FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

IMAGINE CUP 2007

Dokument riadenia projektu

Vedúci projektu:	*prof. Ing. Mária Bieliková PhD.*	*Autori :*	*Bc. Andrej Frlička*
Študijný odbor	*Softvérové inžinierstvo*		*Bc. Marek Tomša*
Akademický rok:	*2006/2007*		*Bc. Richard Veselý*
Ročník:	*1 Inžiniersky*		*Bc. Oto Vozár*
Semester:	*Zimný*		

Obsah

0 Úvod

Kapitola zhŕňa základné informácie o dokumente riadenia projektu. Obsahuje informácie o účele a rozsahu dokumentu ako aj prehľad dokumentu.

0.1 Účel a rozsah dokumentu

Dokument predstavuje základnú informačnú brožúru riadenia projektu. Slúži ako hlavný informačný výstup riadenia projektu a zároveň ako podklad ku hodnoteniu v predmete tvorba softvérového systému v tíme I.

0.2 Prehľad dokumentu

Prvú kapitolu dokumentu tvorí ponuka, za ktorou nasledujú plány projektu úlohy členov tímu ku jednotlivým kontrolným bodom. V ďalšej kapitole sú uvedené záznamy zo stretnutí.

Nasledujúca kapitola sumarizuje štandardy kódovania, ktoré využívajú členovia tímu pri tvorbe zdrojového kódu.

V poradí šiesta kapitola obsahuje posudky, ktoré členovia vytvorili pre konkurenčný tím ako aj posudky druhého tímu na vytvorenú projektovú dokumentáciu. Zároveň zahŕňa aj vyjadrenie sa ku posudzovaniu konkurenčným tímom

V predposlednej kapitole sa nachádzajú preberacie protokoly.

Dokument uzatvára kapitola, v ktorej sa nachádzajú informácie o práci na projektovej dokumentácii.

1 Ponuka

Kapitola obsahuje plné znenie dokumentu ponuky tímu v rámci tímového projektu. Číslovanie kapitol, ako aj formátovanie bolo upravené pre potreby dokumentu riadenia.

1.1 Úvod

Dokument predstavuje ponuku tímu na tému Imagine Cup v rámci predmetu Tvorba softvérového systému v tíme. Prvá kapitola uvádza našu motiváciu a zloženie tímu, druhá predstavuje návrh riešenia vychádzajúci z analýzy úspešných projektov a nápadov, ktoré vzišli počas stretnutí tímu vo forme sumáru konceptov a ideí a tretia uvádza predpokladané zdroje potrebné pri riešení. V prílohe uvádzame preferencie výberu ostatných ponúkaných tém a rozvrh jednotlivých členov tímu.

1.2 Motivácia a zloženie tímu

V tejto kapitole uvádzame našu motiváciu k riešeniu témy Imagine Cup v rámci predmetu Tvorba softvérového systému v tíme. V druhej časti kapitoly sú informácie o záujmoch, skúsenostiach a schopnostiach jednotlivých členov tímu v kontexte riešenia zadania projektu.

1.2.1 Motivácia

Na túto tému sme sa prihlásili preto, lebo pred dvomi rokmi sme vyhrali československé kolo spomínanej súťaže v Budapešti, minulý rok slovenské kolo a zúčastnili sme sa východoeurópskeho finále v Maribore a tento rok sa chceme ísť pozrieť do Soulu.

Veríme, že sme schopní vytvoriť tím s najlepšími predpokladmi na to, aby spolu s podporou fakulty vytvoril projekt, ktorý bude mať vysoké šance na úspech v súťaži Imagine Cup.

Všetci štyria členovia majú bohaté skúsenosti s platformou .NET, čo je esenciálna podmienka pre účasť v tejto súťaži. Traja z členov tímu pracujú spolu už štvrtý rok a tvoria veľmi zohraný tím. Bakalárske štúdium ukončili spoločnou účasťou na projekte CSIDC, s ktorým sa zúčastnili a mali úspech v minulom ročníku súťaže Imagine Cup, kedy postúpili do východoeurópskeho finále, kde nadobudli neoceniteľné skúsenosti a oboznámili sa s mnohými projektmi, ktoré boli úspešné aj v celosvetovom finále. Tiež sa zúčastnili súťaže WESC, čo dokazuje ich pripravenosť pracovať s plným nasadením.

Marek Tomša, Richard Veselý a Oto Vozár, pôsobia ako Microsoft Student Partneri, čo im zabezpečuje prístup k technológiám a zdrojom od firmy Microsoft z prvej ruky. Vďaka účasti v tomto programe a osobnej angažovanosti mali šancu oboznámiť sa s pre-release verziami platformy .NET 3.0 a tieto novonadobudnuté skúsenosti sú pripravení aplikovať pri prácach na implementácii projektu.

Štvrtým členom je kreatívny človek s veľkým potenciálom schopný nadchnúť sa a pracovať tak, ako to účasť v tejto súťaži vyžaduje.

1.2.2 Členovia tímu

Andrej Frlička – dizajnér, dokumentarista

Bakalárske štúdium ukončil projektom využívajúcim technológiu .NET v n-vrstvovej distribuovanej aplikácii, pri ktorej riešení získal skúsenosti s .NETom na všetkých úrovniach. Získal tiež pochvalný list dekana za vynikajúco vypracovanú bakalársku prácu.

K riešeniu problémov pristupuje s nadšením a vysokou osobnou angažovanosťou. Je to kreatívna osobnosť s dobre rozvinutým umeleckým cítením. Venuje sa skladaniu elektronickej hudby, editácii zvuku a pedagogickej činnosti. Má bohaté skúsenosti s prácou v grafických a hudobných editačných programoch a s tvorbou webových stránok a rozsiahle skúsenosti s tvorbou používateľskej dokumentácie.

Ovláda programovacie jazyky C/C++, C#, Java a technológie Windows Forms, ASP.NET, SQL Server a ASP.NET Web Services.

Vo voľnom čase sa venuje behaviorálnej analýze.

Marek Tomša – manažér, architekt

Bakalárske štúdium ukončil cum laude obhájením bakalárskej práce na tému CSIDC 2006, ktorú riešil v tíme spolu s Richardom Veselým, Otom Vozárom a Michalom Dobišom.

Z programovacích jazykov ovláda jazyk C# a má hlboké znalosti platformy .NET. Z riešenia minulých projektov má skúsenosti s vedením tímu a s písaním technickej a prezentačnej dokumentácie.

Z technológií preferuje klientské technológie ako Windows Forms a Windows Presentation Foundation, ale má skúsenosti aj so serverovými technológiami MS SQL Server, a ASP.NET Web Services a s vývojom pre malé zariadenia. Má skúsenosti s pre-release verziami platformy .NET vo verzii 3.0, ktoré plánuje uplatniť pri riešení tohto projektu.

Vo voľnom čase sa venuje práci s handicapovanými ľuďmi.

Richard Veselý – kódič

Bakalárske štúdium ukončil riešením projektu CSIDC 2006. Pracuje ako programátor v nadnárodnej spoločnosti na enterprise projektoch nasadených v zahraničí. Jeho najsilnejšou stránkou je schopnosť veľmi vysokého pracovného nasadenia pri implementácii. Je schopný naučiť sa akúkoľvek novú technológiu v priebehu projektu a nové skúsenosti okamžite aplikovať pri implementácii.

Ovláda programovacie jazyky C#, VB (VB6, VB.NET, VBA, VBScript...) a Java. Má skúsenosti s klientskými technológiami Windows Forms, ADO.NET, serverovými technológiami MS SQL Server 2000/2005, IIS, ASP.NET 1.0/1.1/2.0 a Web Services, webovými technológiami (D/X)HTML, CSS, JavaScript, AJAX, s XML, XPath, XQuery, XSLT, regulárnymi výrazmi, a s mnohými ďalšími bude mať čoskoro.

Medzi jeho kvality patrí tiež vynikajúca schopnosť prezentovať v anglickom jazyku, čo je veľmi dôležitý aspekt pre úspech tímu v súťaži Imagine Cup.

Vo voľnom čase wrappuje webové stránky.

Oto Vozár – algoritmik, databázový špecialista, grafik

Bakalárske štúdium ukončil cum laude obhájením bakalárskej práce, týkajúcej sa problematiky vizualizácie neurónových sieti.

Dokáže pracovať s nesmiernym zápalom a svojou usilovnosťou ženie tím dopredu. Rád dotiahne všetko do posledného detailu. V tíme často zastáva úlohu grafika, pričom aplikuje svoj rozvinutý umelecký cit.

Z programovacích jazykov preferuje jazyk C# a platformu .NET, pričom v rámci projektov pracoval najmä s technológiami Windows Forms, ASP.NET, SQL, ADO.NET, ASP.NET Web Services a WCF, GDI+ a DirectX.

Vo voľnom čase hrá na akordeón a študuje 3D algoritmy a enginy.

1.3 Imagine Cup – Imagine a world where technology enables a better education for all

Špecifikom súťaže Imagine Cup je, že zadanie témy dáva veľmi široké možnosti výberu konkrétneho riešenia. Táto kapitola neuvádza len jeden konkrétny návrh riešenia, ale najmä nápady a koncepty, ktorých zlúčením a aplikovaním môže vzniknúť úspešný projekt. Skúsenosti nám ukázali, že úspešný nápad takmer zákonite nevznikne na začiatku riešenia projektu, ale prichádza počas fázy prototypovania a projekt dotvára svoju tvár počas všetkých etáp životného cyklu a preto aj návrh, uvedený na konci tejto kapitoly, je potrebné považovať za predbežný a pripravený na akúkoľvek zmenu.

Podkapitola 3.1 uvádza analýzu aspektov, ktoré zabezpečili úspech predchádzajúcich projektov v tejto súťaži a podkapitola 3.2 súhrn konceptov a tiež konkrétnych nápadov, ktoré plánujeme aplikovať pri riešení projektu. V poslednej podkapitole uvádzame predbežný návrh realizácie projektu spolu s nápadmi na jeho rozšírenie.

