SLOVENSKÁ TECHNICKÁ UNIVERZITA BRATISLAVA
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

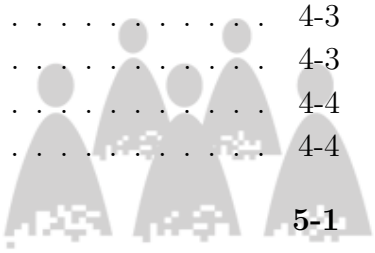
SIMULÁCIA DAVU

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 bola prezentovane 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 jednotlivé šprinty naplánované

postupne do týždňov, ktoré zahŕňajú týkajúce sa úlohy spolu s menom zodpovedného člena tímu za ich plnenie a výhradeného času na ich splenenie.

Kapitola 4 obsahuje zoznam týkajúci sa úloh členov v tíme. Sú tu vymenované identifikované roly členov, krátkodobé ako aj dlhodobé úlohy.

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

Metodiky potrebné pri vývoji sú rozobraté v kapitole 6. Jedná sa predovšetkým o dodržiavanie pravidiel v rámci definovania úloh v systéme Redmine a verziovania súborov prostredníctvom SVN.

V kapitole 7 sú priložené záznamy zo stretnutí počas jednotlivých týždňov vývoja aplikácie. Zápisnice obsahujú potrebné informácie o stave plnenia navrhovaných úloch 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:

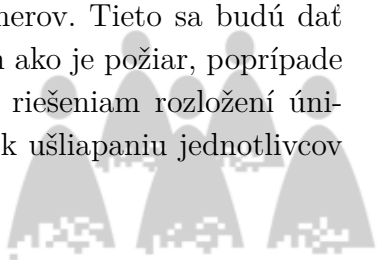
- Problém priradenia viet - viaceré vety môžu byť preložené na viacej spôsobov
- Zložené slová
- Frázy - slovné frázy sa dajú preložiť viacerými spôsobmi vzhľadom na kontext použitia
- Pomiešaný slovosled
- Neznáme slová - slová existujúce a používané mimo slovník
- Syntax

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

Kapitola obsahuje plán jednotlivých šprintov spolu s úvodným rozbehom počas dvanástich týždňov zimného semestra. V tabuľke 3.1 možno pozorovať podľa jednotlivých týždňov sumarizáciu vykonávaných činností spolu s identifikovanými šprintami.

2. - 4. týždeň	Rozbeh	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
4. - 6. týždeň	Šprint#1	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
6. - 8. týždeň	Šprint#2	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
10. týždeň	-	Vytvorenie základnej fyzikálnej simulácie, vytvorenie navigácie cez navigačnú geometriu
12. týždeň	-	Vytvorenie implementácie vnútornej mapy na základe navigačnej geometrie, implementovanie FlowField algoritmu pre pohyb agentov

Tabuľka 3.1: Rozbeh a naplánované šprinty

3.1 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.

3.2 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.

3.3 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	-
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.

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**
Príprava webovej stránky	-	-	-	100%	-	-
Otestovanie funkčnosti offline verzie stránky	-	-	-	100%	-	-

Vytvorenie google skupiny	-	-	-	100%	-	-
Vytvorenie plagátu tímu	-	-	-	-	100%	-
Vytvorenie šablóny pre zápisy	-	-	-	100%	-	-
Príprava dokumentu použitých technológií	-	-	-	100%	-	-
Vytvorenie šablóny pre dokumentácie	-	-	-	-	100%	-
Refactoring a review zdrojového kódu prototypu	-	-	70%	30%	-	-
Vytvorenie modelu tried pre prototyp	-	100%	-	-	-	-
Dopracovanie modelu tried	-	-	100%	-	-	-
Pripomienkovanie metodiky pre písanie zdrojového kódu	25%	25%	25%	-	-	25%
Vytvorenie návodu pre doinštalovanie Visual Assist do Visual Studia	-	-	-	100%	-	-
Vytvorenie základnej sady máp vo formáte DXF	-	-	-	-	-	100%
Vytvorenie návodu pre Boost	-	50%	50%	-	-	-
Vytvorenie návodov pre Google Test	-	-	100%	-	-	-
Dodatočná analýza koncepcie FlowField	-	-	100%	-	-	-

