z rôznych pohľadov, ako sa dá manažovať a čo všetko je nutné pre zabezpečenie konkrétnej oblasti manažmentu vykonať. Do úvahy sa brali nasledovné oblasti:

- Manažment kvality
- Manažment rizík
- Manažment rozvrhu a plánovanie
- Manažment podpory vývoja
- Manažment monitorovania projektu
- Manažment komunikácie

6.1 Manažment kvality

Predpokladom zabezpečenia plynulého chodu projektu a eliminovania oneskorených odovzdávok je zadefinovanie procesov, ktoré pomocou merateľných faktorov určujú, ktoré časti projektu sa dajú pokladať za kvalitne prevedené a ktoré nie. Táto oblasť sa nazýva manažment kvality. V rámci timového projektu boli primárne uvažované nasledovné aspekty:

- **Code guidelines**
  Na zaistenie čo najväčšej kvality bola vytvorená príručka – súbor pravidiel na vytváranie zdrojového kódu (po anglicky code guidelines). Cieľom vytvorenia tejto príručky je písanie zdrojového kódu v jednotnom štýle. To má za následok rýchle pochopenie cudzieho zdrojového kódu a zjednodušenie upravovania a doplňovania cudzieho kódu. Táto príručka je k dispozícii ako príloha A.

- **Jednotkové testovanie** Pre účely jednotkového testovania bol zvolený framework GoogleTest. Tento framework je pomerne jednoduchý na inštaláciu a používanie. V rámci práce na predmete MSI bola vytvorená metodika, ktorá popisuje inštaláciu a používanie tohto frameworku a definuje pravidla vytvárania jednotkových testov. Metodika je k dispozícii ako príloha B.

- **Verziovací systém** Ako verziovací systém bol zvolený Apache Subversion – ide o rozšírený, osvedčený nástroj, s ktorým majú viacerí členovia tímu už skúsenosti. Pre každého člena tímu bolo vytvorené osobitné konto, pod ktorým pristupuje do SVN. To umožňuje zistenie pôvodu vykonaných zmien.
  Na prácu so súbormi v SVN v operačnom systéme Windows používame nástroj TortoiseSVN. Ten umožňuje pohodlné prehliadanie súborov, ale aj ich úpravu a pridávanie nových súborov.
  Na integráciu SVN s vývojovým prostredím Microsoft Visual Studio 2010 používame voľne dostupný doplnok AnkhSVN.
  Návody na inštaláciu používanie SVN a súvisiacich nástrojov sú k dispozícii v podobe dokumentu, ktorý sa nachádza na dokumentovom serveri Google prislúchajúcemu našej Google skupine.
  SVN úložisko dát ma nasledujúcu štruktúru (popísané sú iba adresáre určené pre používateľské súbory):

  – **branches** - obsahuje verzie aplikácie, na ktorých sa testujú zásadné zmeny vo funkcionalite
  – **trunk** - obsahuje hlavnú vývojovú vetvu aplikácie a všetky zdieľané súbory, týkajúce sa tímového projektu (mapy, dokumenty, diagramy, obrázky)

6.2 Manažment rizík

Neodmysliteľnou súčasťou každého projektu sú riziká, ktoré môžu výrazne ovplyvniť výsledok a kvalitu výstupného produktu. Aj ked mnohé z nich sa nedajú do úplnej miery predpovedať, existuje spôsob, ako sa pokúsiť ich aspoň zmierniť. V rámci tímového projektu bol vypracovaný manažment rizík, v ktorom sa vhodnými akciami podniknutými ako reakcia na dané riziká snažíme predísť zbytočným komplikáciám.

Vzhľadom ku charakteru projektu a veľkosti tímu boli spolu s preventívnymi opatreniami identifikované nasledovné riziká:

- **Riziko vypadnutia člena z tímu**
  Spôsob eliminácie/zmiernenia rizika: zámerom je podobne ako v technike párového programovania vytvoriť užšiu spoluprácu medzi dvojica programátorov

na diaľku, aby v prípade vypadnutia jedného z nich nedošlo ku ohrozeniu pokračovania projektu

- **Riziko nedokončenia naplánovaných prác**
  Spôsob eliminácie/zmiernenia rizika: pri vytváraní dlhodobých plánov uvažovať viac pesimisticky ako optimisticky za účelom dodržania naplánovaných termínov

- **Riziko zvolenej technológie**
  Spôsob eliminácie/zmiernenia rizika: znalosti v rámci identifikovanej problematickej technológie distribuovať medzi viacerých členov, aby sa predišlo časovému sklzu pri vypadnutí kľúčového člena tímu

- **Riziko straty zdrojového kódu**
  Spôsob eliminácie/zmiernenia rizika: centrálne umiestnenie programového kódu na vzdialený server

- **Riziko nekonzistentnej formy zdrojového kódu**
  Spôsob eliminácie/zmiernenia rizika: povinnosť dodržiavať metodiku pre korektné písanie programového kódu

- **Riziko práce s neaktuálnym kódom**
  Spôsob eliminácie/zmiernenia rizika: použítie verziovacieho systemu SVN pre programový kód projektu a takisto aj dokumentáciu

- **Riziko nekonzistentného pomenovávania úloh**
  Spôsob eliminácie/zmiernenia rizika: povinosť dodržiavať metodiku pre zadávanie úlohy v systéme Redmine

6.3 Manažment rozvrhu a plánovanie

Manažment rozvrhu a plánovania je vhodnou voľbou najmä ako ukazovateľ, či sa projekt uberá správnou cestou a či sú naplánované činnosti vykonávané načas. Plánovanie prebiehalo na základe odhadov schopností jednotlivých členov tímu, pričom do úvahy bolo nutné vziať aj fakt, že existovalo stále riziko odchodu niektorého člena z tímu. Plány boli vypracovávané skôr pesimisticky, aby sa zabránilo výraznému sklzu projektu.

Pri plánovaní boli vypracované dva dlhodobé plány zvlášť pre zimný a letný semester. Dlhodobý plán pre zimný semester bol pravidelne každý mesiac revidovaný, pretože sa objavovali postupom času nové nápady, spôsoby a prístupy, ako riešiť danú problematiku a zefektívniť výsledný produkt.

V tabuľkách 3.4, 3.2, 3.3 možno pozorovať podľa jednotlivých týždňov sumarizáciu činností naplánovaných k danému dátumu. Vplyvom pesimistického plánovania sa dosiahol stav, že sa v konečnom ponímaní dokázalo stihnúť viac, ako bolo naplánované. Tomu svedčí aj vývoj plánov.

V prvej verzii dlhodobého plánu k zimnému semestru nemal vývojový tím ešte dostatočné znalosti k tomu, aby dokázal reálne zhodnotiť, čo je možné počas dvanástich týždňov dosiahnuť. Preto sa plán v druhej revízii čiastočne zmenil. Po úvodných analýzach ohľadom možností uberania sa projektu bola úplne zmenená koncepcia štruktúry programu, ktorý bol rozdelený do samostatných modulov. Takýto krok bol z dlhodobého hľadiska veľmi prospešný, pretože to napomohlo rýchlosti aj efektívnosti práce jednotlivých členov tímu.

Do tretej verzie plánu boli zahrnuté už pomerne náročné koncepty riešenia aspektov simulovania davu. Pribudla navigačná geometria označovaná ako Navigation Mesh a ako posledná bola zapracovaná efektívna koncepcia ovplyvňovania pohybu agenta na základe pohybu davu (označovaná ako FlowField technika). Všetky stanovené ciele v plánoch sa podarilo splniť načas, čo je dôkazom toho, že manažment plánovania nezlyhal.

6.4 Manažment podpory vývoja

Podpora vývoja svojou podstatou nepatrí medzi neodmysliteľné súčasti projektu, avšak jej zabezpečením sa výrazne zvyšuje spokojnosť a komport ľudí podieľajúcich sa na vývoji. Manažér podpory vývoja preto jednotlivé kroky plne prispôsobil tímovému projektu zloženého zo šiestich členov, pričom jeho prvoradou úlohou bolo adaptovať známe a všeobecne používané nástroje a techniky.

Ako operačný systém pre server tímového projektu bola zvolená linuxová distribúciu Ubuntu Server 10.4 64bit, s ktorou boli schopní pracovať všetci členovia. Pre webový server bol vybratý Apache HTTP Server a pre databázový engine sa použilo MySQL. Stránka tímu je vytvorená vo frameworku Joomla.

Voľbou systému Redmine sa zabezpečila požiadavka na kvalitný systém na podporu plánovania a súčasne aj systém na sledovanie zmien. Redmine má dobrú podporu vývoja SCRUM metódou, obsahuje Ganttov diagram a podľa potreby je možnosť rozšíriť ho rôznymi zásuvnými modulmi.

Ako centrálne úložisko dát projektu bol použitý Apache Subversion (SVN). Jeho výhody sú centralizovanosť a možnosť prepojiť so systémom Redmine a Visual Studiom. Klientská aplikácia pre SVN TortoiseSVN je pravdepodobne najznámejším zástupcom spomedzi aplikácii pre prácu s SVN a vzhľadom na to, že s ňou už členovia tímu mali skúsenosti, voľba padla práve na ňu. Aby už od začiatku nevznikali problémy s identifikáciou prispievateľa, správca servera vytvoril návod ako si každý nastaví používateľské meno a heslo do SVN.

Ako vývojové prostredie pre projekt bolo zvolené Microsoft Visual Studio 2010 Professional (VS). Bolo to najmä z dôvodu výbornej podpory C/C++, v ktorom je projekt vyvíjaný a tiež kvôli tomu, že je v rámci školy voľne dostupné cez MSDNAA. Edícia Professional má takisto prostriedky na analyzovanie behu programu z pohľadu vyťaženia operačnej pamäte a procesora, čo sa môže hodiť v neskoršej fáze vývoja pri narazení na výkonnostný problém. Pre pohodlnejšiu prácu so zdrojovým kódom boli ako zásuvné moduly vybraté AnksSVN a Visual Assist. Prvý pridáva podporu SVN priamo do Visual Studia a druhý možnosti refaktoringu, vyhľadávania v kóde, zvýrazňovania syntaxe a mnoho ďalších pomôcok.

Podporu vývoja bolo potrebné zabezpečiť aj z hľadiska komunikácie. Na začiatku semestra tím komunikoval prostredníctvom skupiny vytvorenej v sociálnej sieti Facebook, ale neskôr bola komunikácia presunutá do skupiny vytvorenej cez Google Groups, kde si formou mailov vymieňali členovia tímu potrebné informácie.