1.4 Úspech v súťaži Imagine Cup

Podkapitola obsahuje súhrn aspektov podieľajúcich sa na úspechu v súťaži Imagine Cup, vychádzajúci z analýzy úspešných projektov z národných a medzinárodných kôl súťaže, ktorých sa traja z členov tímu v posledných rokoch zúčastnili. Okrem podmienky inovatívnosti, ktorú obvykle spĺňa väčšina projektov, sa ukazuje, že isté druhy projektov majú v súťaži väčší úspech ako iné.

Deti

Orientovať projekt tak, že sa dá na ňom ukázať, že je jeho používanie tak jednoduché, že to zvládnu aj malé deti zvyšuje veľmi v očiach porotcov kvalitu. Toto sa ukázalo na východoeurópskom kole v minulom roku, kedy vyhral tím zo Slovinska, ktorý mal síce jednoduchú (a už existujúcu) myšlienku, spočívajúcu na jednoduchom princípe rozpoznávania obrazu, ale dokázal ju predviesť vo forme hier pre deti, čo sa stretlo s nadšeným ohlasom nielen u publika, ale aj u porotcov. Navyše vzhľadom na tému vzdelávanie ja orientácia smerom k deťom veľmi vhodná, keďže pre nie je vzdelanie najdôležitejšie.

Handicapovaní

Veľa úspešných projektov sa týka integrácii handicapovaných ľudí. V svetovom finále sa umiestnili projekty venujúce sa strojovému rozpoznávaniu znakovej reči pre

nepočujúcich a v roku 2006 zvíťazil vo svetovom finále projekt riešiaci navigáciu nevidiacich v interiéroch hlasom.

Nie čisto softvérové riešenie

Projekty, ktoré pozostávajú len zo softvérového riešenia, v súťaži často len málo zaujmú a takmer nemajú šancu na úspech. Dobre sa hodnotí využitie inovatívnych používateľských rozhraní (napr. rozoznávaním obrazu), PDA zariadení, RFID snímačov, GPS lokátorov a podobne.

Potenciál reálneho uplatnenia

Aj keď v podmienkach hodnotenia súťaže tento bod nie je zahrnutý, ukázalo sa, že porotcovia nehodnotili dobre projekt, o ktorého potenciáli reálne ho využiť ich súťažiaci nepresvedčili a napriek tomu, že projekt bol v mnohých aspektoch zaujímavý, nemal úspech.

1.4.1 Idey a koncepty

Táto podkapitola obsahuje popis konceptov a konkrétnych nápadov, ktorých spojením sa môže projekt stať veľmi atraktívnym a úspešným v súťaži.

Adaptívny prístup

Každý má rád, keď sa posadí za program, ktorý mu „rozumie“ nech je na akejkoľvek úrovni zručnosti. Ak má byť vzdelávanie efektívne, je dôležitá najmä jeho dlhodobosť. Adaptívny prístup svojím postupným prispôsobovaním dlhodobosť umožňuje, ale nezabezpečuje. Preto je potrebné motivovať študujúceho k jeho používaniu.

Okrem „globálnej adaptívnosti“ zabezpečujúcej, že produkt sa bude meniť a bude rásť s dieťaťom, je dôležité zohľadniť aj „lokálnu adaptívnosť“, ktorá zabezpečí aktuálne prispôsobenie sa nálade, momentálnym preferenciám alebo predchádzajúcemu progresu používateľa.

Motivácia – kto sa rád učí? Nikto. Kto sa rád hrá? Všetci.

Už od detstva sa radi hráme. Pomyslenie na štúdium v nás naopak vyvoláva pocit nutnosti, čím sa musí do zamýšľaného procesu zapájať vôľa. Hoci je vôľa silný psychologický fenomén, nie je schopná zabezpečiť vytrvalosť. Efektívne sa učiť znamená nadobúdať znalosti, a mať pri tom pocit dobre vykonanej práce a chuť pokračovať ďalej. Motiváciu je možné jednoducho docieliť prostredníctvom hry. Hra nesmie byť jednotvárna, a musí umožňovať rozširovanie.

Ak sa deťom asociuje od raného veku učenie so zábavou, prechod medzi hrou a učením neskôr nemusia ani postrehnúť a budú sa učiť rady.

Konektivita a kolaborácia

Ak má byť učenie efektívne, nestačí len, aby program obsahoval aktuálnu databázu informácií. Študujúci človek musí za väčšinou aktivít vidieť ľudí. Učenie je efektívne vtedy, keď pri ňom človek nie je sám. Projekt preto umožní prepojenie ľudí aj dát za účelom spoločného hrania a vzdelávania sa.

Dostupnosť

Sústavné vzdelávanie je v dnešnej dobe pre mnoho ľudí nedostupnou komoditou. Trvalosť vzdelávania je možné dosiahnuť jedine vtedy, keď je neustále dostupný jeho

zdroj. Dnešní študujúci sa nesmú a nechcú spoliehať len na jednu formu prístupu ku vzdelaniu. Vzdelanie musí byť prístupné všade a vždy podľa potreby.

1.4.2 Uvažované inovatívne technológie

Okrem nápadov je potrebné pre úspech projektu zabezpečiť aj v niečom málo tradičné alebo úplne nové prístupy vo forme použitých technológií. My sme doteraz uvažovali o nasledovných:

Rozhrania cez dotykové obrazovky (TabletPCs)

Takto realizované rozhranie umožní jednoduché ovládanie pre širokú škálu používateľov a prináša nové možnosti spôsobov interakcie.

Lego Mindstorm

Ide o produkt spoločnosti Microsoft, ktorý umožňuje programovať robotov postavených s využitím špeciálnej sady Lego v prostredí .NET. V projekte by sme ho mohli použiť na vytvorenie jednoducho použiteľného a atraktívneho rozhrania pre deti.

1.5 Hrubý návrh

Táto kapitola obsahuje popis približného návrhu podoby realizácie projektu, ktorý nie je však v ničom záväzný a predpokladáme jeho zmenu.

1-1 Motivačný záber

Predpokladá sa realizovanie systému, ktorý by predstavoval framework na zabezpečenie silne adaptívneho procesu výučby a jednej konkrétnej realizácie aplikácie postavenej na tomto frameworku. Takýto koncept sa ukázal v minulom roku súťaže ako veľmi úspešný.

Aplikácia postavená na navrhovanom frameworku by predstavovala komplexný systém meniaci sa tak ako dieťa rastie, ktorý by sa prispôsoboval tempu učenia sa dieťaťa počas jeho života. Začínalo by to perifériami vo forme herných kociek, formičiek, skladačiek, postavených zo sady Lego Mindstorm a využívajúcich senzory a akčné členy na zabezpečenie interaktivity pri hre, navrhnutými špeciálne tak, aby ich mohli používať veľmi malé deti, učiace sa rozoznávať farby a tvary. Neskôr by pribudlo rozhranie dotykovej obrazovky a deti by využívali systém na kreslenie, kedy by mohli obkresľovať tvary na obrazovke, učiť sa zároveň rozoznávať zvuky, systém by im vedel pustiť rozprávku...(dieťa obkreslí a vyfarbí kravičku, naučí sa ako robí kravička a môže dostať hneď rozprávku o kravičke).

V tomto momente prichádza do úvahy možnosť prispôsobovania sa, kedy systém zistí, čo sa dieťaťu páči viac a toto mu v budúcnosti ponúka častejšie. Napríklad by malo zo začiatku dieťa na výber obkresliť alebo vyfarbiť zviera, auto, dom alebo geometrické tvary, a podľa toho, ako by sa rozhodovalo by systém odhadoval jeho schopnosti a záujmy, na základe ktorých by sa mu prispôsoboval.

V školskom veku by sa zo systému stal akýsi zápisník, synchronizovaný so serverom, kde by malo dieťa uložené svoje poznámky a nemuselo by si ich nosiť medzi domovom a školou. Prekážkou môže byť relatívna nepohodlnosť pri písaní dotykovým perom na tablet, čo by sa dalo vyriešiť zapojením webcam, ktorá by snímala to čo dieťa nakreslí/napíše na obyčajný papier, a tieto obrázky po spracovaní (spracovanie by mohlo určovať ako dobre sa dieťa učí jednotlivé písmená a nechávalo by to dieťa precvičovať písmená ktoré mu nejdú viac ako tie ktoré zvláda (v druhej fáze napríklad využiť OCR na prevod poznámok do textu)) ukladá na server. Dáta na serveri by boli synchronizované a doma by bol prístup k tomu istému obsahu ako v škole.

Ponúka sa možnosť pridať do projektu niečo na spôsob e-learningu, kedy by bola časť školskej výučby zabezpečená s pomocou tohto systému a niektoré predmety by sa žiaci učili výlučne elektronicky. V tejto fáze by bolo možné využiť výsledky výskumu na fakulte a zakomponovať do projektu niečo, čo ukáže reálne uplatnenie nejakej metódy/prístupu vyvinutej v rámci výskumu.

Ďalšie nápady a vlastnosti produktu:

- na jednom počítači viacero používateľov aplikácie (profily),
- porovnanie úspechov jednotlivých študentov,
- virtuálne triedy,
- rozhrania na integrovanie modulov na skúšanie, na integráciu systému na informačný systém v ktorom sú hodnotenia, na systém prihlášok na ďalšiu školu...
- integrácia viacerých netradičných hier – napríklad dieťa si vystrihne z papiera objekty v hre (tieto buď „vloží“ do tabletu obkreslením dotykovým perom

alebo vymyslíme spôsob ako rozpoznávať dotyk takéhoto predmetu na obrazovke) a hrá sa cez internet s iným dieťaťom tak, že nimi pohybuje po obrazovke tabletu

- karaoke
- mnohé ďalšie adaptívne hry pri ktorých sa deti učia

1.6 Predpokladané zdroje

Vzhľadom na podmienku účasti v súťaži Imagine Cup, ktorá predpisuje implementovať riešenie v jednom z produktov rady Microsoft Visual Studio 2005 a jazyku C#, VB.NET alebo C++ predpokladáme využitie prostredia Microsoft Visual Studio 2005 Prefessional. Ďalšou z podmienok súťaže je použiť v projekte ASP.NET webovú službu, čo implikuje potrebu webového servera Microsoft IIS. Tieto zdroje sme schopní zabezpečiť si aj svojpomocne na súkromných počítačoch, tak ako aj ostatné hardvérové a softvérové zdroje.