Kontaktovanie hodnotiteľa kvality	-	100%	-	-	-	-
Dodatočná analýza vektorovej grafiky	-	100%	-	-	-	-
Pridanie obsahu webovej stránky	-	100%	-	-	-	-
Optimalizácia spustenia DLL modulov	-	100%	-	-	-	-

Tabuľka 4.2: Zoznam krátkodobých úloh

4.3 Dlhodobé úlohy

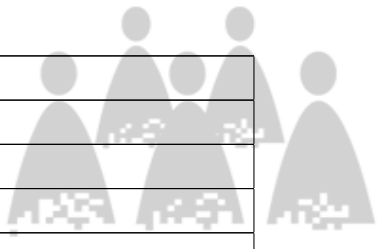
V tabuľke 3.1 sú uvedené dlhodobé úlohy v time spolu s percentuálny vyjadrením podielu práce konkrétnych členov na danej činnosti.

Úloha	**Pomothy**	**Petráš**	**Košický**	**Pavlech**	**Fornádeľ**	**Hlaváč**
Webová stránka	-	-	-	100%	-	-
Systém Redmine	-	-	-	-	100%	-
Prepis do Latex-u	-	-	-	-	100%	-
Implementácia	18%	18%	18%	18%	10%	18%

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**
	Šablona dokumentácie	Michal Fornádeľ
	Prepis textu do Latexu	Michal Fornádeľ
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
4	Úlohy	Michal Fornádeľ
5	Prostriedky technickej podpory	Michal Fornádeľ
6	Metodiky potrebné pri vývoji	Michal Fornádeľ
7	Záznamy zo stretnutí	Všetci
A	Metodika vytvárania zdrojového kódu	Adam Pomothy

4.4.3 Projektová dokumentácia

Kapitola	**Názov**	**Autor**
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 SVNka 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	
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

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.

- **TortoiseSVN** - subversion klient, 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 voice** - hlavným prostriedkom pre hlasovú komunikáciu. Vďaka tomuto nástroju sú v prípade potreby členovia okamžite schopní vytvoriť hlasové konferencie.

Kapitola 6

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.

6.1 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”.

6.2 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
- úlohy na ďalšie stretnutie

6.3 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.

Kapitola 7

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
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

Vyhodnotenie úloh z predchádzajúceho stretnutia

1. Vybrať meno tímu (úloha 1.1) – splnené
2. Vytvoriť plagát tímu (úloha 1.2)– splnené
3. Vybratie servera (1.3) – splnené
4. Vytvorenie šablóny pre zápisy (1.4) – splnené
5. Inštalácie linuxového servera (úloha 1.5)– splnené
6. Vytvorenie web prezentácie projektu (úloha 1.6)– splnené
7. Sprevádzkovanie Redmine (úloha 1.7)– splnené
8. Sprevádzkovanie SVN (úloha 1.8) – nesplnené
9. Analýza súčasných prostredí (úloha 1.9)– splnené, treba dať do jedného dokumentu
10. Návrh rozhrania pre simulačné prostredie a agentov (úloha 1.10)–na úlohe sa stále pracuje

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

Úlohy na ďalšie stretnutie

Číslo úlohy	ID úlohy v Redmine	Popis úlohy	Kto	Termín ukončenia
2.1	11	Sprevádzkovanie SVN	Adam Pomothy	12.10.2011
2.2	78	Analýza súčasných prostredí	Všetci	12.10.2011