6.5 Manažment monitorovania projektu

Monitorovanie projektu zohráva pri vývoji softvéru dôležitú úlohu. Cieľom je prostredníctvom zaznamenaných údajov a postrehov z vykonanej práce prispôsobiť nasledovný vývoj. Takýmto spôsobom je možné skoré identifikovanie problémov a vyhnutie sa im so zámerom minimalizácie negatívneho dopadu na progres prác v projekte. V projekte používame viaceré monitorovacie nástroje a techniky, pomocou ktorých monitorujeme stav projektu a prispôsobujeme tak výber a plánovanie úloh pre ďalší šprint. Medzi hlavné prostriedky monitorovania patria:

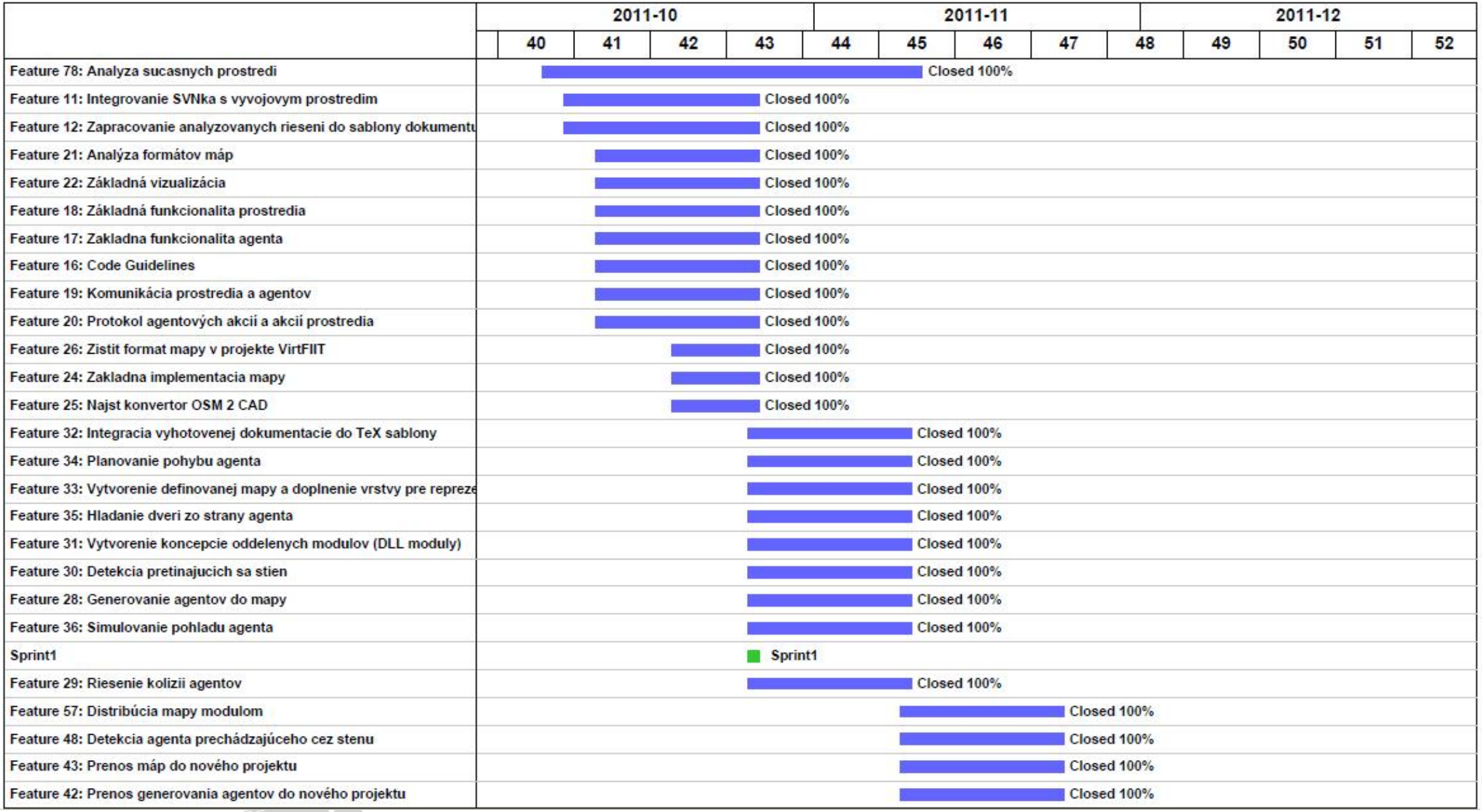
- **Tímové stretnutia**
  Tímové stretnutia sú najdôležitejším článkom monitorovania projektu. Šprinty plánujeme každé dva týždne a na jednotlivých stretnutiach diskutujeme progres prác všetkých členov tímu po prvej polovici šprintu a vyhodnotenie prác s prezentáciou na konci dvojtýždňového šprintu.

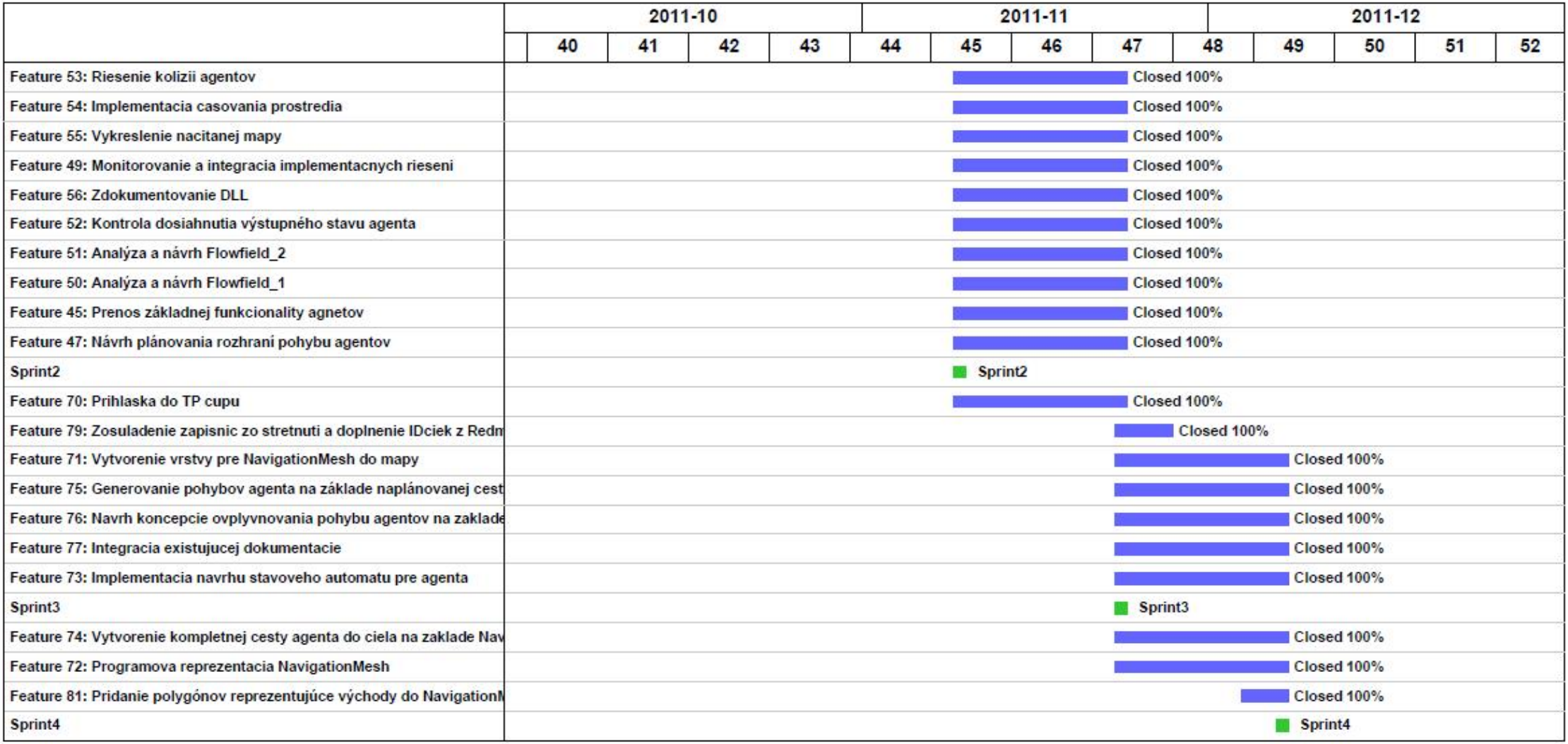
  Stretnutia poskytujú ideálne prostredie pre komunikáciu medzi členmi tímu,

pretože je možné priamo diskutovať akékoľvek problémy alebo nápady. Retrospektívny charakter nám umožňuje preberať témy zo šprintov a teda dynamicky prispôsobovať aktuálny stav projektu so zámerom úspešného dokončenia v presne špecifikovanej miere.

Plánovanie nasledujúceho šprintu je vykonávané ohodnotením úloh na základe odhadov členov tímu a ich výber, ktorý ovplyvňuje počet spálených hodín v minulom šprinte. Plány šprintov sú priložené v kapitole 3. Odhady k jednotlivým úlohám je možné vidieť v kapitole 8 v príslušných zápisniciach jednotlivých týždňových tímových stretnutí.

- **Monitorovanie nástrojom Redmine**
  Prostredníctvom nástroja Redmine sme informovaní o aktuálnom stave projektu. Všetky úlohy, bugy a pomocné úlohy sa nachádzajú v systéme aj s detailnými informáciami ohľadom odhadov času, stráveného času, progresu, členmi pridelenými pre úlohu a ďalšími informáciami. Redmine poskytuje vizualizáciu úloh s termínmi ukončenia pomocou kalendára a Ganttovho diagramu, ktorý nám z hľadiska časového charakteru znázorňuje progres prác jednotlivých úloh v šprintoch. Výstupný diagram prác je na nasledujúcej strane.


2011-10
2011-11
2011-12
40
41
42
43
44
45
46
47
48
49
50
51
52
Feature 78: Analyza sucasnych prostredi
Closed 100%
Feature 11: Integrovanie SVNka s vyvojovym prostredim
Closed 100%
Feature 12: Zapracovanie analyzovanych rieseni do sablony dokumentu
Closed 100%
Feature 21: Analýza formátov máp
Closed 100%
Feature 22: Základná vizualizácia
Closed 100%
Feature 18: Základná funkcionalita prostredia
Closed 100%
Feature 17: Zakladna funkcionalita agenta
Closed 100%
Feature 16: Code Guidelines
Closed 100%
Feature 19: Komunikácia prostredia a agentov
Closed 100%
Feature 20: Protokol agentových akcií a akcií prostredia
Closed 100%
Feature 26: Zistit format mapy v projekte VirtFIIT
Closed 100%
Feature 24: Zakladna implementacia mapy
Closed 100%
Feature 25: Najst konvertor OSM 2 CAD
Closed 100%
Feature 32: Integracia vyhotovenej dokumentacie do TeX sablony
Closed 100%
Feature 34: Planovanie pohybu agenta
Closed 100%
Feature 33: Vytvorenie definovanej mapy a doplnenie vrstvy pre repreze
Closed 100%
Feature 35: Hladanie dveri zo strany agenta
Closed 100%
Feature 31: Vytvorenie koncepcie oddelenych modulov (DLL moduly)
Closed 100%
Feature 30: Detekcia pretinajucich sa stien
Closed 100%
Feature 28: Generovanie agentov do mapy
Closed 100%
Feature 36: Simulovanie pohladu agenta
Closed 100%
Sprint1
Sprint1
Feature 29: Riesenie kolizii agentov
Closed 100%
Feature 57: Distribúcia mapy modulom
Closed 100%
Feature 48: Detekcia agenta prechádzajúceho cez stenu
Closed 100%
Feature 43: Prenos máp do nového projektu
Closed 100%
Feature 42: Prenos generovania agentov do nového projektu
Closed 100%


2011-10
2011-11
2011-12
40
41
42
43
44
45
46
47
48
49
50
51
52
Feature 53: Riesenie kolizii agentov
Closed 100%
Feature 54: Implementacia casovania prostredia
Closed 100%
Feature 55: Vykreslenie nacitanej mapy
Closed 100%
Feature 49: Monitorovanie a integracia implementacnych rieseni
Closed 100%
Feature 56: Zdokumentovanie DLL
Closed 100%
Feature 52: Kontrola dosiahnutia výstupného stavu agenta
Closed 100%
Feature 51: Analýza a návrh Flowfield_2
Closed 100%
Feature 50: Analýza a návrh Flowfield_1
Closed 100%
Feature 45: Prenos základnej funkcionality agnetov
Closed 100%
Feature 47: Návrh plánovania rozhrani pohybu agentov
Closed 100%
Sprint2
Sprint2
Feature 70: Prihlaska do TP cupu
Closed 100%
Feature 79: Zosuladenie zapisnic zo stretnuti a doplnenie IDciek z Redn
Closed 100%
Feature 71: Vytvorenie vrstvy pre NavigationMesh do mapy
Closed 100%
Feature 75: Generovanie pohybov agenta na základe naplánovanej cest
Closed 100%
Feature 76: Navrh koncepcie ovplyvnovania pohybu agentov na zaklade
Closed 100%
Feature 77: Integracia existujucej dokumentacie
Closed 100%
Feature 73: Implementacia navrhu stavoveho automatu pre agenta
Closed 100%
Sprint3
Sprint3
Feature 74: Vytvorenie kompletnej cesty agenta do ciela na zaklade Nav
Closed 100%
Feature 72: Programova reprezentacia NavigationMesh
Closed 100%
Feature 81: Pridanie polygónov reprezentujúce východy do NavigationM
Closed 100%
Sprint4
Sprint4

6.6 Manažment komunikácie

Komunikácia pri riešení tímového projektu je jedným z najdôležitejších aspektov spolupráce členov virtuálneho tímu. Počas riešenia tímového projektu sa stretávame jedenkrát týždenne na formálnom osobnom stretnutí s vedúcim projektu. Počas osobného stretnutia si rozdeľujeme prácu na projekte, ohodnocujeme časovú náročnosť jednotlivých čiastkových problémov a komunikujeme ohľadne problémov, ktoré sa vyskytli pri riešení úloh.