1.7 Záver

V súťaži ako je Imagine Cup je okrem vedomostí a skúseností potrebný najmä kreatívny nápad a plné nasadenie všetkých členov tímu.

Vedomosti majú mnohí, skúsenosti už menej ľudí, kreatívny nápad málokto, ale nájsť skupinu ľudí, ktorá okrem toho, že má faktické predpoklady na úspech, má aj dostatočný zápal na to, aby projekt dotiahla s využitím maximálneho času a úsilia do úspešného konca môže byť ťažké.

My sme taký tím a sme pripravení a odhodlaní dosiahnuť v tejto súťaži veľa. Ďakujeme za Váš čas a pozornosť pri čítaní tohto dokumentu.

1.8 Príloha A – Preferencie tímu

Vzhľadom na fakt, že ide o požadovanú časť ponuky, uvádzame tu preferencie členov tímu ohľadom výberu ostatných ponúkaných projektov, aj keď v prípade potreby výberu inej témy by tím pracoval v inom zložení a predpokladáme, že v riadnom kole výberu sa bude vypracovávať ešte jedna ponuka.

Preferujeme témy v nasledovnom poradí:

- Softvérový návrh v rámci súťaže Imagine Cup (Imagine Cup)
- Kandidát na najlepší multimediálny produkt roku 2007 (EuroPrix)
- Počítačová hra - plánovanie a simulácia horolezeckej expedície (HOREX) s p. Ing. Štefanovičom
- Znalostný manažment na báze technológie .NET (Knowledge Office)
- Tvorba obaľovačov na získavanie informácií z webu (WRAPPER)
- Tvorba testov s využitím LaTeXu (TESTY)
- Softvérová podpora životného cyklu študentského projektu (PROJEKTY)
- RoboCup – nové stratégie (RoboCup S)
- Robocup – tretí rozmer (RoboCup 3D)
- Počítačová hra - plánovanie a simulácia horolezeckej expedície (HOREX) s p. Doc. Šperkom

1.9 Príloha B – Rozvrh členov tímu

Tabuľka 1-1Rozvrh členov tímu

	7:20 8:10	8:15 9:05	9:15 10:05	10:10 11:00	11:10 12:00	12:05 12:55	13:05 13:55	14:00 14:50	15:00 15:50	15:55 16:45	16:55 17:45	17:50 18:40	18:50 19:45
Pon	APS všetci			Mimoškolské aktivity OV			Stretnutie k TP 2			TSST1 všetci	@TSST1 všetci		
Uto													
Str	NS RV, OV				Stretnutie k TP 4			ZK všetci		Stretnutie k TP 1			
Stv		NP všetci		ASS všetci					MSI všetci		@MSI všetci		
Pia		@ZK všetci		@NP všetci			Stretnutie k TP 3						

Legenda:

AF Andrej Frlička

MT Marek Tomša

RV Richard Veselý

OV Oto Vozár

Preferujeme stretnutia v uvedenom poradí (najprv č. 1, potom 2, 3 a 4).

2 Plán projektu

Kapitola obsahuje genézu plánov vývoja projektu ku jednotlivým kontrolným bodom.

2.1 Plán projektu ku 1. kontrolnému bodu (19.10.2006)

Podkapitola obsahuje plán projektu ku 1. kontrolnému bodu. Ide o hrubý plán na zimný semester.

Tabuľka 2-1Plán projektu ku 1. kontrolnému bodu

ID		Opis	začiatok	predpokladaný koniec	dátum ukončenia
úloha	Činnosť				
1		Špecifikácia a analýza			
	1	Zber nápadov pre prvý prototyp	10/1/2006	10/30/2006	10/30/2006
	2	Špecifikácia rámca	10/23/2006	10/30/2006	11/16/2006
	3	Špecifikácia obsahu	10/18/2006	11/6/2006	11/6/2006
	4	Zber nápadov pre rozšírenia	10/1/2006	12/5/2006	12/5/2006
2		Návrh a implementácia prvého prototypu			
	1	Implementácia prototypu rámca	10/30/2006	11/3/2006	prebieha
	2	Implementácia obsahu	11/2/2006	11/13/2006	prebieha
	3	Príprava a odovzdanie dokumentácie analýzy špecifikácie a návrhu	10/23/2006	11/16/2006	11/16/2006
3		Rozširovanie prototypu			
	1	Špecifikácia rozšírení	11/7/2006	11/14/2006	12/11/2006
	2	Implementácia a testovanie rozšírení prototypu	11/15/2006	12/10/2006	prebieha
	3	Integrácia a odovzdanie prototypu	12/11/2006	12/18/2006	prebieha
4		Dokumentovanie			
	1	Príprava a odovzdanie projektovej dokumentácie	10/23/2006	12/18/2006	prebieha
	2	Prezentácia prototypu	12/21/2006	12/21/2006	plánované

2.2 Plán projektu ku 2. kontrolnému bodu (19.12.2006)

Podkapitola obsahuje plán projektu ku 2. kontrolnému bodu. Ide o hrubý plán na letný semester.

Tabuľka 2-2 Plán projektu ku 1. kontrolnému bodu

ID		Opis	začiatok	predpokladaný koniec	dátum ukončenia
úloha	činnosť				
5		Klientská časť			
	1	Implementácia enginu	2/12/2007	2/28/2007	
	2	Implementácia klienta	2/19/2007	3/14/2007	
	3	Návrh a implementácia hlasového rozhrania	2/12/2007	2/26/2007	
	4	Integrácia prvej iterácie sady grafiky	3/1/2007	3/8/2007	
	5	Návrh peta	2/12/2007	2/19/2007	
	6	Návrh a implementácia rozhrania na detekciu písma	2/12/2007	3/14/2007	
	7	Implementácia peta v gadgete	2/19/2007	3/5/2007	
	8	Návrh a implementácia synchronizácie peta s PDA	3/7/2007	3/12/2007	
6		Serverová časť			
	1	Implementácia vybraných questov	2/20/2007	3/20/2007	
	2	Návrh a implementácia mapy	2/12/2007	2/20/2007	
	3	Návrh služieb personalizácie	3/1/2007	3/8/2007	
	4	Implementácia služieb personalizácie	3/5/2007	3/19/2007	
	5	Integrácia s klientom a testovanie	3/14/2007	4/1/2007	
7		Mobilná časť			
	1	Implementácia peta v mobilnej časti	2/20/2007	2/28/2007	
	2	Implementácia mechanizmu synchronizácie	3/1/2007	3/7/2007	
8		Práca na dokumentácii			
	1	Vypracovanie posudku prototypu druhého tímu	12/20/2006	12/22/2006	
	2	Technická dokumentácia	2/12/2006	4/14/2007	
	3	Dokument riadenia	2/12/2007	4/14/2007	
	4	Finalizácia dokumentácie	4/14/2007	termín odovzdania	
9		Integrácia			
	1	Integrácia	20.2.2007	30.4.2007	

	2	Finálna integrácia a testovanie	14.4.2007	28.4.2007	
10		Prezentácia v národnom kole súťaže			
	1	Príprava špecifikácie odovzdanej v súťaži	3/14/2007	3/20/2007	
	2	Príprava prezentácie	apríl 2007	máj 2007	
	3	Účasť v súťaži	máj 2007	august 2007	

3 Úlohy členov tímu

Kapitola definuje rozdelenie úloh členov tímu ku jednotlivým kontrolným bodom. Okrem rozdelenia rolí, definuje aj podiel členov na tvorbe projektovej dokumentácie.

3.1 Roly členov tímu ku 1. kontrolnému bodu (19.10.2006)

Podkapitola popisuje roly všetkých členov v tíme ku dňu 19.10.2006.

3.1.1 Roly manažmentu projektu

Nasledujúci zoznam popisuje všetky roly súvisiace s manažmentom projektu. Osoby uvedené sú zodpovedné za riadenie príslušnej oblasti tvorby softvéru.

Tabuľka 3-1Roly manažmentu projektu v 1. kontrolnom bode

Rola	Meno
Vedúci tímu	Marek Tomša
Manažment integrácie projektu	Marek Tomša
Manažment rozvrhu projektu	Marek Tomša
Manažment rizík	Andrej Frlička
Manažment akosti	Oto Vozár
Manažment ľudských zdrojov	Andrej Frlička
Manažment komunikácie	Andrej Frlička
Manažment rozsahu projektu	Rišo Veselý
Manažment vývoja	Rišo Veselý

3.1.2 Roly tvorby projektu

Tabuľka rozdeľuje činnosti tvorby tímového projektu. Osoby, ktoré sú v zozname uvedené majú hlavné slovo pri vykonávaní zodpovedajúcej činnosti. Ostatní sa podieľajú aj na činnostiach pri ktorých nie sú pridelení podľa potreby.

Tabuľka 3-2Roly tvorby projektu v 1. kontrolnom bode

Rola	Mená
Programovanie	Rišo Veselý, Oto Vozár
Dokumentácia	Andrej Frlička, Marek Tomša
Dizajn a grafika	Oto Vozár, Andrej Frlička
Prezentácia	Richard Veselý

3.2 Roly členov tímu ku 2. kontrolnému bodu (19.12.2006)

Podkapitola popisuje roly všetkých členov v tíme ku dňu 19.10.2006.