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

Vyhodnotenie úloh z predchádzajúceho stretnutia

1. Analýza súčasných prostredí (úloha 2.2) – splnené
2. Sprevádzkovanie SVN (úloha 2.1) – splnené

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

Úlohy na ďalšie stretnutie

Číslo úlohy	ID úlohy v Redmine	Popis úlohy	Kto	Termín ukončenia
3.1	16	Code Guidelines	Adam Pomothy	26.10.2011
3.2	17	Základná funkcionalita agenta	Michal Fornádeľ	26.10.2011
3.3	18	Základná funkcionalita prostredia	Lukáš Pavlech	26.10.2011
3.4	19	Komunikácia prostredia a agentov	Daniel Petráš	26.10.2011
3.5	20	Protokol agentových akcií a akcií prostredia	Daniel Petráš	26.10.2011
3.6	21	Analýza formátov máp	Marek Hlaváč	26.10.2011
3.7	22	Základná vizualizácia	Martin Košický	26.10.2011
3.8	25	Nájsť konvertor OSM -> CAD	Marek Hlaváč	26.10.2011

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.

Vyhodnotenie úloh z predchádzajúceho stretnutia

1. Code Guidelines (úloha 3.1) – splnená
2. Analýza formátov máp (úloha 3.6) – splnená
3. Nájsť konvertor OSM -> CAD (úloha 3.8) – splnená

   Ostatné úlohy sú rozpracované.

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)

Úlohy na ďalšie stretnutie

Číslo úlohy	ID úlohy v Redmine	Popis úlohy	Kto	Termín ukončenia
4.1	24	Základná implementácia mapy	Marek Hlaváč	26.10.2011
4.2	26	Zistiť formát mapy v projekte VirtFIIT	Lukáš Pavlech	26.10.2011
3.2	17	Základná funkcionalita agenta	Michal Fornádeľ	26.10.2011
3.3	18	Základná funkcionalita prostredia	Lukáš Pavlech	26.10.2011
3.4	19	Komunikácia prostredia a agentov	Daniel Petráš	26.10.2011
3.5	20	Protokol agentových akcií a akcií prostredia	Daniel Petráš	26.10.2011
3.7	22	Základná vizualizácia	Martin Košický	26.10.2011

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)

Vyhodnotenie úloh z predchádzajúceho stretnutia

1. Základná implementácia mapy (úloha 4.1) – splnené
2. Zistiť formát mapy v projekte VirtFIIT (úloha 4.2) – splnené
3. Základná funkcionalita agenta (úloha 3.2) – splnené
4. Základná funkcionalita prostredia (úloha 3.3) – splnené
5. Komunikácia prostredia a agentov (úloha 3.4) – splnené
6. Protokol agentových akcií a akcií prostredia (úloha 3.5) – splnené sčasti, chýba dokumentácia
7. Základná vizualizácia (úloha 3.6) - splnené

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

Úlohy na ďalšie stretnutie

Číslo úlohy	ID úlohy v Redmine	Popis úlohy	Kto	Termín ukončenia
5.1	36	Simulovanie pohľadu agenta	Martin Košický, Michal Fornádeľ	9.11.2011
5.2	35	Hľadanie dverí zo strany agenta	Adam Pomothy	9.11.2011
5.3	33	Vytvorenie definovanej mapy a doplnenie vrstvy pre reprezentáciu dverí	Marek Hlaváč	9.11.2011
5.4	31	Vytvorenie koncepcie oddelených modulov (DLL moduly)	Martin Košický	9.11.2011
5.5	30	Detekcia pretínajúcich sa stien	Daniel Petráš	9.11.2011
5.6	28	Generovanie agentov do mapy	Lukáš Pavlech	9.11.2011

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

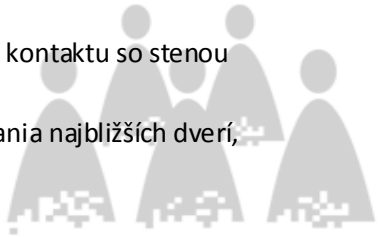
Dopracovávanie úloh ohľadom inteligencie agenta a prostredia, prezentovanie myšlienky pokročilých súčasných techník pre simulovanie davu (NavigationMesh, FlowField Continuum Crowds koncept)