Mimo oficiálnych stretnutí sa stretávame len príležitostne, zvyčajne v menších skupinkách. Čiastkové problémy, ktoré sú rozdelené na osobnom stretnutí vypracovávame samostatne na rozdielnych geografických umiestneniach. Počas riešenia problému je každý člen odkázaný len na komunikáciu cez elektronické komunikačné prostriedky.

Na e-mailovú komunikáciu sme si zvolili použitie služby **Google Groups**. Služba je vo svojej podstate e-mailovou adresou, ktorú je možné nastaviť podľa špecifických potrieb. Keďže v komunikácií sa vyskytujú dôverné informácie (ako napr. kontakty na jednotlivých členov), skupina je prístupná len pre členov tímu. Pre iných ľudí je skupina prístupná pod mailovou adresou *tim08@googlegroups.com*, kde môžu zaslať svoj mail, ktorý sa rozpošle všetkým členom skupiny.

Pre účely okamžitej komunikácie sme si zvolili službu **Google Talk**. Služba nám ponúka komunikačné nástroje ako sú okamžité správy a po doinštalovaní rozšírenia do webového prehliadača aj telefonát a videotelefonát. Všetky vymenované nástroje je možné použiť aj v skupinovej podobe. V prípade potreby zdieľania pracovnej plochy tím využíva komunikačný nástroj **Skype**, ktorý obsahuje tiež už zmienené okamžité správy, telefonát a videotelefonát.

V rámci nástroja na manažment úloh **Redmine**, je komunikácia poslabšia. Na druhú stranu jej výpovedná hodnota je vyššia, pretože komunikácia je priamo zviazaná s témou, najčastejšie s chybou systému. V nástroji je možné sa vyjadrovať k jednotlivým úlohám, zmenám v repozitári, kontrolovať aktivitu jednotlivých členov atď. Nástroj na manažment úloh je v našom tíme prepojený s verziovacím nástrojom SVN. V rámci verziovacieho systému má každý člen svoje vlastné konto a pri vykonaní zmeny je povinný zmenu okomentovať. Dokumenty vytvorené pri riešení projektu sú rozdelené do dvoch častí:

- **Oficiálne dokument** – sú uložené v SVN a na stránke projektu
- **Neoficiálne dokumenty** – sú uložené a zdieľané v službe Google Docs

Pre prezentačné účely tímu je zriadená internetová stránka na adrese:

- **http://vm24.ucebne.fiit.stuba.sk**

Stránka je vytvorená pomocou CMS systému Joomla. Vďaka tejto voľbe je pridávanie nových príspevkov veľmi jednoduché aj pre členov ktorý s webom aktívne nepracujú.

Kapitola 7

Metodiky potrebné pri vývoji

Pri vytváraní dokumentácie, programového kódu, ale aj počas vykonávania akýchkoľvek iných podporných činnosti je v zásade odporúčané dodržiavať určité pravidlá. V rámci tohto procesu boli v tíme vytvorené metodiky za účelom zjednoteného a kvalitného procesu vytvárania softvéru. Nasledujúce podkapitoly podrobnejšie opisujú spomínané postupy a objasňujú zadefinované pravidlá.

7.1 Dokumentácia

Dokumentácia riadenia aj projektová dokumentácia sú vytvorené v jazyku Latex prostredníctvom editora TeXworks. Požadované výstupné súbory sú vo formáte pdf. Pri písaní dokumentácie je doležité dodržiavať nasledovné základné pravidlá:

- vytvárané súbory musia zapadať do koncepcie vytvorenej štruktúry. Prílohy sú umiestňované do podadresára ”prilohy” v hlavnom koreňovom adresári dokumentácie (či už projektovej alebo riadiacej)
- názov vytvoreného súboru musí obsahovať názov kapitoly
- prílohy v inom formáte ako .tex sa umiestňujú takisto do podadresára "prilohy"
- obrázky nachádzajúce sa mimo príloh sú umiestňované do podadresára ”screens” v hlanom koreňovom adresári dokumentácie. V opačnom prípade je potreba ich umiestnenia do adresára "screens"v podadresári "prilohy"v hlavnom adresári.
- zápisnice vo formáte pdf sú umiestňované do podadresára ”zapisnice” v hlavnom koreňovom adresári dokumentácie (týka sa len dokumentácie riadenia)

Formát dokumentu korešponduje s odporúčaným formátom spomenutým v knižnom diele prof. Márie Bielikovej s názvom ”Ako úspešne vyriešiť projekt”.

7.2 Zápisnice

Priebeh stretnutia sa zapisuje do šablóny zápisnice, ktorá bola vytvorená na začiatku celého projektu. Pri vytváraní zápisnice je jej názov stanovený ako "Zapisnica<číslo stretnutia>.pdf". Podľa stanovených pravidiel hlavička zápisnice musí obsahovať nasledovné náležitosti:

- názov stretnutia
- meno pedagogického vedúceho
- mená zúčastnených členov tímu
- mená chýbajúcich členov tímu
- dátum vypracovania
- miesto a čas stretnutia
- meno zapisujúceho

Samotný zápis zo stretnutia potom obsahuje tieto časti:

- Téma stretnutia
- Vyhodnotenie úloh z predchádzajúceho stretnutia - každá úloha musí pritom obsahovať nasledovné informácie:
  - Číslo úlohy
  - Redmine ID úlohy
  - Popis úlohy
  - Kto zodpovedá za úlohu
  - Strávený čas v hodinách
  - Percentuálne vyjadrenie dokončenia úlohy
  - Stav úlohy
- Úlohy na ďalšie stretnutie - každá úloha musí obsahovať nasledovné informácie
  - Číslo úlohy
  - Redmine ID úlohy
  - Popis úlohy
  - Kto zodpovedá za úlohu
  - Strávený čas v hodinách
  - Percentuálne vyjadrenie dokončenia úlohy

7.3 Zdrojový kód

Pre dosiahnutie čo najväčšej kvality produktu z vývojárskeho hľadiska je veľmi dôležité stanoviť pravidlá, ktoré vývojári dodržiavajú pri implementácii. Výsledkom je zefektívnenie vývoja, väčšia produktivita a lepšia komunikácia medzi členmi vývojárskeho tímu. Pravidlá použité pri vývoji tejto aplikácie sa nachadzajú v prílohe A.

7.4 Testovanie

Okrem vytvorenie zdrojového kódu je dôležitou súčasťou vývoja softvérového produktu aj testovanie samotnej funkcionality. V užívateľskom prostredí Microsoft Visual Studio 2010 je za pomoci frameworku GoogleTest možné vykonávať testovanie od zadefinovania testov až po ich spúšťanie. Detailnejšie je proces testovania popísaný v prílohe B.

7.5 Úlohy v systéme Redmine

Vzhľadom k tomu, že v rámci timového projektu je používaný plánovací systém Redmine, je z dôvodu zabezpečenia konzistentnosti zadávaných úloh potrebné definovať konkrétny proces pomenovávania a kategorizovania spomínaných úloh. Podrobnejšie informácie sú obsiahnuté v prílohe C.

Kapitola 8

Záznamy zo stretnutí

Kapitola obsahuje jednotlivé zápisnice zo stretnutí počas dvoch semestrov. Zápisnice obsahujú základné informácie ako mená zúčastnených členov tímu, témy stretnutí, vyhodnotenie úloh z predchádzajúcich stretnutí a takisto aj úlohy týkajúce sa nasledujúcich stretnutí.

Cieľom formulovaných zápisnic bolo priniesť ucelený prehľad vývoja projektu rozdeleného do týždňov. V zápisniciach možno pozorovať úspešnosť plnenia úloh, aktivity jednotlivých členov a takisto aj vývoj projektu od počiatočného štádia až do úspešného ukončenia.

Zápis 1. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš Bc. Vojtech Villaris	Dátum: 28.9.2011 Miestnosť: Softvérové štúdio Čas: 18:00 – 19:00
Chýbajú:	Zápis vypracoval: Bc. Lukáš Pavlech

Téma stretnutia

Úvod, bližšie oboznámenie sa s projektom

Vyhodnotenie úloh z predchádzajúceho stretnutia

1. Neexistovali žiadne úlohy (prvé stretnutie)

Opis stretnutia

1. Zoznámenie členov tímu s vedúcim
2. Informovanie tímu o súťaži TP cup
3. Oboznámenie sa s požiadavkami na výsledok projektu
4. Informovanie vedúceho o problémoch so serverom (DNS)
5. Rozdelenie úloh na ďalšie stretnutie

Úlohy na ďalšie stretnutie

Číslo úlohy	Popis úlohy	Kto	Termín ukončenia	Doba trvania
1.1	Vybrať meno tímu	Všetci	1.10.2011	-
1.2	Vytvoriť plagát tímu	Michal Fornádeľ	2.10.2011	-
1.3	Vybratie servera	Všetci	29.9.2011	-
1.4	Vytvorenie šablóny pre zápisy	Lukáš Pavlech	1.10.2011	-
1.5	Inštalácie linuxového servera	Lukáš Pavlech	29.9.2011	-
1.6	Vytvorenie web prezentácie projektu	Lukáš Pavlech	30.9.2011	-
1.7	Sprevádzkovanie Redmine	Michal Fornádeľ	5.10.2011	-
1.8	Sprevádzkovanie SVN	Adam Pomothy	5.10.2011	-
1.9	Analýza súčasných prostredí	Všetci	5.10.2011	-
1.10	Návrh rozhrania pre simulačné prostredie a agentov	Všetci	5.10.2011	-

Zápis 2. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš Bc. Vojtech Villaris	Dátum: 05.10.2011 Miestnosť: Softvérové štúdio Čas: 8:00 – 10:00
Chýbajú:	Zápis vypracoval: Bc. Adam Pomothy

Téma stretnutia

Použité technológie, technické záležitosti (Redmine, SVN), prvá diskusia ohľadom konkrétneho riešenia

Opis stretnutia

1. Diskusia o výbere implementačných technológií a vývojového prostredia
2. Diskusia o vytvorení backlogu
3. Spresnenie zadania tímového projektu (evakuácia)
4. Diskusia o tvorbe dokumentácie – zvolenie systému Tex na spracovávanie dokumentácie
5. Diskusia o existujúcich riešeniach
6. Diskusia o konkrétnom riešení zadania tímového projektu

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Popis úlohy	Kto	Strávený čas	Hotovo (%)	Stav
1.1	Vybrať meno tímu	Všetci	-	100%	Dokončené
1.2	Vytvoriť plagát tímu	Michal Fornádeľ	-	100%	Dokončené
1.3	Vybratie servera	Všetci	-	100%	Dokončené
1.4	Vytvorenie šablóny pre zápisy	Lukáš Pavlech	-	100%	Dokončené
1.5	Inštalácie linuxového servera	Lukáš Pavlech	-	100%	Dokončené
1.6	Vytvorenie web prezentácie projektu	Lukáš Pavlech	-	100%	Dokončené
1.7	Sprevádzkovanie Redmine	Michal Fornádeľ	-	100%	Dokončené
1.8	Sprevádzkovanie SVN	Adam Pomothy	-	70%	Rozpracované
1.9	Analýza súčasných prostredí	Všetci	-	40%	Rozpracované
1.10	Návrh rozhrania pre simulačné prostredie a agentov	Všetci	-	100%	Dokončené

Úlohy na ďalšie stretnutie

Neboli pridelené žiadne nové úlohy

Zápis 3. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 12.10.2011 Miestnosť: Softvérové štúdio Čas: 8:00 – 10:00
Chýbajú:	Zápis vypracoval: Bc. Marek Hlaváč

Téma stretnutia

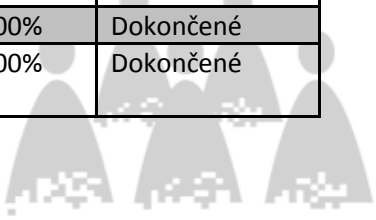
Návrh architektúry prototypu, naplánovanie úloh pre 1. Šprint