3.2.1 Roly manažmentu projektu

V rolách manažmentu projektu sa oproti 1. kontrolnému bodu stav nezmenil. (viď 3.1.1)

3.2.2 Roly tvorby projektu

V rolách tvorby projektu sa oproti 1. kontrolnému bodu stav nezmenil (viď 3.1.2)

4 Záznamy zo stretnutí

4.1 Zápisnica k 1. Stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
1	11.10.2006	Soft. Štúdio	18.10.2006
Pedagóg	prof. Ing. Mária Bieliková (MB)	Vypracoval	Andrej Frlička
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.1.1 Priebeh stretnutia

Privítanie a zoznámenie

Zoznámenie sa, zahájenie činnosti na projekte, rozdelenie úloh a povinností, začiatok analýzy

Pokyny pre prácu v rámci stretnutia tímového projektu

- Priložiť dôraz na manažment rizík
- Robiť manažment rizík
- Nastaviť si očakávania od ľudí
- Písanie dokumentácie
- Štýl dokumentácie – technická
- Neoddeľuje sa text od diagramov (Nie do samostatných príloh)
- Vytvoriť blok na synchronizáciu údajov pre tvorbu technickej dokumentácie
- Časť o riadení projektu
- Časť o projekte
- Priebežne dokumentovať aj neúspešné pokusy, resp. Alternatívy
- Vytvoriť si projektový denník
- Forma: individuálna
- Min 1x za týždeň používať.
- Každý, kto píše zápis, by ho mal okamžite uverejniť na webe.
- Využitie nástrojov na podporu manažmentu
- Len evidencia úloh v nástroji
- Poznámky a zápisy zo stretnutia – voliteľné.

- Všetky drobné úlohy do konca septembra doimplementovať.

Rozdelenie úloh v tíme

Začiatok analýzy problémovej oblasti

- Integrovať systém s ostatnými systémami
- Prispôsobovať veku dieťaťa
- Využiť výstupy z iných vzdelávacích systémov
- Zamyslieť sa nad rolou učiteľa , rodiča, staršieho kamaráta, starého rodiča
- Virtuálna trieda

4.1.2 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.1	Vytvoriť web prezentáciu projektu	11.10.2006	20.10.2006	AF	Začaté
1.2	Rozdeliť úlohy v tíme	11.10.2006	18.10.2006	MT	Začaté
1.3	Vymedziť alternatívy riešenia	11.10.2006	20.10.2006	Všetci	Začaté

4.2 Zápisnica k 2. Stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
2	16.10.2006	Soft. Štúdio	18:00
Pedagóg	Neprítomný	Vypracoval	Andrej Frlička
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.2.1 Téma stretnutia

Motion capturing

Rozpoznávanie hovoreného slova

Analýza inovatívnych rozhraní

Pokus o integráciu nápadov

Nápady

- Výučba hudby
- Dospelý pripraví pre dieťa herné prostredie
- Dieťa sa postaví, môže hovoriť, ukazovať
- 2D plocha
- 2 rôzne vizuálne značky pre rodiča a pre dieťa
- Motion capture
- Tímy a družstvá
- Písanie, spelling, čítanie
- Rozvoj schopností interaktívnymi hrami.
- Ukazovanie písmen, tvarov
- Dve značky
- Hľadanie značky v miestnosti
- Stereoskopické okuliare

4.2.2 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.1	Vytvoriť web prezentáciu projektu	11.10.2006	20.10.2006	AF	Prebieha
1.2	Rozdeliť úlohy v tíme	11.10.2006	18.10.2006	MT	Prebieha
1.3	Vymedziť alternatívy riešenia	11.10.2006	20.10.2006	Všetci	Prebieha

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.4	Naštudovať vhodnú metodológiu a zvoliť vhodný nástroj na riadenie projektu	16.10.2006	24.10.2006	Všetci	Začaté
1.5	Distribuovať programovacie smernice.	16.10.2006	24.10.2006	RV	Začaté
1.6	Zistiť vlastnosti technológie rozpoznávania zvuku	16.10.2006	24.10.2006	OV,MT	Začaté
1.7	Analyzovať možnosti alternatívnych možností ovládania	16.10.2006	24.10.2006	MT	Začaté
1.8	Zistiť informácie o motion capturing	16.10.2006	24.10.2006	AF,OV	Začaté

4.3 Zápisnica k 3. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
3	18.10.2006	Soft. Štúdio	8:15-10:00
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Marek Tomša
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.3.1 Priebeh stretnutia

- Diskusia o možnostiach alternatívneho rozhrania
  - zamietnutie rozoznávania ruky ako kresliaceho nástroja – prijatá alternatíva kreslenia na TabletPC ako vhodnejšia
  - prijatie alternatívy speech rozhrania
- Vyhodnotené že trh s výukovým sw/hrami pre pre deti vo veku 6+ je slabý
  - Barney z roku 1995
  - Magic Schoool bus z roku 1995
  - Takmer žiadna zmienka na internete
  - dostupné hry úzko špecializované
  - Kolaboratívne hry pre deti nie sú žiadne
  - Second Life od 13 rokov vyššie
- Diskusia o vlastnostiach produktu
  - aktivity s rodičom prijaté ako veľmi výhodná vlastnosť
  - skilly a vytváranie tímov
  - Odporúčanie zloženia tímov tak aby boli rôznorodé
  - Zásah rodiča (učiteľa)
  - kolaborácia – deti sa učia navzájom, motivované spoločným úspechom tímu
- Možnosti rozšírenia herného sveta
  - Editor rozšírení – nateraz úloha s nízkou prioritou
  - Návrh prepojiť jednotlivé druhy aktivít tak aby sa dieťa učilo to čo mu nejde
  - Matematika + hudba
  - Vyklepávanie rytmu = teraz ťukneš dvakrát, naučí sa zlomky ako takty a rytmy (?)
  - určovať kombinácie na základe nálady – určená na začiatku dieťaťom alebo rodičom/učiteľom

- K tomu možnosť riadiť výber aktivít rodičom/učiteľom priamo – pri výučbe jednotlivých predmetov

- Diskusia o zdrojoch obsahu
  - Prevziať z nejakého výučbového systému
  - z detskej knihy – pridať sprievodcu ako postavičku z knihy
  - prevziať metodiku z učebníc pre deti
  - spelling cez Speech API

4.3.2 Závery stretnutia

- revidovaný plán stretnutia bude ako príloha zápisov zo stretnutí
- zápisy budeme vkladať ako PDF, nie do šablón v joomle
- budeme mať v CMS wiki do ktorej budeme vpisovať nápady
- prijímame myšlienku realizovať projekt ako rámec virtuálneho sveta pre deti s výukou vo forme hier
- pokračujeme v zbere nápadov pre obsah

4.3.3 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.4	Naštudovať vhodnú metodológiu a zvoliť vhodný nástroj na riadenie projektu	16.10.2006	24.10.2006	Všetci	Dokončené
1.5	Distribuovať programovacie smernice.	16.10.2006	24.10.2006	RV	Prebieha
1.6	Zistiť vlastnosti technológie rozpoznávania zvuku	16.10.2006	24.10.2006	OV,MT	Dokončené
1.7	Analyzovať možnosti alternatívnych možností ovládania	16.10.2006	24.10.2006	MT	Dokončené
1.8	Zistiť informácie o motion capturing	16.10.2006	24.10.2006	AF,OV	Dokončené

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.9	Vytvoriť plán projektu	18.10.2006	23.10.2006	MT	Plánované
1.10	Zapracovať požadované úpravy stránky	18.10.2006	23.10.2006	AF	Začaté
1.11	Analyzovať Speech API v .NET 3.0 vs. MS Speech Server a odporučiť jednu z technológií	19.10.2006	23.10.2006	RV	Plánované
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Začaté

1.13	Zistiť aké predmety sa učia v prvom ročníku ZŠ a analyzovať možnosti implementácie	19.10.2006	23.10.2006	AF	Plánované
1.14	Spraviť prieskum medzi deťmi aké hry sa hrajú a čo by sa im páčilo	21.10.2006	23.10.2006	AF	Plánované
1.15	Zistiť ako sa počíta násobilka na prstoch	18.10.2006	23.10.2006	MB	Plánované

4.4 Zápisnica k 4. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
4	23.10.2006	Soft. Štúdio	12:30-13:30
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Richard Veselý
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.4.1 Priebeh stretnutia

- Návrh pokrytia problémovej domény do hĺbky, nie do šírky
- Návrh vytvoriť 4 až 5 hier, ktoré budú koncepčne zvládnuté a funkčné (demonštrovateľné)
    - Hľadať námety
    - Hľadať možnosti interakcie hier s deťmi (motion, speech)
- Návrh automatizovať systém
    - Autonómnosť rozhodnutia, aké aktivity by malo dieťa vykonávať
    - Schopnosť vystupovať v roli učiteľa, v prípade, že reálny absentuje alebo nezasahuje do pedagogického procesu
    - Rodič by mal mať schopnosť do výučby zasahovať a konfigurovať niektoré aspekty pedagogického procesu
- Zaznamenávať odpovede dieťaťa
    - Možnosť vyhotovenia prospechovej štatistiky
    - Analýza problémových oblastí
- Návrh zaviesť do systému funkcionalitu žiackej knižky
    - Zvážiť spracovanie externých dát (hodnotení prospechu zo školy)
    - Evidencia virtuálnych výsledkov
- Personalizácia – „milión detí, milión hier"
- Návrh kombinácie vnemov v hrách – napr. prezentácia zvuku alebo obrázku zvieraťa, pričom dieťa napíše prvé písmeno prezentovaného objektu
- Analýza domény – matematika
    - Personalizácia príkladov (v závislosti od toho, kto stojí pri tabuli)
    - Kategórie príkladov (násobilka, reťazovky, ...)
    - Adaptívna obtiažnosť
        - Zohľadňuje úspešnosť odpovedí a na jej základe upravuje zložitosť príkladov