Vyhodnotenie úloh z predchádzajúceho stretnutia

1. Simulovanie pohľadu agenta (úloha 5.1) - nesplnená
2. Hľadanie dverí zo strany agenta (úloha 5.2) - nesplnená
3. Vytvorenie definovanej mapy a doplnenie vrstvy pre reprezentáciu dverí (úloha 5.3) - splnená
4. Vytvorenie koncepcie oddelených modulov (DLL moduly) (úloha 5.4) – splnená sčasti, niektoré moduly ešte neobsahujú funkcionalitu ako v doterajšej verzii projektu
5. Detekcia pretínajúcich sa stien (úloha 5.5) – nesplnená, externá knižnica riešiaca funkcionalitu neidentifikovaná, potreba implementácie vlastného riešenia
6. Generovanie agentov do mapy (úloha 5.6) – splnená

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)



Úlohy na ďalšie stretnutie

Číslo úlohy	ID úlohy v Redmine	Popis úlohy	Kto	Termín ukončenia
5.1	36	Simulovanie pohľadu agenta	Martin Košický, Michal Fornádeľ	9.11.2011
5.2	35	Hľadanie dverí zo strany agenta	Adam Pomothy	9.11.2011
5.4	31	Vytvorenie koncepcie oddelených modulov (DLL moduly)	Martin Košický	9.11.2011
5.5	30	Detekcia pretínajúcich sa stien	Daniel Petráš	9.11.2011
6.1	34	Plánovanie pohybu agenta	Marek Hlaváč	9.11.2011
6.2	32	Integrácia vyhotovenej dokumentácie do TeX šablóny	Michal Fornádeľ	9.11.2011
6.3	29	Riešenie kolízii agentov	Lukáš Pavlech	9.11.2011

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

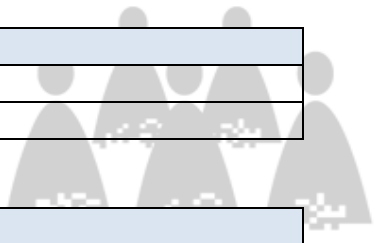
16. Ak sa niečo inicializuje, tak sa použije slovo **initialize**
printer.initializeFontSet();
Skratka **init** by sa nemala používať.

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

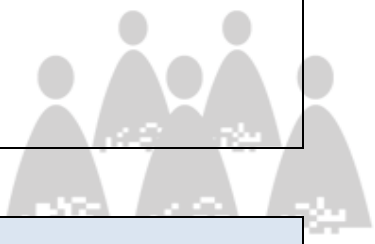
for (int i = 0; i < nTables); i++) {
  :
}
 	
for (vector<MyClass>::iterator i = list.begin(); i != list.end(); i++) {
  Element element = *i;
  ...
}

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.

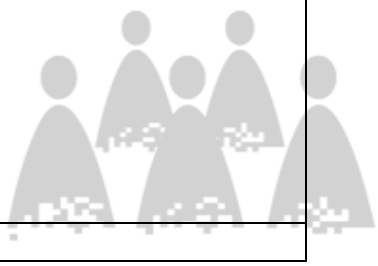
4. Includy

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;

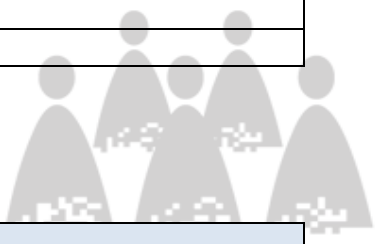
6. Premenné a konštanty

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"))) {
  :
}
```

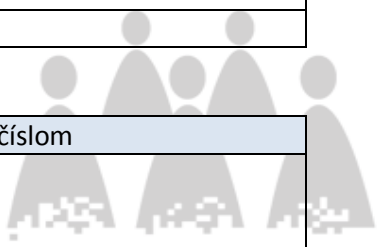
3. Do podmienky nedávať vykonateľný kód

```
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

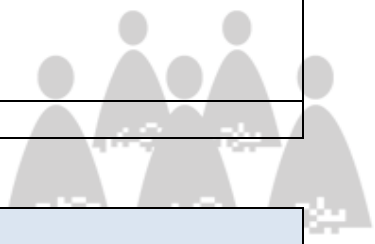
```
try {
  statements;
}
catch (Exception& exception) {
  statements;
}
```

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;
}
```