Opis stretnutia

1. Diskusia ohľadom architektúry prototypu
2. Vytvorenie návrhu architektúry prototypu
3. Spresnenie rozsahu implementácie prototypu
4. Rozdelenie prác na prototype do úloh pre 1. šprint
5. Hlasovanie ohľadom zložitosti jednotlivých úloh
6. Pridelenie úloh pre 1. šprint členom tímu

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Popis úlohy	Kto	Strávený čas	Hotovo (%)	Stav
1.1	Vybrať meno tímu	Všetci	-	100%	Dokončené
1.2	Vytvoriť plagát tímu	Michal Fornádeľ	-	100%	Dokončené
1.3	Vybratie servera	Všetci	-	100%	Dokončené
1.4	Vytvorenie šablóny pre zápisy	Lukáš Pavlech	-	100%	Dokončené
1.5	Inštalácie linuxového servera	Lukáš Pavlech	-	100%	Dokončené
1.6	Vytvorenie web prezentácie projektu	Lukáš Pavlech	-	100%	Dokončené
1.7	Sprevádzkovanie Redmine	Michal Fornádeľ	-	100%	Dokončené
1.8	Sprevádzkovanie SVN	Adam Pomothy	-	100%	Dokončené
1.9	Analýza súčasných prostredí	Všetci	-	100%	Dokončené
1.10	Návrh rozhrania pre simulačné prostredie a agentov	Všetci	-	100%	Dokončené



Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)
3.1	16	Code Guidelines	Adam Pomothy	8h	0%
3.2	17	Základná funkcionalita agenta	Michal Fornádeľ	8h	0%
3.3	18	Základná funkcionalita prostredia	Lukáš Pavlech	10h	0%
3.4	19	Komunikácia prostredia a agentov	Daniel Petráš	10h	0%
3.5	20	Protokol agentových akcií a akcií prostredia	Daniel Petráš	2h	0%
3.6	21	Analýza formátov máp	Marek Hlaváč	4h	0%
3.7	22	Základná vizualizácia	Martin Košický	8h	0%
3.8	25	Nájsť konvertor OSM -> CAD	Marek Hlaváč	2h	0%

Zápis 4. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 19.10.2011 Miestnosť: Softvérové štúdio Čas: 08:00 – 10:00
Chýbajú: -	Zápis vypracoval: Bc. Daniel Petráš

Téma stretnutia

Kontrola priebehu práce na úlohách.

Opis stretnutia

1. Kontrola práce na úlohách, vyplnenie priebehu úloh v Redmine
2. Prezentácia analýzy formátov máp (OSM vs. CAD, VirtFIIT)
3. Návrh štruktúry súborov a projektov (viac projektov vs. jeden projekt)
4. Reprezentácia pozície agenta (celé vs. reálne čísla)
5. Ozrejmenie implementácie komunikačného rozhrania (úloha 3.4)
6. Návrh testovacieho prostredia (GTest vs. Unit test vo Visual Studiu)
7. Návrh použitia STL + boost knižnice (smart pointre, parsovanie)

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
3.1	16	Code Guidelines	Adam Pomothy	6h	80%	Rozpracované
3.2	17	Základná funkcionalita agenta	Michal Fornádeľ	4h	60%	Rozpracované
3.3	18	Základná funkcionalita prostredia	Lukáš Pavlech	6h	50%	Rozpracované
3.4	19	Komunikácia prostredia a agentov	Daniel Petráš	7h	50%	Rozpracované
3.5	20	Protokol agentových akcií a akcií prostredia	Daniel Petráš	0h	0%	Rozpracované
3.6	21	Analýza formátov máp	Marek Hlaváč	3h	100%	Dokončené
3.7	22	Základná vizualizácia	Martin Košický	6h	80%	Rozpracované
3.8	25	Nájsť konvertor OSM -> CAD	Marek Hlaváč	2h	100%	Dokončené

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)
4.1	24	Základná implementácia mapy	Marek Hlaváč	6h	0%
4.2	26	Zistiť formát mapy v projekte VirtFIIT	Lukáš Pavlech	1h	0%

Zápis 5. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 26.10.2011 Miestnosť: Softvérové štúdio Čas: 8:00 – 10:00
Chýbajú:	Zápis vypracoval: Bc. Martin Košický

Téma stretnutia

Naplánovanie úloh pre druhý šprint, identifikovanie úloh pre pohyb agenta v prostredí, vyjasnenie problematiky použitia pokročilých techník pre použitie v rámci projektu (NavigationMesh)

Opis stretnutia

1. Diskusia o možnom použití virtuálnych dverí ako možný spôsob navigácie agenta
2. Diskusia ohľadom využitia budovy fakulty ako model pre simuláciu evakuácie
3. Diskusia na tému vytvorenie prostredia, ktoré vidí agent na základe polygónov (vytvorila by sa samostatná vrstva na mape ktorá by určovala, kade sa môže agent pohybovať) – funkcionalita momentálne zamietnutá, možnosť realizácie v ďalšom šprinte.
4. Diskusia ohľadom využitia externých knižníc pre počítanie priesečníkov úsečiek a inej matematiky, diskusia ohľadom vytvorenia plánovacieho modulu

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
3.1	16	Code Guidelines	Adam Pomothy	8h	100%	Dokončené
3.2	17	Základná funkcionalita agenta	Michal Fornádeľ	8h	100%	Dokončené
3.3	18	Základná funkcionalita prostredia	Lukáš Pavlech	15h	100%	Dokončené
3.4	19	Komunikácia prostredia a agentov	Daniel Petráš	15h	100%	Dokončené
3.5	20	Protokol agentových akcií a akcií prostredia	Daniel Petráš	2h	90% *1	Rozpracované
3.7	22	Základná vizualizácia	Martin Košický	8h	100%	Dokončené
4.1	24	Základná implementácia mapy	Marek Hlaváč	8h	100%	Dokončené
4.2	26	Zistiť formát mapy v projekte VirtFIIT	Lukáš Pavlech	0.5h	100%	Dokončené

*1 - chýba dokumentácia

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)
5.1	36	Simulovanie pohľadu agenta	Martin Košický, Michal Fornádeľ	40h	0%
5.2	35	Hľadanie dverí zo strany agenta	Adam Pomothy	8h	0%
5.3	33	Vytvorenie definovanej mapy a doplnenie vrstvy pre reprezentáciu dverí	Marek Hlaváč	3h	0%
5.4	31	Vytvorenie koncepcie oddelených modulov (DLL moduly)	Martin Košický	5h	0%
5.5	30	Detekcia pretínajúcich sa stien	Daniel Petráš	8h	0%
5.6	28	Generovanie agentov do mapy	Lukáš Pavlech	5h	0%

Zápis 6. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 2.11.2011 Miestnosť: Softvérové štúdio Čas: 8:00 – 10:00
Chýbajú:	Zápis vypracoval: Bc. Michal Fornádeľ

Téma stretnutia

Dopracovávanie úloh ohľadom inteligencie agenta a prostredia, prezentovanie myšlienky pokročilých súčasných techník pre simulovanie davu (NavigationMesh, FlowField Continuum Crowds koncept).

Opis stretnutia

1. Oboznámenie sa so spôsobom pohybu davu na základe hustoty a intuitívnou zmenou formácie davu. Diskusia na tému eventuálneho použitia v rámci projektu.
2. Diskusia ohľadom sprevádzkovania súboru knižníc Boost pre potreby projektu (spôsoby buildovania)
3. Diskusia na tému vytvorenie prostredia, ktoré vidí agent na základe polygónov (vytvorila by sa samostatná vrstva na mape ktorá by určovala, kade sa môže agent pohybovať) – funkcionalita momentálne zamietnutá, možnosť realizácie v ďalšom šprinte.
4. Diskusia ohľadom spôsobu riešenia konfliktu agentov a následného ich kontaktu so stenou (riešenie spočíva v kontrolovaní oboch možných situácií)
5. Opätovné riešenie problému hľadania dverí (plánovaná realizácia hľadania najbližších dverí, realizácia zatiaľ iba jednoduchých situácií)

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
5.1	36	Simulovanie pohľadu agenta	Martin Košický, Michal Fornádeľ	0h	0%	Rozpracované
5.2	35	Hľadanie dverí zo strany agenta	Adam Pomothy	6h	60%	Rozpracované
5.3	33	Vytvorenie definovanej mapy a doplnenie vrstvy pre reprezentáciu dverí	Marek Hlaváč	3h	100%	Dokončené
5.4	31	Vytvorenie koncepcie oddelených modulov (DLL	Martin Košický	4h	90%	Rozpracované

		moduly)				
5.5	30	Detekcia pretínajúcich sa stien	Daniel Petráš	12h	60%	Rozpracované
5.6	28	Generovanie agentov do mapy	Lukáš Pavlech	4.5h	100%	Dokončené

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)
6.1	34	Plánovanie pohybu agenta	Marek Hlaváč	10h	0%
6.2	32	Integrácia vyhotovenej dokumentácie do TeX šablóny	Michal Fornádeľ	15h	0%
6.3	29	Riešenie kolízii agentov	Lukáš Pavlech	13h	0%

Zápis 7. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 9.11.2011 Miestnosť: Softvérové štúdio Čas: 8:00 – 11:00
Chýbajú:	Zápis vypracoval: Bc. Lukáš Pavlech

Téma stretnutia

Ukončenie druhého šprintu, odovzdanie dokumentácie kontrolného bodu, naplánovanie tretieho šprintu.

Opis stretnutia

1. Odovzdanie tlačenej verzie dokumentácie vedúcemu
2. Zabudli sme priniesť preberací protokol – potreba priniesť nabudúce
3. Vyhodnotenie druhého šprintu
4. Diskutovanie o ďalšom šprinte (potreba dokončenia nesplnených častí z druhého šprintu + pridanie nových features)
5. Diskutovanie na tému vnútorný svet agenta – dohodnutie koncepcie navigácie agenta:
    a. Hľadanie dverí – obzeranie sa, identifikácia dverí
    b. Výber priechodu – dverí
    c. Pohyb k priechodu
    d. Aktualizácia navštívených miestností
    e. Obchádzanie prekážok (agentov)
    f. Výber alternatívnej cesty

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
5.1	36	Simulovanie pohľadu agenta	Martin Košický, Michal Fornádeľ	0h	0% *1	Nesplnené
5.2	35	Hľadanie dverí zo strany agenta	Adam Pomothy	8h	100%	Dokončené
5.4	31	Vytvorenie koncepcie oddelených modulov (DLL moduly)	Martin Košický	5h	100%	Dokončené
5.5	30	Detekcia pretínajúcich sa stien	Daniel Petráš	12h	60% *2	Rozpracované

6.1	34	Plánovanie pohybu agenta	Marek Hlaváč	2h	10%*3	Nesplnené
6.2	32	Integrácia vyhotovenej dokumentácie do TeX šablóny	Michal Fornádeľ	10h	100%	Dokončené
6.3	29	Riešenie kolízii agentov	Lukáš Pavlech	2h	15%	Rozpracované

*1 - zatiaľ sa odkladá

*2 – úloha sa predlžuje, predpokladala sa existencia knižníc riešiacich problematiku

*3 – zmena logiky simulácie, úloha bola zrušená

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)
7.1	42	Prenos generovania agentov do nového projektu	Lukáš Pavlech	2h	0%
7.2	43	Prenos máp do nového projektu	Marek Hlaváč	2h	0%
7.3	45	Prenos základnej funkcionality agentov	Lukáš Pavlech	2h	0%
7.4	47	Návrh plánovania rozhraní pohybu agentov	Martin Košický	8h	0%
7.5	48	Detekcia agenta prechádzajúceho cez stenu	Daniel Petráš	8h	0%
7.6	49	Monitorovanie a integrácia implementačných riešení	Michal Fornádeľ	-	0%
7.7	50	Analýza a návrh Flowfield_1	Michal Fornádeľ	10h	0%
7.8	51	Analýza a návrh Flowfield_2	Adam Pomothy	10h	0%
7.9	52	Kontrola dosiahnutia výstupného stavu agenta	Adam Pomothy	5h	0%
7.10	53	Riešenie kolízií agentov	Lukáš Pavlech	13h	0%
7.11	54	Implementácia časovania prostredia	Daniel Petráš	5h	0%
7.12	55	Vykreslenie načítanej mapy	Marek Hlaváč	7h	0%
7.13	56	Zdokumentovanie DLL	Martin Košický	5h	0%
7.14	57	Distribúcia mapy modulom	Martin Košický	4h	0%

Zápis 8. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 16.11.2011 Miestnosť: Softvérové štúdio Čas: 09:00 – 10:30
Chýbajú: Ing. Peter Lacko, PhD.	Zápis vypracoval: Bc. Adam Pomothy

Téma stretnutia

Riešenie implementačných problémov, súvisiacich s úlohami pridelenými na predchádzajúcom stretnutí.