        - V prípade nízkej úspešnosti prechod do režimu učenia sa
    - Opakovať príklady, ktoré boli v minulosti neúspešne zodpovedané
- Rozdelenie virtuálneho sveta na bezpečné a nebezpečné zóny
    - Bezpečné zóny – vstup do sveta, dedinky, mestá, arény
    - Nebezpečné zóny – okolie osídlených oblastí
        - Konfrontácia (napr. s príšerami po vzore MMORPG) a skúšanie z nadobudnutých vedomostí
- Personalizácia s ohľadom na pohlavný dimorfizmus
- Použitie tútorov – zvieratok/postavičiek s vysokým „cute“ faktorom (Mokona Modoki)
- Skill balancing – výber vhodnej množiny skúšaných detí v prípade skupinovej hry, aby neboli vedomosti jednotlivých členov skupiny príliš nevyvážené

4.4.2 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.9	Vytvoriť plán projektu	18.10.2006	23.10.2006	MT	Dokončené
1.10	Zapracovať požadované úpravy stránky	18.10.2006	23.10.2006	AF	Dokončené
1.11	Analyzovať Speech API v .NET 3.0 vs. MS Speech Server a odporučiť jednu z technológií	19.10.2006	23.10.2006	RV	Dokončené
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Prebieha
1.13	Zistiť aké predmety sa učia v prvom ročníku ZŠ a analyzovať možnosti implementácie	19.10.2006	23.10.2006	AF	Dokončené
1.14	Spraviť prieskum medzi deťmi aké hry sa hrajú a čo by sa im páčilo	21.10.2006	23.10.2006	AF	Prebieha
1.15	Zistiť ako sa počíta násobilka na prstoch	18.10.2006	23.10.2006	MB	Prebieha

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.16	Analyzovať ScrumWorks – podporný prostriedok na manažovanie softvérového projektu	23.10.2006	28.10.2006	RV	Plánované
1.17	Analýza hudobnej	23.10.2006	28.10.2006	OV	Plánované

	pedagogickej domény z hľadiska možnosti implementácie				
1.18	Špecifikovať rámec	23.10.2006	28.10.2006	Všetci	Plánované
1.19	Vytvoriť počiatočnú verziu dokumentácie – šablóna, štruktúra (riadenie a výsledok)	23.10.2006	28.10.2006	AF	Plánované
1.20	Pridať do plánu Ganttov diagram	23.10.2006	28.10.2006	MT	Plánované

4.5 Zápisnica k 6. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
5	6.11.2006	Soft. Štúdio	13:30-15:00
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Andrej Frlička
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.5.1 Priebeh stretnutia

- Diskusia ku dokumentácii
    - V časti riadenia je potrebné pridať obsah
    - Každá časť v sekcii riadenia je zložená z hlavičky a dokumentov, ktoré sa prikladajú v nezmenenej podobe
    - Sekcia riadenia musí obsahovať obsah.
    - Úvodný text ku každej kapitole v sekcii riadenia by mal zahŕňať popis kapitoly a dátumy pridania jednotlivých jej častí (napríklad ponuka).
    - Preberací protokol, ako aj posudky nie sú povinnou súčasťou dokumentácie ku prvému kontrolnému bodu

- Diskusia o štandardoch
    - Sekcia štandardy obsahuje najmä pravidlá písania dokumentácie, písania zdrojového kódu. Môže zahŕňať aj šablónu, podľa ktorej sa píšu dokumenty
    - Do dokumentu je možné pridať upravené programovacie štandardy.
    - Neskôr bude možné pridať aj štandardy práce s databázou

- Diskusia o vytvorených prototypoch
    - Vhodné implementovať metronóm pre prácu so zložitejšími rytmami
    - Vhodné naštudovať štandard midi (Musical Instrument Digital Interface). Ako abstrakciu nad nástrojmi a ako reprezentáciu hudobných
- Diskusia o hrách
    - Dieťa by mohlo ukazovať časti tela na svojom vlastnom obraze
    - Mohli by sa určovať mláďatá zvierat, pretože s tým majú deti obvykle problémy

- Hra čiarový labyrint by sa mohla uplatniť ako matematická hra (navigácia v priestore)
- Kolaboratívne kreslenie
- Zaujímavé by bolo zaradiť hru, ktorá by pomohla deťom uvedomiť si, že slová sú zložené z písmen – Deti s tým majú obvykle problémy
- SCRABBLE, HANGMAN – predstavitelia slovných hier

4.5.2 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Prebieha
1.14	Spraviť prieskum medzi deťmi aké hry sa hrajú a čo by sa im páčilo	21.10.2006	23.10.2006	AF	Zrušené
1.16	Analyzovať ScrumWorks – podporný prostriedok na manažovanie softvérového projektu	23.10.2006	28.10.2006	RV	Prebieha
1.20	Pridať do plánu Ganttov diagram	23.10.2006	28.10.2006	MT	Dokončené
1.21	Príprava dokumentácie (analýza problému, špecifikácia požiadaviek a návrh riešenia)	30.10.2006	16.11.2006	Všetci	Prebieha
1.22	Vytvorenie prvého prototypu prostredia	30.10.2006	6.11.2006	Všetci	Dokončené

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.23	Pridať obsah do časti riadenia	6.11.2006	13.11.2006	AF	Plánované
1.24	Vytvoriť šablónu písania dokumentácie (Definovať štýly a ich použitie)	6.11.2006	13.11.2006	OV	Plánované
1.25	Vytvoriť preberací protokol	6.11.2006	13.11.2006	AF	Plánované
1.26	Vytvoriť systém číslovania obrázkov	6.11.2006	13.11.2006	AF	Plánované
1.27	Vymyslieť spôsob zápisu rytmu	6.11.2006	13.11.2006	OV	Plánované
1.28	Zistiť, či máme k dispozícii x-box	6.11.2006	13.11.2006	OV	Plánované
1.29	Informácie o XNA z konferencie	6.11.2006	13.11.2006	OV	Plánované
1.30	Pozrieť frekvenčnú analýzu	6.11.2006	13.11.2006	Všetci	Plánované
1.31	Navrhnúť sieťovú	6.11.2006	13.11.2006	Všetci	Plánované

	technológiu				
1.32	Implementovať metronóm	6.11.2006	13.11.2006	RV	Plánované

4.6 Zápisnica k 6. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
6	6.11.2006	Soft. Štúdio	13:30-15:00
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Andrej Frlička
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.6.1 Priebeh stretnutia

- Diskusia ku dokumentácii
    - V časti riadenia je potrebné pridať obsah
    - Každá časť v sekcii riadenia je zložená z hlavičky a dokumentov, ktoré sa prikladajú v nezmenenej podobe
    - Sekcia riadenia musí obsahovať obsah.
    - Úvodný text ku každej kapitole v sekcii riadenia by mal zahŕňať popis kapitoly a dátumy pridania jednotlivých jej častí (napríklad ponuka).
    - Preberací protokol, ako aj posudky nie sú povinnou súčasťou dokumentácie ku prvému kontrolnému bodu
    - Nutné zvoliť správne číslovanie obrázkov pri ich vysokom počte

- Diskusia o štandardoch
    - Sekcia štandardy obsahuje najmä pravidlá písania dokumentácie, písania zdrojového kódu. Môže zahŕňať aj šablónu, podľa ktorej sa píšu dokumenty
    - Do dokumentu je možné pridať upravené programovacie štandardy.
    - Neskôr bude možné pridať aj štandardy práce s databázou

- Diskusia o vytvorených prototypoch
    - Vhodné implementovať metronóm pre prácu so zložitejšími rytmami
    - Vhodné naštudovať štandard MIDI (Musical Instrument Digital Interface). Ako abstrakciu nad nástrojmi a ako reprezentáciu hudobných udalostí.
    - Vizualizovať závislosť chybovosti vyklepkávania od času (Tak aby prípadná chyba neznedohnotila skóre vo vyklepkávaní príliš)
    - XNA sa javí ako vhodná platforma pre programovanie herného sveta

- Diskusia o hrách
    - Dieťa by mohlo ukazovať časti tela na svojom vlastnom obraze
    - Mohli by sa určovať mláďatá zvierat, pretože s tým majú deti obvykle problémy
    - Hra čiarový labyrint by sa mohla uplatniť ako matematická hra (navigácia v priestore)
    - Kolaboratívne kreslenie
    - Zaujímavé by bolo zaradiť hru, ktorá by pomohla deťom uvedomiť si, že slová sú zložené z písmen – Deti s tým majú obvykle problémy
    - SCRABBLE, HANGMAN – predstavitelia slovných hier

- Diskusia o hernom svete
    - Na rôzne hry by mohli existovať rôzne ihriska, resp. rôzne arény
    - Vytvoriť ligu pozostávajúcu z arén a trofejí
    - Mohli by existovať lokálne (oblastne zamerané) alebo globálne súťaže
    - Deti by si mohli hry zadávať vzájomne, čím by mohli meniť silu „útoku“ na súpera
    - Mohlo by existovať niekoľko rôznych typov súbojov s rôznymi pravidlami
    - Náročnosť súboju by závisela od úrovne znalostí (minimálna by sa zvolila podľa minimálnej skúsenosti najslabšieho člena
    - Postavy by mohli mať špeciálne vlastnosti v závislosti od svojej úrovne
    - Redukovať odčítavanie a znižovanie skóre , prípadne ho slušne zaobaliť.
    - Herná mena by mohla existovať vo viacerých minciach (prínos ku matematike)
    - Mágia, kúzla prípadne vedomosti by sa mohli využívať nielen v dueloch ale aj ako konštruktívny prvok (Liečenie, pomoc spoluhráčovi...), prípadne ako staviteľský prvok (čím by sa odbúrala nutnosť hernej meny a všetko by bolo priamo závislé na vedomostiach)
    - Vznikol problém, ako kvantifikovať rozdiely medzi jednotlivými oblasťami výuky, a ako kvantifikovať jednotlivé výsledky hier (ako ich reprezentovať v hre).

4.6.2 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Prebieha

1.14	Spraviť prieskum medzi deťmi aké hry sa hrajú a čo by sa im páčilo	21.10.2006	23.10.2006	AF	Zrušené
1.16	Analyzovať ScrumWorks – podporný prostriedok na manažovanie softvérového projektu	23.10.2006	28.10.2006	RV	Prebieha
1.20	Pridať do plánu Ganttov diagram	23.10.2006	28.10.2006	MT	Dokončené
1.21	Príprava dokumentácie (analýza problému, špecifikácia požiadaviek a návrh riešenia)	30.10.2006	16.11.2006	Všetci	Prebieha
1.22	Vytvorenie prvého prototypu prostredia	30.10.2006	6.11.2006	Všetci	Dokončené

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.23	Pridať obsah do časti riadenia	6.11.2006	13.11.2006	AF	Plánované
1.24	Vytvoriť šablónu písania dokumentácie (Definovať štýly a ich použitie)	6.11.2006	13.11.2006	OV	Plánované
1.25	Vytvoriť preberací protokol	6.11.2006	13.11.2006	AF	Plánované
1.26	Vytvoriť systém číslovania obrázkov	6.11.2006	13.11.2006	AF	Plánované
1.27	Vymyslieť spôsob zápisu rytmu	6.11.2006	13.11.2006	OV	Plánované
1.28	Zistiť, či máme k dispozícii x-box	6.11.2006	13.11.2006	OV	Plánované
1.29	Informácie o XNA z konferencie	6.11.2006	13.11.2006	OV	Plánované
1.30	Pozrieť frekvenčnú analýzu	6.11.2006	13.11.2006	Všetci	Plánované
1.31	Navrhnúť sieťovú technológiu	6.11.2006	13.11.2006	Všetci	Plánované
1.32	Implementovať metronóm	6.11.2006	13.11.2006	RV	Plánované

4.7 Zápisnica k 7. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
7	8.11.2006	D212	16:00-18:00
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Andrej Frlička
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	

4.7.1 Priebeh stretnutia

- Diskusia o komunikácii
    - Na komunikáciu medzi jednotlivými zložkami systému sa bude používať technológia WCF (Windows communication foundation) od firmy Microsoft. Technológia umožňuje obojsmernú riadenú komunikáciu s kontextom.
    - Problémom pri komunikácii môže byť viacero klientov, ktorý majú na server protichodné požiadavky. Situáciu je možné riešiť dvomi spôsobmi:
    - využitím stavových automatov
    - dekomponovaním servera a rozdelením zodpovednosti na jednotlivé jeho časti.
- Diskusia o prostriedkoch na manažment tímu
    - Po prekonzultovaní vlastností a požiadaviek na softvér riadenia projektu sme sa rozhodli využiť vlastnosti systému DotPreject, ktorý ponúka prehľadné webové rozhranie.
- Diskusia o architektúre komunikácie
    - Klient sa bude pripájať na webový server, ktorý mu poskytne nasledujúce možnosti.
    - Ponúkne mu informácie o aktuálnom stave výrezu sveta, v ktorom sa práve nachádza.
    - Pomôže mu vytvoriť, resp. sprostredkovať komunikáciu s jedným alebo viacerými ďalšími klientmi
    - Poskytne mu informácie o pripojení na jednotlivé dátové webové služby.
    - Zabezpečí zber štatistických informácií o používateľovi
    - Komunikácia bude prebiehať vo viacerých komunikačných kanáloch:
    - Textový kanál
    - Zvukový kanál
    - Audiovizuálny kanál

- V prípade potreby server len pomôže vytvoriť spojenie (audioviz. Kanál) a ďalej ho nesleduje.

- Diskusia o komunikačných vlastnostiach klienta, servera a webových služieb
  - Na serveri budú dáta neustále zavedené v pamäti, aby sa zvýšila rýchlosť a spoľahlivosť práce s nimi. Dáta sa budú v pravidelných intervaloch alebo počas plánovanej odstávky servera serializovať na pevný disk, aby sa zamedzilo prípadným stratám v profiloch používateľov.
  - Ukladanie dát na server bude v prvom priblížení blokujúce.
  - Herné webové služby budú sprístupnene bez potreby autentifikácie.
  - Klient eviduje informácie o predošlom stave herného okna, a aktualizuje ich podľa nového stavu. Ich diferenciou a prepočtami zabezpečuje plynulosť pohybu hráčov na mape aj v prípade zníženia dátovej šírky siete.

- Diskusia o mapovom serveri
  - Mapy budú uložené aj na klientovi aj na serveri. Klient si v prípade ich neaktuálnosti mapy od servera vyžiada.
  - Na mapovom serveri bude bežať niekoľko inštancií objektov nazývaných svety, a niekoľko inštancií nazývaných hráč(a samozrejme aj ďalšie objekty)
  - Svet je objekt, ktorý združuje informácie o hernej ploche po ktorej sa hráči pohybujú.
  - Hráč je objekt, ktorý združuje informácie o pozícii používateľa v ňom.
  - Bude existovať niekoľko svetov, ktoré budú navzájom prepojené. Svet je napríklad aj dungeon.... Medzi svetmi existujú spojnice. Server zabezpečiť prechod hráčov medzi svetmi.
  - Server eviduje, kde sa hrá nachádza a informuje tých hráčov, čo sú v jeho blízkosti o zmenách jeho stavu. Server zabezpečí v prípade zmeny kontextu hráča (napríklad uprostred hry) , že ho nebude informovať o dianí v hernom svete.

4.7.2 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Prebieha
1.16	Analyzovať ScrumWorks – podporný prostriedok na manažovanie softvérového projektu	23.10.2006	28.10.2006	RV	Dokončené
1.21	Príprava dokumentácie (analýza problému, špecifikácia požiadaviek a návrh riešenia)	30.10.2006	16.11.2006	Všetci	Prebieha
1.23	Pridať obsah do časti	6.11.2006	13.11.2006	AF	Prebieha

	riadenia				
1.24	Vytvoriť šablónu písania dokumentácie (Definovať štýly a ich použitie)	6.11.2006	13.11.2006	OV	Prebieha
1.25	Vytvoriť preberací protokol	6.11.2006	13.11.2006	AF	Prebieha
1.26	Vytvoriť systém číslovania obrázkov	6.11.2006	13.11.2006	AF	Prebieha
1.27	Vymyslieť spôsob zápisu rytmu	6.11.2006	13.11.2006	OV	Prebieha
1.28	Zistiť, či máme k dispozícii x-box	6.11.2006	13.11.2006	OV	Prebieha
1.29	Informácie o XNA z konferencie	6.11.2006	13.11.2006	OV	Prebieha
1.30	Pozrieť frekvenčnú analýzu	6.11.2006	13.11.2006	Všetci	Prebieha
1.31	Navrhnúť sieťovú technológiu	6.11.2006	13.11.2006	Všetci	Prebieha
1.32	Implementovať metronóm	6.11.2006	13.11.2006	RV	Prebieha

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.33	Nainštalovať DotPreject do webového sídla tímu	8.11.2006	13.11.2006	AF	Plánované

4.8 Zápisnica k 8. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
8	13.11.2006	Soft. Štúdio	13:10-15:00
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Marek Tomša
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.8.1 Priebeh stretnutia

- Diskusia o architektúre
  - model používateľa ako samostatná služba v skupine dátových služieb
- Grafický engine
  - diskusia o možnostiach a nutnosti interpolácie medzi jednotlivými políčkami pri pohybe
- Ekonomika v rámci hry
  - neobmedzené zásoby
  - hráči nepredávajú
  - vymieňajú peniaze za predmety (odmeny, vylepšenia) za fixné ceny
- Pet ako Vista gadget
  - Slúžiaci ako brána do sveta
  - premiestňujúci sa ako jedna inštancia medzi PDA a gadgetom/hrou
  - napojený na RSS feedy, špeciálne kanály v hre
  - vylepšenia zvieratka
    - Možnosť kúpiť mu hudobný nástroj a nacvičiť ho hrou v hudbe – bude hrať lepšie skladby
    - Kúpi mu teplomer a bude hovoriť a učiť dieťa rozoznávať počasie
- Krátka inštruktáž o Worde
  - prof. Bieliková nám vysvetlila prácu so sekciami a krížovými odkazmi vo Worde
- Dokumentovanie
  - prečíslovanie úloh – X.Y X = číslo zápisu, Y = číslo úlohy

4.8.2 Závery stretnutia

- Grafický engine bude interpolovať pohyb
- Prijímame architektúru orientovanú na služby
- Model používateľa bude samostatná služba
- Prijímame ideu zvieratka ako ústredného činiteľa v hre
  - v PDA, v gadgete, v hre
- Finalizujeme dokumentáciu a pripravujeme sa na jej odovzdanie

4.8.3 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Prebieha
1.21	Príprava dokumentácie (analýza problému, špecifikácia požiadaviek a návrh riešenia)	30.10.2006	16.11.2006	Všetci	Prebieha
1.23	Pridať obsah do časti riadenia	6.11.2006	13.11.2006	AF	Dokončené
1.24	Vytvoriť šablónu písania dokumentácie (Definovať štýly a ich použitie)	6.11.2006	13.11.2006	OV	Dokončené
1.25	Vytvoriť preberací protokol	6.11.2006	13.11.2006	AF	Dokončené
1.26	Vytvoriť systém číslovania obrázkov	6.11.2006	13.11.2006	AF	Dokončené
1.27	Vymyslieť spôsob zápisu rytmu	6.11.2006	13.11.2006	OV	Dokončené
1.28	Zistiť, či máme k dispozícii x-box	6.11.2006	13.11.2006	OV	Prebieha
1.29	Informácie o XNA z konferencie	6.11.2006	13.11.2006	OV	Dokončené
1.30	Pozrieť frekvenčnú analýzu	6.11.2006	13.11.2006	Všetci	Dokončené
1.31	Navrhnúť sieťovú technológiu	6.11.2006	13.11.2006	Všetci	Dokončené
1.32	Implementovať metronóm	6.11.2006	13.11.2006	RV	Prebieha

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
8.2	Dohodnúť odovzdanie dokumentu s konkurenčným tímom	13.11.2006	16.11.2006	MT	Plánované
8.3	Zrevidovať architektúru – pridať model používateľa	13.11.2006	20.11.2006	MT	Začalo