Opis stretnutia

1. Diskusia ohľadom správnej reprezentácie 2D objektov v mape.
2. Návrh abstraktnej továrne (factory) na vytváranie objektov máp.
3. Refaktoring zdrojového kódu s cieľom úspešného spustenia aplikácie pri použití novej architektúry (aplikácia sa rozdelila na 3 oddelené, navzájom komunikujúce projekty)

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
7.1	42	Prenos generovania agentov do nového projektu	Lukáš Pavlech	1h	60%	Rozpracované
7.2	43	Prenos máp do nového projektu	Marek Hlaváč	2h	40%	Rozpracované
7.3	45	Prenos základnej funkcionality agentov	Lukáš Pavlech	2h	70%	Rozpracované
7.4	47	Návrh plánovania rozhraní pohybu agentov	Martin Košický	3h	60%	Rozpracované
7.5	48	Detekcia agenta prechádzajúceho cez stenu	Daniel Petráš	3h	30%	Rozpracované
7.6	49	Monitorovanie a integrácia implementačných riešení	Michal Fornádeľ	1h	0%	Rozpracované
7.7	50	Analýza a návrh Flowfield_1	Michal Fornádeľ	4h	30%	Rozpracované
7.8	51	Analýza a návrh Flowfield_2	Adam Pomothy	5h	40%	Rozpracované

7.9	52	Kontrola dosiahnutia výstupného stavu agenta	Adam Pomothy	5h	70%	Rozpracované
7.10	53	Riešenie kolízií agentov	Lukáš Pavlech	1h	25%	Rozpracované
7.11	54	Implementácia časovania prostredia	Daniel Petráš	0.5h	60%	Rozpracované
7.12	55	Vykreslenie načítanej mapy	Marek Hlaváč	7h	80%	Rozpracované
7.13	56	Zdokumentovanie DLL	Martin Košický	4h	60%	Rozpracované
7.14	57	Distribúcia mapy modulom	Martin Košický	2h	30%	Rozpracované

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)

Na tomto stretnutí neboli pridelené žiadne nové úlohy.

Zápis 9. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 23.11.2011 Miestnosť: Softvérové štúdio Čas: 8:00 – 11:00
Chýbajú:	Zápis vypracoval: Bc. Marek Hlaváč

Téma stretnutia

Ukončenie tretieho šprintu, stanovenie cieľov pre štvrtý šprint, naplánovanie štvrtého šprintu.

Opis stretnutia

1. Ukážka výsledku práce po 3 šprintoch
2. Kontrola zadaných úloh pre tretí šprint
3. Vyhodnotenie tretieho šprintu
4. Diskusia ohľadom plánu na ďalší šprint a identifikácia cieľov šprintu:
    a. Analýza možností použitia techniky FlowField pri plánovaní pohybu agentov
    b. Návrh a spôsob implementácie NavigationMesh umožňujúci komplexnejšiu navigáciu agenta prostredníctvom navigačnej vrstvy
    c. Návrh a spôsob implementácie inteligentného agenta s pokročilým plánovaním

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
7.1	42	Prenos generovania agentov do nového projektu	Lukáš Pavlech	2h	100%	Dokončené
7.2	43	Prenos máp do nového projektu	Marek Hlaváč	4h	100%	Dokončené
7.3	45	Prenos základnej funkcionality agentov	Lukáš Pavlech	3h	100%	Dokončené
7.4	47	Návrh plánovania rozhraní pohybu agentov	Martin Košický	6h	100%	Dokončené
7.5	48	Detekcia agenta prechádzajúceho cez stenu	Daniel Petráš	10h	100%	Dokončené
7.6	49	Monitorovanie a integrácia implementačných riešení	Michal Fornádeľ	3h	100%	Dokončené
7.7	50	Analýza a návrh	Michal Fornádeľ	15h	100%	Dokončené

		Flowfield_1				
7.8	51	Analýza a návrh Flowfield_2	Adam Pomothy	12h	100%	Dokončené
7.9	52	Kontrola dosiahnutia výstupného stavu agenta	Adam Pomothy	7h	100%	Dokončené
7.10	53	Riešenie kolízií agentov	Lukáš Pavlech	2h	100%	Dokončené
7.11	54	Implementácia časovania prostredia	Daniel Petráš	1h	100%	Dokončené
7.12	55	Vykreslenie načítanej mapy	Marek Hlaváč	10h	100%	Dokončené
7.13	56	Zdokumentovanie DLL	Martin Košický	6h	100%	Dokončené
7.14	57	Distribúcia mapy modulom	Martin Košický	6h	100%	Dokončené

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)
9.1	71	Vytvorenie vrstvy pre NavigationMesh do mapy	Marek Hlaváč	5h	0%
9.2	72	Programová reprezentácia NavigationMesh	Marek Hlaváč	8h	0%
9.3	73	Implementácia návrhu stavového automatu pre agenta	Martin Košický	12h	0%
9.4	74	Vytvorenie kompletnej cesty agenta do cieľa na základe NavigationMesh	Lukáš Pavlech	14h	0%
9.5	75	Generovanie pohybov agenta na základe naplánovanej cesty	Daniel Petráš	10h	0%
9.6	76	Návrh koncepcie ovplyvňovania pohybu agentov na základe hustoty výskytu ostatných agentov	Adam Pomothy	10h	0%
9.7	77	Integrácia existujúcej dokumentácie	Michal Fornádeľ	12h	0%
9.8	79	Zosúladenie zápisníc zo stretnutí a doplnenie ID z Redminu	Michal Fornádeľ	3h	0%

Zápis 10. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 30.11.2011 Miestnosť: Softvérové štúdio Čas: 08:30 – 10:00
Chýbajú: -	Zápis vypracoval: Bc. Daniel Petráš

Téma stretnutia

Kontrola priebehu práce na úlohách a diskusia o FlowField.

Opis stretnutia

1. Vedúci tímu sa zúčastnil virtuálne pomocou Skype (kvôli chorobe).
2. Kontrola každej naplánovanej úlohy z predchádzajúceho stretnutia.
3. Identifikácia chýbajúcej funkcionality v úlohe 9.2 – chýbajúci graf uzlov.
4. Prezentácia navrhnutého riešenia úlohy 9.6.
5. Rozprava o použití FlowField metódy a vlastnej navrhnutej metódy v úlohy 9.6.
6. Dospeli sme k záveru, že obe metódy riešia trochu iný problém a bolo by vhodné v istej podobe implementovať obidve.
7. Identifikácia chyby vo vykresľovaní mapy, kde sa vnútorné steny nevykresľujú.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
9.1	71	Vytvorenie vrstvy pre NavigationMesh do mapy	Marek Hlaváč	2.5h	60%	Rozpracované
9.2	72	Programová reprezentácia NavigationMesh	Marek Hlaváč	6h	60%	Rozpracované
9.3	73	Implementácia návrhu stavového automatu pre agenta	Martin Košický	0h	0%	Rozpracované
9.4	74	Vytvorenie kompletnej cesty agenta do cieľa na základe NavigationMesh	Lukáš Pavlech	3h	30%	Rozpracované
9.5	75	Generovanie pohybov agenta na základe naplánovanej cesty	Daniel Petráš	7,5h	40%	Rozpracované
9.6	76	Návrh koncepcie	Adam Pomothy,	10h + 8h	30%	Rozpracované

		ovplyvňovania pohybu agentov na základe hustoty výskytu ostatných agentov	Michal Fornádeľ			
9.7	77	Integrácia existujúcej dokumentácie	Michal Fornádeľ	0h	0%	Rozpracované
9.8	79	Zosúladenie zápisníc zo stretnutí a doplnenie ID z Redminu	Michal Fornádeľ	3h	100%	Dokončené

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)
10.1	80	Vykresľovanie vnútorných stien	Martin Košický	2h	0%
10.2	83	Integrácia existujúcej dokumentácie	Adam Pomothy	10h	0%
10.3	84	Úprava a zosúladenie zápisníc zo stretnutí	Adam Pomothy	6h	0%

Zápis 11. stretnutia tímu č. 8

Vedúci pedagóg: Ing. Peter Lacko, PhD.	
Zúčastnení členovia tímu: Bc. Lukáš Pavlech Bc. Michal Fornádeľ Bc. Adam Pomothy Bc. Martin Košický Bc. Marek Hlaváč Bc. Daniel Petráš	Dátum: 7.12.2011 Miestnosť: Softvérové štúdio Čas: 08:30 – 10:00
Chýbajú: -	Zápis vypracoval: Bc. Michal Fornádeľ

Téma stretnutia

Ukončenie štvrtého šprintu a pokračovanie diskusie ohľadom navrhnutého FlowField algoritmu

Opis stretnutia

1. Uzavretie vykonaných úloh v štvrtom šprinte
2. Predvedenie funkčného prototypu aplikácie na rôznych mapách.
3. Diskusia na tému smerovania implementácie ovplyvňovania pohybu agentov na základe pohybu davu – FlowField koncepcia
4. Diskusia ohľadom poteciálnych možností pre simuláciu davu v letnom semestri

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Strávený čas (h)	Hotovo (%)	Stav
9.1	71	Vytvorenie vrstvy pre NavigationMesh do mapy	Marek Hlaváč	5h	100%	Dokončené
9.2	72	Programová reprezentácia NavigationMesh	Marek Hlaváč	10h	100%	Dokončené
9.3	73	Implementácia návrhu stavového automatu pre agenta	Martin Košický	10h	100%	Dokončené
9.4	74	Vytvorenie kompletnej cesty agenta do cieľa na základe NavigationMesh	Lukáš Pavlech	12h	100%	Dokončené
9.5	75	Generovanie pohybov agenta na základe naplánovanej cesty	Daniel Petráš	19.5h	100%	Dokončené
9.6	76,85	Návrh koncepcie ovplyvňovania pohybu agentov na základe hustoty výskytu	Adam Pomothy, Michal Fornádeľ	10h + 8h	100%	Dokončené

		ostatných agentov				
9.7	77,83	Integrácia existujúcej dokumentácie	Adam Pomothy, Michal Fornádeľ	14h+14h	100%	Dokončené
10.1	80	Vykresľovanie vnútorných stien	Martin Košický	1h	100%	Dokončené
10.3	84	Úprava a zosúladenie zápisníc zo stretnutí	Adam Pomothy	6h	100%	Dokončené

Úlohy na ďalšie stretnutie

Číslo úlohy	Redmine ID	Popis úlohy	Kto	Odhadovaná doba trvania (h)	Hotovo (%)

Dodatok A

Metodika vytvárania zdrojového kódu

Príloha sa zameriava na pravidlá definované počas vytvárania programového kódu. Cieľom je priniesť ucelený pohľad na štruktúru a spôsob zápisu kódu. Jednotlivé pravidlá sú umiestnené do nasledovných častí:

1. Všeobecné pravidlá
2. Pomenovávanie
3. Pomenovávanie súborov a komentovanie
4. Klauzuly "include"
5. Výrazy
6. Premenné a konštanty
7. Podmienky
8. Iné

Štruktúra pravidla

Krátky popis
Príklad ak je to možné
Popis

1. Všeobecné pravidlá

1. Každé pravidlo sa môže porušiť, ak to zvýši čitateľnosť kódu

2. Pravidlá sa môžu porušiť, ak má proti nim niekto silné námietky
Tieto pravidlá nemajú nikoho obmedzovať ani v produktivite, ani v kreatívnosti. Ak má skúsený programátor dobrý dôvod porušiť nejaké pravidlo, je dobré, aby o tom ostatní vedeli a on svoj dôvod aj vysvetlil – výsledkom môže byť zmena alebo doplnenie pravidiel.

2. Pomenovávanie

1. Názvy tried začína veľkými písmenami a na začiatku písmeno **C**, každé ďalšie slovo veľkým
CAgent, CAccountFetcher

2. Názvy premenných začínajú malým, každé ďalšie slovo veľkým
line, lineDimension

3. Názvy konštánt sú cele veľkými písmenami, jedNIElivé slová oddelené podtržníkom
MAX_TERATIONS

4. Názvy funkcií začínajú malým, ostatné slová s veľkým a musia obsahovať sloveso
getName(), computeTotalWidth()

5. Názvy menových priestorov (namespaces) sú celé malým
model::analyzer, io::iomanager, common::math::geometry

6. Názvy šablón (templates) sú jedno veľké písmeno
template<class C, class D> ...

7. Skratky v názvoch sa nepíšu celé veľkým, iba počiatočné písmeno

exportHtmlSource(); // NIE: exportHTMLSource();
openDvdPlayer(); // NIE: openDVDPlayer();

Ak by to tak nebolo, tak napríklad premenná by sa musela volať „hTML“ alebo „dVD“, čo nie je moc čitateľné

8. Globálne premenné sa referencujú ako ::operátor
::mainWindow.open(), ::applicationContext.getName()
Celkovo sa ale neodporúča používať globálne premenne.

9. Premenné typu *private* sa označujú sufixom „_“

class SomeClass {
private:
int length_;
}

Ďalšie možnosť je použiť rovnaký prefix namiesto sufixu, ale to výrazne znižuje čitateľnosť danej premennej. (pred tým, než vznesiete námietky, naozaj sa skúste pozrieť na kód so sufixami a potom s prefixami)

10. Generické premenné by mali mať rovnaký názov ako ich typy

void setTopic(Topic* topic) // NIE: void setTopic(Topic* value)
// NIE: void setTopic(Topic* aTopic)
// NIE: void setTopic(Topic* t)
Ale môže byť napr.
Point startingPoint, centerPoint;

Ak takéto pomenovanie logický nesedí, asi je zle pomenovaný samotný typ.

11. Všetko po anglicky
fileName; // NIE: nazovSuboru

12. Premenné s malým dosahom (scope) by mali mať krátke mena, s veľkým dosahom dlhé mena
int i; for(int i=0;…)
Čím vačší dosah, tým dlhšie/reprezentatívnejšie meno.

13. Meno objektu je implicitné a nemalo by sa vyskytovať v mene funkcie
line.getLength(); // NIE: line.getLineLength();

14. Keď funkcia slúži na priame priradenie alebo poskytnutie hodNIEy premennej, tak sa nazýva **get** a **set**

employee.getName();
employee.setName(name);
 	
matrix.getElement(2, 4);
matrix.setElement(2, 4, value);

15. Ak funkcia niečo počíta, tak sa pomenuje **compute**
valueSet->computeAverage(); matrix->computeInverse();
Výrazne zvyšuje čitateľnosť a intuitívnosť kódu

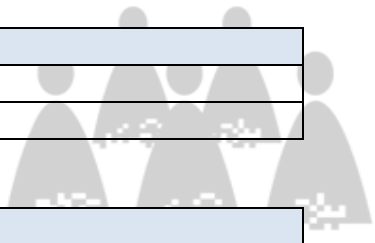
16. Ak sa niečo inicializuje, tak sa použije slovo **initialize**
printer.initializeFontSet();
Skratka **init** by sa nemala používať.

17. Premenné predstavujúce grafický komponent by mali mat sufix s menom komponentu
mainWindow, propertiesDialog, widthScale, loginText, leftScrollbar, mainForm, fileMenu, minLabel, exitButton, yesToggle
Výrazne zvyšuje čitateľnosť a intuitívnosť kódu

18. Ak objekt predstavuje kolekciu (collection) objektov, tak sa použije v názve plurál
vector<Point> points; int values[];

19. Ak objekt predstavuje viac objektov(nie kolekciu) tak sa pomenuje s prefixom **n**
nPoints, nLines

20. Ak premenná predstavuje id číslo entity dáme jej prefix, **No**
tableNo, employeeNo



20. Ak premenná predstavuje id číslo entity dáme jej prefix, **No**
tableNo, employeeNo

21. Iteračné premenné by

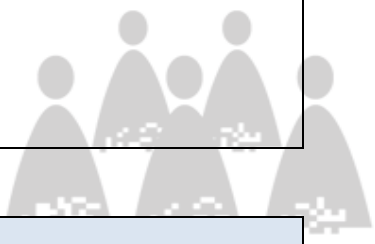
for (int i = 0; i < nTables); i++) {
  :
}
 	
for (vector<MyClass>::iterator i = list.begin(); i != list.end(); i++) {
  Element element = *i;
  ...
}

22. Ak metóda vracia boolean hodNIEu, tak má prefix **is**
isSet, isVisible, isFinished, isFound, isOpen

Vďaka tomuto pravidlu je programátor donútený dávať premenným lepšie názvy ako napr. flag, lebo funkcia isFlag() nedáva zmysel.
 	
Sú aj alternatívy, ktoré sa v niektorých prípadoch hodia viac:
 	
bool hasLicense();
 bool canEvaluate();
 bool shouldSort();

23. Opačné operácie sa musia volať opačnými názvami.
get/set, add/remove, create/destroy, start/stop, insert/delete, increment/decrement, old/new, begin/end, first/last, up/down, min/max, next/previous, old/new, open/close, show/hide, suspend/resume, atď.

24. Nepoužívať skratky
computeAverage(); // NIE: compAvg();
Nepísať ani také skratky, ktoré „lákajú“, napr.: • cmd --> command • cp --> copy • pt --> point • comp --> compute • init --> initialize Ale sú výnimky, ktoré sú známejšie práve pod skratkami: HypertextMarkupLanguage --> html CentralProcessingUnit --> cpu PriceEarningRatio --> pe



25. Pointery sa nazývajú všeobecne

Line* line; // NIE: Line* pLine;
  // NIE: LIne* linePtr;

26. HodNIEy v Enum musia mať všeobecný prefix

enum Color {
  COLOR_RED,
  COLOR_GREEN,
  COLOR_BLUE
};

27. Výnimkové triedy majú sufix **Exception**

class AccessException
{
  :
}

28. Názov funkcie by mal vyjadrovať, čo vracia, názov procedúry čo robí

29. Názvy interface-ov začína veľkými písmenami a na začiatku písmeno **I**, každé ďalšie slovo veľkým
IAgent, IAccountFetcher

30. AK metóda vyhadzuje výnimky, treba ich vymenovať a nie nahradiť všeobecnou „Exception“

3. Pomenovávanie súborov a komentovanie

1. Hlavičkové súbory majú koncovku .h, zdrojové súbory koncovky .c++, .C, .cc or .cpp.
MyClass.c++, MyClass.h

2. Všetky definície sú v zdrojovom súbore

class MyClass
{
public:
  int getValue () {return value_;} // NIE!
  ...
 	
private:
  int value_;
}

3. Oddeľovať logické celky novým riadkom

```
totalSum = a + b + c +
         d + e;

function (param1, param2,
         param3);

setText ("Long line split"
        "into two parts.");

for (int tableNo = 0; tableNo < nTables;
    tableNo += tableStep) {
  ...
}
```

4. Komentáre metód písať v hlavičkových súboroch

5. Komentáre písať ako je naznačené nižšie, aby bolo možne v budúcnosti použiť Doxygen na generovanie dokumentácie

```
/**
 * @file
 * @author  John Doe <jdoe@example.com>
 * @version 1.0
 *
 * @section LICENSE
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
 * General Public License for more details at
 * http://www.gnu.org/copyleft/gpl.html
 *
 * @section DESCRIPTION
 *
 * The time class represents a moment of time.
 */

class Time {

   public:
```

```
    /**
     * Constructor that sets the time to a given value.
     *
     * @param timemillis Number of milliseconds
     *       passed since Jan 1, 1970.
     */
    Time (int timemillis) {
       // the code
    }

    /**
     * Get the current time.
     *
     * @return A time object set to the current time.
     */
    static Time now () {
       // the code
    }
};
```

Nie je potrebné písať všetko ako je naznačené vyššie (napr. file), ale treba krátky slovný popis, popísať parametre a návratovú hodnotu.

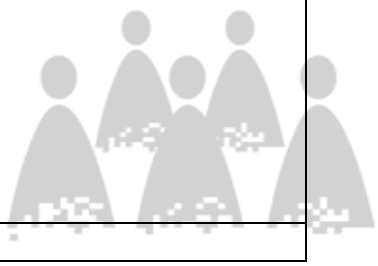
4. Includy

1. Include výrazy by mali byt hierarchicky zoskupene a skupiny oddelene prázdnym riadkom

```
#include <fstream>
#include <iomanip>

#include <qt/qbutton.h>
#include <qt/qtextfield.h>

#include "com/company/ui/PropertiesDialog.h"
#include "com/company/ui/MainWindow.h"
```

2. Include výrazy by mali byt hierarchicky zoskupene a skupiny oddelene prázdnym riadkom

```
#include <fstream>
#include <iomanip>

#include <qt/qbutton.h>
#include <qt/qtextfield.h>

#include "com/company/ui/PropertiesDialog.h"
#include "com/company/ui/MainWindow.h"
```



5.Výrazy

1. Poradie prvkov v triede je public, protected and private
#include <fstream> #include <iomanip> #include <qt/qbutton.h> #include <qt/qtextfield.h> #include "com/company/ui/PropertiesDialog.h" #include "com/company/ui/MainWindow.h"

2. Typovú konverziu je treba robiť explicitne a nespoliehať sa na implicitnú konverziu
floatValue = static_cast<float>(intValue); // NIE: floatValue = intValue;

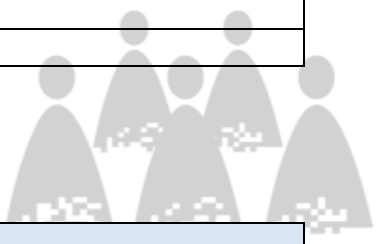
6. Premenné a konštanty

1. Premenné by mali byť nainicializované, kde sú aj deklarované ak je to možné.

2. Implicitný test na nulu sa používa len pri boolean a pointeroch.
if (nLines != 0) // NIE: if (nLines) if (value != 0.0) // NIE: if (value)

3. Konštanty definovať ako **const** a nie pomocou kľúčového slova **DEFINE**
const int pathwidth = 100; //NIE: #define pathwidth 100
Vhodné je konštanty pomenovávať veľkými písmenami + podtrhovník, aby bolo hneď jasne, že sa jedna o konštantu. Čiže: const int PATH_WIDTH = 100;

4. Kľúčového slovo **DEFINE** používať iba v prípade kompilačných nastavení



7. Podmienky

1. Nerobiť komplikované podmienky, radšej podmienku rozdeliť

bool isFinished = (elementNo < 0) || (elementNo > maxElement);
bool isRepeatedEntry = elementNo == lastElement;
if (isFinished || isRepeatedEntry) {
  :

```
}

// NIE:
if ((elementNo < 0) || (elementNo > maxElement)||
    elementNo == lastElement) {
  :
}
```

2. Do podmienky nedávať vykonateľný kód

```
File* fileHandle = open(fileName, "w");
if (!fileHandle) {
  :
}

// NIE:
if (!(fileHandle = open(fileName, "w"))) {
  :
}
```

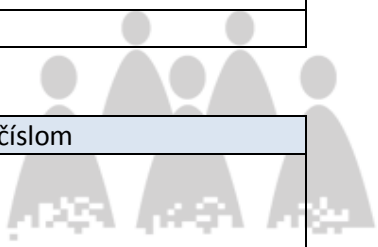
3. Do podmienky nedávať vykonateľný kód

```
File* fileHandle = open(fileName, "w");
if (!fileHandle) {
  :
}

// NIE:
if (!(fileHandle = open(fileName, "w"))) {
  :
}
```

8.Iné

1. Nepoužívať magické čísla

Ak to nie je číslo 1 alebo 0, tak použiť radšej konštanty.