4.9 Zápisnica k 9. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
9	20.11.2006	Soft. Štúdio	13:00-15:00
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Richard Veselý
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.9.1 Priebeh stretnutia

- Súhrn aktivít zvieratka – pri súboji, všeobecne
    - Množina aktivít závislá od aktuálneho stavu hry
- Diskusia o type combat flow (turn-based, time limited turns, real-time)
- Grafická reprezentácia aktivít – reprezentácia hlavným grafickým enginom alebo custom
- Diskusia o diferencovaní služieb v prípade bojov (PC vs. NPC)
    - PvP, PvM
- Návrh štatistík
    - Možnosť použitia unifikovaného rozhrania alebo všeobecnej metódy Log s argumentum typu string
    - Určí sa po návrhu hier
- Zodpovednosť za odmeňovanie za výsledok hry – svet alebo v kompetencii hier
- Komunikačný kanál medzi službami (svet → aktivita, svet → hráči, hráč → aktivity)
    - Half-duplex
        - Synchronizácia súboja
        - Odmeny za výsledok hry
- Implementácia administratívnych rozhraní do služieb – konfigurácia na úrovni dát
- Režimy herných aktivít (tutorial, practice, normal)
- Live tutor – tutorial režim s asistenciou rodiča
- Multiclass challenge
    - Deti kolektívne riešia každý svoju úlohu a v závislosti od úspešnosti sa odvíja sila efektu kúzla jednotlivca alebo skupiny
    - Metrika úspešnosti
        - Matematika – presnosť výsledku
        - Hudba – presnosť určenia rytmu
- Definícia pravidlového vyhodnocovača – definujú sa na rôznych úrovniach pravidlá a tie sa následne (po ukončení aktivity) vyhodnotia
    - Participuje na výbere subaktivity
    - Subaktivity majú definované atribúty a po absolvovaní sady subaktivít sa tieto atribúty použijú ako vstupy pre vyhodnotenie pravidla
    - Analyzovať systém AHA

- Módy = "rýchlosti" (v newbie zóne – kým hráč nedosiahne určitú úroveň vedomosti)
    - „zrýchľovanie" – tutorial → practice → normal – učiteľ v meste
    - „spomaľovanie" – normal → practice → tutorial – newbie príšera
- Kategorizovanie domén – strom (sub-)aktivít
- Playground – pre najmenších

4.9.2 Závery stretnutia

- Definovanie cieľov projektu – presvedčiť, prečo je návrh jedinečný
    - Jedinečnosť – výrazná motivácia (viacero foriem hier, zvieratko, spôsob interakcie)
    - Základné črty
        - Skupinová interakcia
        - Personalizácia a adaptívnosť
        - Integrovanosť výukového procesu (hry + rodičia + učitelia)

4.9.3 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
8.2	Dohodnúť odovzdanie dokumentu s konkurenčným tímom	13.11.2006	16.11.2006	MT	Dokončené
8.3	Zrevidovať architektúru – pridať model používateľa	13.11.2006	20.11.2006	MT	Prebieha
1.32	Implementovať metronóm	6.11.2006	13.11.2006	RV	Zrušené
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Prebieha
1.28	Zistiť, či máme k dispozícii x-box	6.11.2006	13.11.2006	OV	Dokončené

4.10 Zápisnica k 10. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
10	27.11.2006	Soft. Štúdio	13:00-15:00
Pedagóg	prof. Ing. Mária Bieliková, PhD.	Vypracoval	Oto Vozár
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.10.1 Priebeh stretnutia

- Implementácia prototypu PDA klienta
    - AF predviedol prvú verziu prototypu
    - Diskusia k mapke na PDA klientovi – treba ju?
- Model zvieratka
    - Rozdelenie vlastností
        - Nálada
        - „Nálada“ viažuca sa na každý obor zvlášť
        - Úroveň vedomostí pre každý obor
    - Základné potreby
        - Základné potreby, ktorých neplnenie sa negatívne prejaví
        - Plnenie základných potrieb prostredníctvom rôznych hier
            - Napríklad nakŕmenie pomocou výberu správnych ingrediencii (ovocie, zelenina), ktoré zvieratko chce
    - Učenie zvieratka
        - 2 prístupy
            - Hráč hrá v PDA priamo za zvieratko
                - Priama kontrola nad činnosťou v minihrách
                - Za splnenie herných cieľov sú zvieratku zvyšované schopnosti
            - Hráč v hrách interaguje so zvieratkom tak, že ho učí
                - Autonómnejšie vystupovanie zvieratka
                - Napríklad učenie matematiky
                    - Zadá sa príklad
                    - Zvieratko naň odpovie zle
                    - Hráč povie zvieratku, ako to má byť správne
                    - Ak to bolo správne, zvieratku sa zvýšia schopnosti
                - Musí byť dohliadané na hráča, aby zvieratko učil správne (tretia entita)
    - Synchronizácia
        - Synchronizácia medzi serverom a PDA a naopak
        - Využitie Bluetooth
            - Ak sa hráč s PDA priblíži k počítaču s hrou, automaticky sa ho zvieratko opýta, či môže prejsť do gadgetu a odtiaľ prípadne do hry
            - Pri ukončení sa zasa uloží do PDA
        - Sankcionovanie hráčov, ktorí „stratili“ verziu zvieratka v PDA

4.10.2 Závery stretnutia

- Potreba overenia si niektorých myšlienok na prototype
- Zaujimavá možnosť použiť na synchronizáciu Bluetooth
- Nezameriavať sa do šírky ale začať implementovať

4.10.3 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav (%)
8.3	Zrevidovať architektúru – pridať model používateľa	13.11.2006	20.11.2006	MT	Prebieha
1.12	Vymyslieť hry	18.10.2006	23.10.2006	Všetci	Prebieha

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav (%)
10.1	Začať implementovať prototyp	27.11.2006	20.11.2006	Všetci	Prebieha

4.11 Zápisnica k 11. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
11	4.12.2006	D212	12:00-15:30
Pedagóg	-	Vypracoval	Marek Tomša
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.11.1 Priebeh stretnutia

4.11.1.1 Prototyp

- diskusia o rozsahu prototypu
- Analýza možností izometrického premietania
    - ¾ perspektíva – jednoduchá implementácia v existujúcom návrhu enginu pracujúcom so screen koordinátmi
  - Particle efekty sú v screen koordinátoch, čo zjednodušuje implementáciu oproti doteraz uvažovanému 3D izometrickému premietaní
  - Vyplývajúce požiadavky na grafické objekty – budovy a iné kubické objekty kreslené do hexagonálnej šablóny (pripraviť šablóny a metodiku pre kreslenie tilov), „podlaha“ kreslená do štvorcovej siete, transformované pri renderovaní
- Prijatie modelu „choď kam kliknem“ v prototype – vyplývajúca potreba algoritmov hľadania cesty
- Diskusia o postupe pri implementácii prototypu
    - zmena perspektívy
    - pridanie komunikačných rozhraní
    - editor máp

4.11.1.2 Grafika

- Potrebné typy dlaždíc:
  - Podložie pod mesto
  - Tráva
  - Piesok
  - Voda
  - Cesty
  - chodníky
  - Prechody medzi typmi
- Diskusia o možnosti generovať stromy z 3D modelov
- prijímame návrh a zariadime generovanie

Špecifikácia požiadaviek pre grafické návrhy

- **Typy grafických objektov:**
- Pet (2 pohlavia (nie do prototypu))
- NPCs: tútori pre:
    - Hudbu: dirigent vo fraku
    - Prvouku: človek s hlavou ako zemeguľa
    - Matematiku: kalkulačka s nohami a rukami
    - Čítanie: kniha s nohami a rukami
- Aspoň jeden typ avatara
- (do prototypu použiť už dodané noty)
- **Požiadavky:**
- Objekty schopné pohybu kresliť z ôsmich strán
    - Potrebné nakresliť 5 pohľadov
        - Doľava + doprava len symetricky otočené
        - Doľava hore + doprava hore len symetricky otočené
        - Dolu
        - Hore
        - Doľava dole + doprava dole len symetricky otočené
    - Diskusia o možnosti zjednodušiť tvorením 3D modelov
        - Zariadime vytvorenie prototypu a posúdime výsledok a vhodnosť tohto prístupu

4.11.2 Závery stretnutia

- Pokračujeme na prácach na prototype.
- Pre účely prototypu sa vzdávame myšlienky „svet na guli“ – nízka pridaná hodnota s ohľadom na námahu
- Implementujeme editor máp a v ňom jednoduchú mapu na účely prezentovania prototypu.
- Pracujeme na úlohách v súlade s nižšie špecifikovanými

4.11.3 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
10.1	Začať implementovať prototyp	27.11.2006	20.12.2006	Všetci	Prebieha
8.3	Zrevidovať architektúru – pridať model používateľa	13.11.2006	20.12.2006	MT	Prebieha
1.12	Bližšie špecifikovať hry	4.12.2006	11.1.2006	Všetci	Zahájené

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav
11.1	Implementovať A* algoritmus do prototypu klienta	5.12.2006	16.11.2006	MT	Plánované
11.2	Implementovať komunikačné rozhrania do	6.12.2006	9.12.2006	RV	Plánované

	klienta				
11.3	Nechať vypracovať 3D model návrhu postavy	4.12.2006	11.12.2006	RV	Začalo
11.4	Implementovať ¾ perspektívu do enginu	4.12.2006	11.12.2006	RV	Plánované
11.5	Napísať metodiku pre kreslenie štvorčekov mapy a kubických objektov	4.12.2006	5.12.2006	RV	Plánované
11.6	Zistiť možnosti zisťovania výšky tónu od P. Skřípala	4.12.2006	10.12.2006	RV	Plánované
11.7	Zariadiť vygenerovanie stromov pre prototyp	4.12.2006	11.12.2006	MT	Začalo
11.8	Implementovať editor máp	5.12.2006	11.12.2006	RV+OV	Plánované