2. Desatinné čísla písať vždy aspoň s jedným desatinným miestom za celým číslom

```
double total = 0.0;    // NIE:  double total = 0;
double speed = 3.0e8;  // NIE:  double speed = 3e8;
double total = 0.5;  // NIE:  double total = .5;

double sum;
:
sum = (a + b) * 10.0;
```

9.Layout

1. Layout bloku musí byť ako je naznačené nižšie

```
//OK
while (!done) {
  doSomething();
  done = moreToDo();
}

//OK
while (!done)
{
  doSomething();
  done = moreToDo();
}

//NOK
while (!done)
  {
    doSomething();
    done = moreToDo();
  }
```

2. Štruktúra triedy musí byť nasledovná

```
class SomeClass : public BaseClass
{
  public:
    ...

  protected:
    ...

  private:
    ...
}
```

3. Štruktúra bliku if-else musí byť nasledovná

```
if (condition) {
  statements;
}

if (condition) {
  statements;
}
```

```
else {
   statements;
}

if (condition) {
  statements;
}
else if (condition) {
  statements;
}
else {
  statements;
}
```

4. Štruktúra bloku *for* musí byť nasledovná

```
for (initialization; condition; update) {
  statements;
}
```

5. Štruktúra bloku *while* musí byť nasledovná

```
while (condition) {
  statements;
}
```

6. Štruktúra bloku *do-while* musí byť nasledovná

```
do {
  statements;
} while (condition);
```

7. Štruktúra bloku *switch* musí byť nasledovná

```
switch (condition) {
  case ABC :
    statements;
    // Fallthrough

  case DEF :
    statements;
    break;

  case XYZ :
    statements;
    break;
```

```
  default :
    statements;
    break;
}
```

8. Štruktúra bloku *try-catch* musí byť nasledovná

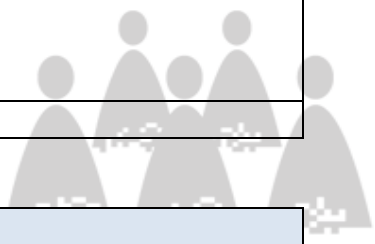
```
try {
  statements;
}
catch (Exception& exception) {
  statements;
}
```

9. Medzery nasledovne

```
a = (b + c) * d; // NIE: a=(b+c)*d

while (true)   // NIE: while(true)
{
  ...

doSomething(a, b, c, d);  // NIE: doSomething(a,b,c,d);

for (i = 0; i < 10; i++) {  // NIE: for(i=0;i<10;i++){
  ...
```

10. Logické bloky kódu oddeliť prázdnym riadkom

```
Matrix4x4 matrix = new Matrix4x4();

double cosAngle = Math.cos(angle);
double sinAngle = Math.sin(angle);

matrix.setElement(1, 1,  cosAngle);
matrix.setElement(1, 2,  sinAngle);
matrix.setElement(2, 1, -sinAngle);
matrix.setElement(2, 2,  cosAngle);

multiply(matrix);
```



11. Zarovnávať treba tam, kde to zvyšuje čitatelnosť

```
if     (a == lowValue)    compueSomething();
else if (a == mediumValue) computeSomethingElse();
else if (a == highValue)   computeSomethingElseYet();
```

```
value = (potential       * oilDensity)   / constant1 +
       (depth           * waterDensity) / constant2 +
       (zCoordinateValue * gasDensity)   / constant3;

minPosition     = computeDistance(min,     x, y, z);
averagePosition = computeDistance(average, x, y, z);

switch (value) {
  case PHASE_OIL   : strcpy(phase, "Oil");   break;
  case PHASE_WATER : strcpy(phase, "Water"); break;
  case PHASE_GAS   : strcpy(phase, "Gas");   break;
}
```

Dodatok B

Metodika pre jednotkové testovanie v prostredí Microsoft Visual Studio 2010 pomocou frameworku GoogleTest

rilohaTestovanie Táto metodika má za úlohu definovať postup inštalácie, konfigurácie, vytvárania a spúšťania jednotkových testov vo vývojovom prostredí Microsoft Visual Studio 2010 pomocou frameworku GoogleTest. Metodika je zameraná primárne na malé tímy vyvýjajúce aplíkáciu v jazyku C++.

Opis vykonávania krokov:

- *Inštalácia a konfigurácia potrebných nástrojov* **??**
- *Návrh a implementácia jednotkových testov* **??**
- *Spustenie jednotkových testov* **??**

v prostredí Microsoft Visual Studio 2010 v jazyku C++ pomocou frameworku GoogleTest.

B.1 Inštalácia a konfigurácia frameworku GoogleTest vo vývojovom prostredí Microsoft Visual Studio 2010

Tento postup sa vykonáva iba jedenkrát - výsledkom je vytvorenie jedného testovacieho projektu spoločného pre všetkých vývojárov. Tí k nemu majú prístup cez

verziovací systém. Testovací projekt pristupuje k zdrojovému kódu aplikácie a obsahuje jednotkové testy pre tento kód.
Systémový administrátor potrebuje nástroje z tabuľky B.1.

Tabuľka B.1: Nevyhnutné nástroje

Názov nástroja	**Opis**	**Dostupnosť**
Microsoft Visual Studio 2010	Vývojové prostredie	Treba zakúpiť
GoogleTest C++	Framework na jednotkové testovanie pre jazyk C++	Voľne dostupné na stiahnutie na [**?**]

Postup inštalácie a konfigurácie:

1. Nainštalovať Microsoft Visual Studio 2010.
2. Vytvoriť adresár napr. gtest, najlepšie priamo v koreňovom adresári operačného systému Windows - **c:\gtest**.
3. Stiahnuť najnovšiu verziu GoogleTest frameworku vo forme *.zip* balíčka.
4. Dekomprimovať stiahnutý balíček.
5. Obsah adresára, do ktorého boli dekomprimované súbory, prekopírovať do predtým (krok 2) vytvoreného adresára **c:\gtest**.
6. Spustiť Microsoft Visual Studio 2010.
7. Vytvoriť projekt (solution), ktorý bude slúžiť na testovanie. Názov projektu je zložený z názvu hlavného projektu a sufixu "Test". Napríklad pri vytváraní testovacieho projektu pre projekt *CrowdSimulation* sa testovací projekt volá *CrowdSimulationTest*. Testovací projekt musí byť typu **Win32 Console Application**.
8. Konfiguráciu testovacieho projektu nastaviť na **Debug**.
9. Cieľovú platformu testovacieho projektu nastaviť na **Win32**.
10. Nastaviť projekt, aby zahŕňal aj adresár (Additional Include Directory) s dekomprimovanými súbormi GoogleTest, konkrétne jeho podadresár *include* - **c:\gtest\include**.
11. Do projektu pridať filter, určený pre zdrojové súbory GoogleTest. Tento filter pomenovať **gtest** a následne doňho pridať vybrať všetky súbory z adresára **c:\gtest\fused-src\gtest**. Filter potom bude obsahovať nasledujúce zdrojové súbory:

- gtest.h
- gtest_main.cc
- gtest_main.cc

12. V testovacom projekte vytvoriť (ak neexistuje) adresár (filter) určený testovaným zdrojovým súborom s názvom *Source Files*, do ktorého sa budú vkladať zdrojové súbory určené na testovanie.

13. V testovacom projekte vytvoriť (ak neexistuje) adresár (filter) určený testovacím zdrojovým súborom s názvom *Test Files*, do ktorého sa budú vkladať zdrojové súbory obsahujúce testy.

B.2 Pridanie zdrojových kódov do testovacieho projektu

1. Vložiť zdrojové súbory určené na testovanie do projektu, do adresára na to určeného (B.1).

2. Nastaviť testovací projekt, aby zahŕňal adresár (Additional Include Directory) so zdrojovými súbormi, ktoré boli pridané do projektu v predošlom kroku.

B.3 Vytváranie jednotkových testov pomocou GoogleTest Frameworku

Postup vytvárania jednotkových testov:

1. V testovacom projekte, v adresári určenom pre testovacie zdrojové súbory (B.1) vytvoriť zdrojový súbor s rovnakým názvom, aký má súbor, ku ktorému vytvárame test.

2. Jednotkový test sa vytvára ako je znázornené v ukážke B.1:

Výpis kódu B.1: Jednotkový test vo frameworku GoogleTest

```
TEST(test_case_name, test_name) {
    Telo testu
}
```

kde názov *test_case_name* označuje všeobecný názov testovacieho prípadu (napr. názov testovanej funkcie) a *test_name* označuje konkretizáciu testovanej funkcionality (napr. HandleZeroOutput).

3. Do tela testovacej metódy pridať zdrojový kód obsahujúci porovnávania očakávanej a vrátenej hodnoty. Na porovnávanie používať funkciu frameworku $EXPECT\_*$ vždy keď je to možné. Funkciu $ASSERT\_*$ používať iba vo výnimočných prípadoch, kedy by prípadné neprejdenie testu znamenalo, že nemá zmysel pokračovať v testovaní. Viac o jednotlivých použitiach funkcií $ASSERT\_*$ a $EXPECT\_*$ v [**?**].

Príklad zdrojového kódu obsahujúceho jednotkové testy je v ukážke B.2.

Výpis kódu B.2: Séria jednotkových testov vo frameworku GoogleTest

```
#include <iostream>
#include "gtest/gtest.h"
#include "Class.h"

TEST(subtraction, lessThanZero){
        CClass class;
// EXPECT_LT(expected, actual) -> expected < actual
        EXPECT_LT(0, class.subtraction(2,4));
        EXPECT_LT(0, class.subtraction(0,7));
        EXPECT_LT(0, class.subtraction(1,2));
}

TEST(subtraction, rightResult){
        CClass class;
        EXPECT_LT(4, class.subtraction(6,2));
        EXPECT_LT(9, class.subtraction(10,1));
}
```

B.4 Spustenie jednotkových testov

Postup spúšťania jednotkových testov:

1. Skompilovať projekt. V prípade neúspešnej kompilácie je potrebné overiť správne nakonfigurovanie projektu na využívanie frameworku GoogleTest (B.1).

2. Spustiť testovací projekt bez debuggovania. Tým sa spustí funkcia, ktorá je obsiahnutá v zdrojových súboroch GoogleTest. Výsledok je spustenie všetkých jednotkových testov, ktoré boli vytvorené ako GoogleTest jednotkové testy (B.3). Výsledky testovania sa zobrazia v konzole (ukážka B.3).

Výpis kódu B.3: Ukážka výsledkov spustenia jednotkových testov vo frameworku GoogleTest

```
Running main() from gtest_main.cc
[==========] Running 2 tests from 2 test cases.
[----------] Global test environment set-up.
[----------] 1 test from sample_test_case
[ RUN      ] sample_test_case.sample_test
[       OK ] sample_test_case.sample_test (0 ms)
[----------] 1 test from sample_test_case (1 ms total)

[----------] 1 test from sample_test_case_int
[ RUN      ] sample_test_case_int.gimeIntTest
[       OK ] sample_test_case_int.gimeIntTest (0 ms)
[----------] 1 test from sample_test_case_int (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 2 test cases ran. (3 ms total)
[  PASSED  ] 2 tests.
Press any key to continue . . .
```

Dodatok C

Metodika pre zadávanie úlohy v systéme Redmine

Účelom tejto metodiky je zadefinovanie postupu pri zadávaní úlohy v plánovacom systéme Redmine. Metodika je vhodná pre malé až stredne veľké tímy, pretože sa riadi agilným spôsobom vývoja softvéru SCRUM. Predpokladá sa, že jednotlivé požiadavky sú už zadefinované a procesy ako napĺňanie backlogov produktu, vydaní a šprintov sú predmetom iných metodík.

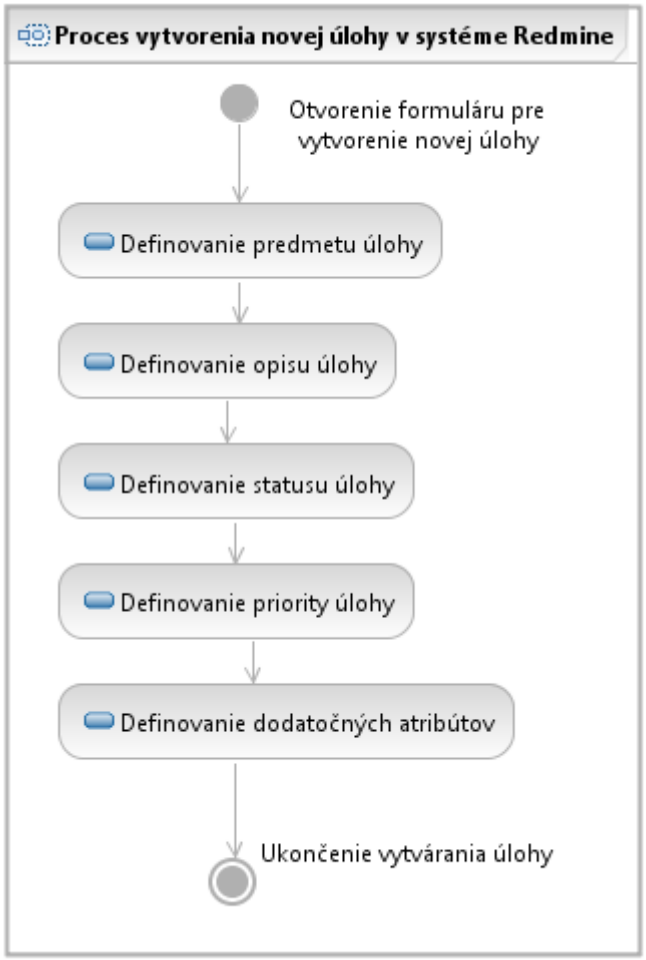
Pridávanie úlohy má na starosti produktový vlastník (v kontexte tímového projektu človek majúci na starosti správu systému Redmine), ktorý na základe požiadavky definovanej od zákazníka (ostatných členov tímového projektu) formuluje úlohu, ktorú následne zadáva do systému Redmine prostredníctvom webového prehliadača.

C.1 Diagram procesu vytvorenia novej úlohy v systéme Redmine

Ako možno vidieť na obrázku C.1, proces pridania úlohy do systému Redmine bol rozdelený do niekoľkých menších podúloh (akcií), ktoré sú uvedené v Tabuľka 2. V tabuľke sú taktiež uvedené čísla podkapitol, ktoré sa podrobnejšie zaoberajú danými činnosťami.

C.2 Otvorenie formuláru pre vytvorenie úlohy

Po prihlásení do systému Redmine a korektnom vytvorení projektu sa formulár pre pridanie novej úlohy zobrazí prostredníctvom záložky “New issue” (“Nová úloha”). V systéme je následne v poli “Tracker” (“Fronta”) na výber niekoľko možností (tabuľka



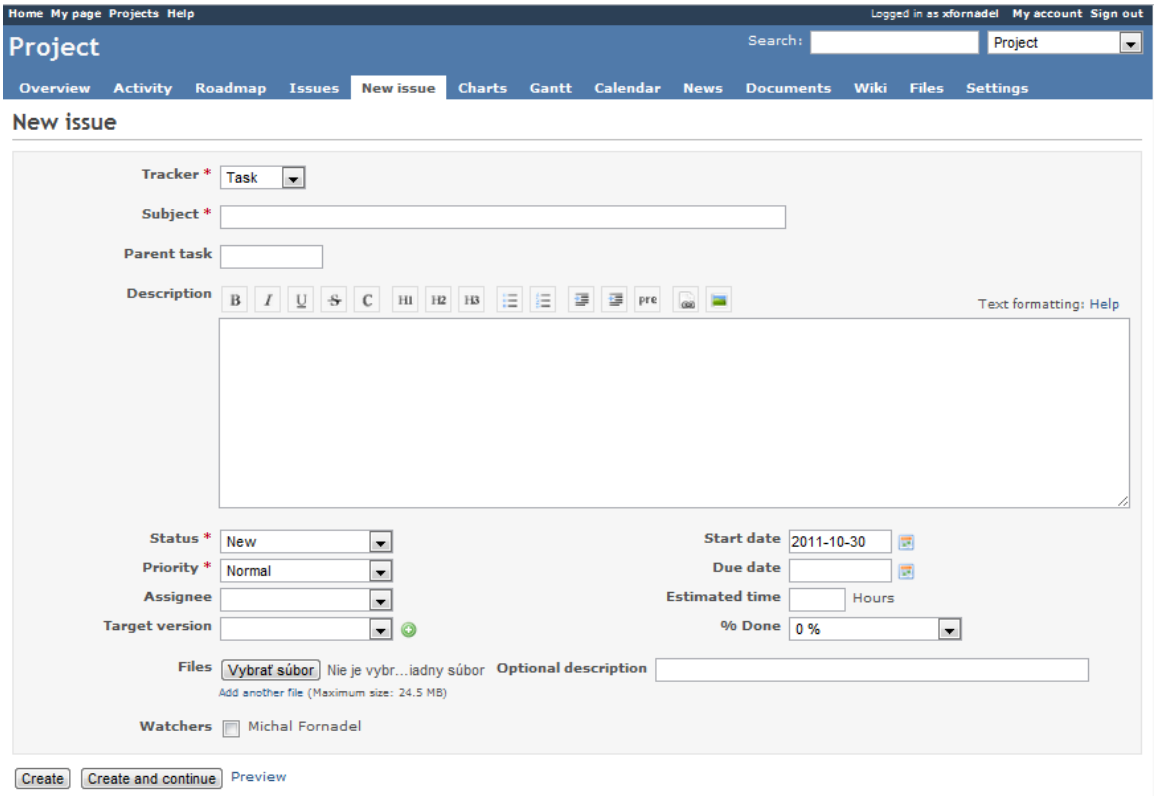
Obr. C.1: Diagram procesu vytvorenia novej úlohy v systéme Redmine

C.1). Vzhľadom na to, že predmetom tejto metodiky je manažment úloh, z ponúkaných možností sa vyberá typ "Task" ("Úloha"). Formulár pre tento typ zodpovedá formuláru na obrázku C.2.

Tabuľka C.1: Typy úloh v systéme Redmine

Možnosť	**Opis**
Požiadavka (User story)	Všeobecný opis požiadavky zákazníka (nie je predmetom tejto metodiky)
Implementácia (Feature)	Opis implementácie analýzy a návrhu (nie je predmetom tejto metodiky)
Úloha (Task)	Opis rôznych druhov úloh (analýza požiadavky, návrh riešenia, testovanie) (je predmetom tejto metodiky)

Chyba (Bug)	Opis chyby s odkazom na implementáciu (nie je predmetom tejto metodiky)
Podpora projektu (Support)	Záležitosti okolo projektu (réžia tímu, web tímu, štúdium technológií) - nie je predmetom tejto metodiky



Obr. C.2: Formulár pre pridanie novej úlohy v systéme Redmine

C.3 Definovanie predmetu úlohy

Predmet úlohy sa uvádza v poli "Subject" ("Predmet"). Je definovaný stručne a výstižne za účelom jednoduchej a rýchlej identifikácie. Pred samotným názvom predmetu sa nachádza identifikátor v hranatých zátvorkách. Vzhľadom na to, že úloha je v kontexte tejto metodiky chápaná ako opis analýzy požiadavky, návrhu riešenia alebo návrhu testovacích scenárov, do hranatých zátvoriek identifikátora sa uvádza referencia na požiadavku zákazníka (user story).

Napríklad, predmet úlohy odkazujúci sa na požiadavku od zákazníka môže vyzerať nasledovne: *"[#23] Návrh komunikácie medzi agentom a prostredím"*

C.4 Definovanie opisu úlohy

Pole "Description" ("Popis") sa používa pre definovanie opisu úlohy. Opis tvorí ucelený text štruktúrovaný do logických celkov (odrážok, odstavcov), ktorý môže byť dodatočne formátovaný za účelom zvýraznenia podstatných častí (tučné písmo, kurzíva, podčiarknuté písmo). Nie je však odporúčané používať príliš mnoho štýlov formátovania, aby sa predišlo zníženiu čitateľnosti a formálnosti písaného opisu.

C.5 Definovanie statusu úlohy

Pri vytváraní úlohy v systéme sa aktuálny stav uvádza v poli "Status" ("Stav"). Ak je úloha pridávaná do systému nová, t.j. úloha je identifikovaná na základe požiadavky a neexistujú žiadne rozpracovania konkrétnej problematiky, stav sa volí ako "New" ("Nová"). V opačnom prípade je stav (status) úlohy definovaný podľa tabuľky C.2.

C.6 Definovanie statusu úlohy

Tabuľka C.2: Stavy úlohy v systéme Redmine

Status úlohy	**Opis**
Nová (New)	Na novovytvorenej úlohe ešte nikto nepracuje
Pridelená (Assigned)	Úloha je pridelená konkrétnemu členovi
Vykonaná (Resolved)	Úloha je vykonaná a pripravená na kontrolu a zhodnotenie iného člena tímu
Čakajúca na odpoveď (Feedback)	V rámci úlohy existujú otázky, ktoré musia byť zodpovedané, aby sa mohlo ďalej pokračovať
Ukončená (Closed)	Úloha je akceptovaná produktovým vlastníkom aj zákazníkom a je uzavretá
Zrušená (Rejected	Úloha je odmietnutá zákazníkom
Rozpracovaná (In Progress)	Na úlohe sa pracuje

C.7 Definovanie priority úlohy

Za účelom definovania dôležitosti úlohy sa musí zvoliť hodnota poľa "Priority" ("Priorita"). Stupeň dôležitosti je škálovaný do piatich stupňov, ktoré sú uvedené v tabuľke C.3. Z hľadiska bežného fungovania projektu sa najviac používajú prvé tri typy priorít. Používanie ostatných dvoch je charakteristické iba pre výnimočné situácie.

Tabuľka C.3: Typy priority v systéme Redmine

Typ priority	**Použitie**
Nízka (Low)	Nepovinné úlohy
Stredná (Normal)	Bežné úlohy
Vysoká (High)	Úlohy s veľkým významom pre projekt
Urgentná (Urgent)	Úlohy, ktoré sú prednostne vykonávané pred ostatnými úlohami s menšou prioritou. Jedná sa väčšinou o úlohy, ktoré boli dodatočne identifikované a musia byť čo najskôr riešené.
Okamžitá (Immediate)	Úlohy, ktoré sa musia okamžite riešiť (ostatné úlohy s nižšou prioritu sú automaticky pozastavené), pretože by mohli mať vplyv na celkový chod projektu. Jedná sa väčšinou o vykonané úlohy, ktoré neprešli fázou kontroly.

C.8 Definovanie dodatočných atribútov

Pre novovytváranú úlohu sa v systéme definujú aj atribúty uvedené v tabuľke C.4. Všetky sú nepovinné, ale v rámci jednoznačnej identifikácie úlohy sa väčšina z nich vždy definuje. Výnimku tvoria iba "Súbory", ktoré nemusia existovať alebo nemusia byť k dispozícií v čase vytvárania úlohy.

Tabuľka C.4: Dodatočné atribúty v systéme Redmine

Typ atribútu	**Opis**
Priradené (Assignee)	Zo zoznamu uvedených mien sa vyberá člen tímu, ktorému je úloha pridelená

Priradené k verzii (Target version)	Zo zoznamu ponúkaných verzií sa vyberá tá, pre ktorú je úloha vytváraná (verzie v kontexte ponímania SCRUM metódy korešpondujú so šprintmi)
Začiatok (Start date)	Uvádza sa dátum vytvorenia danej úlohy
Uzavrieť do (Due date)	Uvádza sa dátum, do ktorého musí byť úloha ukončená
Odhadovaná doba (Estimated time)	Odhadovaný čas na úspešne dokončenie členom tímu, ktorý pracuje na úlohe
% Hotovo (% Done)	Percentuálne vyjadrenie progresu prác na úlohe
Súbory (Files)	Priložené súbory, ktoré sú relevantné v kontexte definovanej úlohy (externé analýzy, externé návrhy podobných riešení, diagramy)
Pozorovatelia (Watchers)	Zoznam ľudí, ktorí budú dostávať notifikačné maily reportujúce postupy práce na úlohe