4.12 Zápisnica k 12. stretnutiu

Zápisnica č.	Dátum	Miesto	Čas stretnutia
12	11.12.2006	Soft. Štúdio	13:00-15:00
Pedagóg	Prof. Ing. Mária Bieliková PhD.	Vypracoval	Andrej Frlička
Zúčastnení:	Bc. Andrej Frlička	(AF)	
	Bc. Marek Tomša	(MT)	
	Bc. Richard Veselý	(RV)	
	Bc. Oto Vozár	(OV)	

4.12.1 Priebeh stretnutia

- Predvádzanie výsledkov práce jednotlivých členov tímu
  - Marek Tomša popisoval vytvorený rámec komunikácie na príklade hry Hangman
    - Predstavil rámec komunikácie
    - Predstavil metodiku pridávania aktivít a komponentov na serveri
  - Rišo Veselý ukazoval výsledky práce na grafickej knižnici
    - Ukázal implementovaný nástroj zobrazovania animácie postáv
    - Ukázal implementovaný prototyp izometrického zobrazovania
  - Oto Vozár predviedol editor máp
    - Predviedol proces tvorby mapy
    - Vysvetlil plán ďalšieho zlepšovania editora
    - Ukázal vstupné a výstupné charakteristiky programu
  - Andrej Frlička predviedol postup práce na PET v PDA
    - Ukázal implementovaný systém prechodu medzi miestnosťami
    - Poukázal na ďalší postup pri implementácii
- Konzultácia ku používateľskej príručke
  - používateľská dokumentácia nemá mať charakter technickej dokumentácie, pretože ide o dokumentáciu prototypu na zahodenie
  - vhodné opísať dátové formáty a rozhrania medzi jednotlivými službami
  - opísať webové služby
  - pridať obrázky z vytvorených výsledkov
- Profesorka Bieliková oboznámila pravidlá, ktoré musia tímy dodržiavať pri predvádzaní prototypu
  - opísať čo sme prototypovali
  - opísať ako sme prototypovali
  - všetky výsledky nemusia byť robustné (nachádzajú sa v štádiu vývoja)
  - hneď po predvádzaní si sadnúť a napísať posudok

4.12.2 Závery stretnutia

- Pokračujeme v implementácii prototypu a zahajujeme integrovanie jeho súčasti
- Zahajujeme integráciu projektovej dokumentácie
- Pripravujeme sa na prezentáciu prototypu

4.12.3 Úlohy

Úlohy z minula					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav (%)
10.1	Začať implementovať prototyp	27.11.2006	20.12.2006	Všetci	Prebieha
8.3	Navrhnúť a implementovať službu modelu používateľa	13.11.2006	20.12.2006	MT	Plánované
11.1	Implementovať A* algoritmus do prototypu klienta	5.12.2006	16.12.2006	MT	Dokončené
11.7	Zariadiť vygenerovanie stromov pre prototyp	4.12.2006	11.12.2006	MT	Dokončené
11.8	Implementovať editor máp	5.12.2006	11.12.2006	RV+OV	Dokončené
11.2	Implementovať komunikačné rozhrania do klienta	6.12.2006	9.12.2006	RV	Prebieha
11.3	Nechať vypracovať 3D model návrhu postavy	4.12.2006	11.12.2006	RV	Prebieha
11.4	Implementovať ¾ perspektívu do enginu	4.12.2006	11.12.2006	RV	Dokončené
11.5	Napísať metodiku pre kreslenie štvorčekov mapy a kubických objektov	4.12.2006	16.12.2006	RV	Zahájené
11.6	Zistiť možnosti zisťovania výšky tónu od P. Skřípala	4.12.2006	10.12.2006	RV	Dokončené
1.12	Bližšie špecifikovať hry	4.12.2006	23.10.2006	Všetci	Prebieha

Nové úlohy					
ID	Opis úlohy	Začaté	Plánované	Kto	Stav (%)
12.1	Vytvoriť formát mapy	11.12.2006	18.12.2006	OV+RV	Zahájené
12.2	Upraviť editor mapy (export mapy)	11.12.2006	18.12.2006	OV	Zahájené
12.3	Dokumentovať časti prototypu	11.12.2006	18.12.2006	Všetci	Zahájené
12.4	Integrácia a testovanie prototypu	11.12.2006	18.12.2006	Všetci	Zahájené
12.5	Aktualizácia projektového plánu	11.12.2006	18.12.2006	MT	Zahájené
12.6	Vytvorenie pravidlového systému pre PET	11.12.2006	18.12.2006	AF	Zahájené
12.7	Implementovať prototyp PET	11.12.2006	18.12.2006	AF	Zahájené
12.8	Finalizovať projektovú dokumentáciu	11.12.2006	11.1.2006	AF, MT	Zahájené

5 Štandardy tímu

Kapitola štandardy tímu sa zaoberá metódami, ktoré sú platné pre všetkých členov tímu. Obsahujú základné informácie o často používaných postupoch pri práci vo vývojových prostrediach ako aj pri písaní projektovej dokumentácie.

5.1 Tvorba dokumentácie

Podkapitola obsahuje kostru dokumentu, ktorý sa používa ako podklad pri písaní dokumentácie ku tímovému projektu. Forma textu je upravená pre potreby dokumentu riadenia projektu.

5.1.1 Titulná strana

Podkapitola zobrazuje šablónu titulnej strany.

FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Titul

Podtitul 1

Podtitul 2

Vedúci projektu: *prof. Ing. Mária Bieliková PhD.*

Akademický rok: *2006/2007*

Semester: *Zimný*

Autori : *Bc. Andrej Frlička*
Bc. Marek Tomša
Bc. Richard Veselý
Bc. Oto Vozár

5.1.2 Štrukturovanie textu

Kapitola rozoberá štrukturovanie textu či už pre potreby sémantického oddelenia a navigácie.

5.1.2.1 Kapitoly

Kapitolám zodpovedá štýl *Heading 1*. Podkapitolám zodpovedajú štýly *Heading 2* resp. *3*. V prípade, že potrebujeme text jemnejšie štrukturovať, volíme Štýl *Heading 4 a 5*.

Pre každú kapitolu je zavedené samostatné číslovanie strán.

5.1.2.2 Zoznamy

Ak vyvstane potreba štrukturovať text kvôli prehľadnosti môžeme využiť číslované alebo nečíslované zoznamy.

Príklad nečíslovaného zoznamu ktorý tvoríme štýlom *Bulleted:*

- Položka 1. úrovne
  - Položka 2. úrovne
    - Položka 3. úrovne
    - Položka 3. úrovne

Príklad číslovaného zoznamu ktorý tvoríme štýlom *Numbered:*

1. Položka 1.úrovne
   a. Položka 2. úrovne
      i. Položka 3. úrovne

5.1.2.3 Tabuľky

Tabuľky majú štýl *Mriežka tabuľky* a ich text je potrebné nastaviť na štýl *Text tabuľky*. Tabuľku uvádzame textom, ktorý má štýl *popis*.

Tabuľka 5-1. Ukážka tvorby tabuľky

X	Y
Bunka 1	Bunka 2

5.1.3 Formátovanie textu

5.1.3.1 Číslovanie

Kapitoly a podkapitoly číslujeme arabskými číslicami ktoré sú oddelené bodkami.

Popisy obrázkov, tabuliek, príkladov aj vzorcov zahŕňajú čísla kapitol.

5.1.3.2 Zdrojový kód

Zdrojový kód v texte uvádzame štandardným spôsobom a nastavujeme mu štýl *code*, čím sa odlíši od okolitého textu. V prípade, že ide o samostatne stojaci dôležitý výrez z kódu, je vhodné pridať ku nemu popis.

```
public bool Equals(VariableBinder other)

{

Dictionary<string, string>.ValueCollection.Enumerator x=
this.binding.Values.GetEnumerator();

Dictionary<string, string>.ValueCollection.Enumerator x2=
this.binding.Values.GetEnumerator();

for (int i = 0; i < binding.Keys.Count; i++)

            {

if (x.Current != x2.Current) return false;

            }

return true;

}
```

Príklad 5-1. Titulok príkladu.

5.1.3.3 Poznámky

Poznámky do textu uvádzame za pomoci štýlu *DefCu.*

> Toto je text poznámky, ktorý je iba skúšobný. Predstavuje ukážku formátovania takéhoto typu textu.

5.1.3.4 Vzorce

V prípade, že chceme uviesť vzorec, použijeme štýl *Equation*. Popis použitého vzorca uvedieme v štýlom *popis.*

$$A = 3*B + C$$

Vzorec 5-1Výpočet lineárnej rovnice

5.1.3.5 Obrázky

Obrázky definujú vkladáme do textu priamo a využívame na ich správne naformátovanie štýl *figure*. Pod každým obrázkom vložíme popis, čím ho jednoznačne identifikujeme. Popis obsahuje číslo kapitoly, v ktorej sa text nachádza a číslo obrázku v rámci kapitoly.

Obrázok 5-1. Logo imagine cup.

5.1.4 Odkazovanie v texte

5.1.4.1 Referencie

Referencie na objekty v dokumente vkladáme pomocou príkazu :

Insert – Reference – Cross-reference.

Nastavíme typ referencie, zvolíme požadovaný objekt zo zoznamu a do textu sa automaticky vloží požadovaná referencia.

5.1.4.2 Literatúra

Použité zdroje sa uvádzajú vo formáte, ktorý predpisuje príslušná norma. Literatúra sa zapisuje použitím štýlov *reference* a *reference name.* Štýl reference name je založený na štýle reference ku ktorému pridáva kapitálky.

1. BIELIKOVÁ, M. (2003). Adaptívna prezentácia hypermédií na webe. In: Proc. of DATAKON 2003, L. Popelínský (Ed.), Brno, Október 18-21, pp. 72—91.

5.2 Práca v platforme .Net framework

Dokument poukazuje na základy práce v prostredí .Net framework, zvýrazňuje vhodné praktiky pri programovaní a pomáha odstraňovať nevhodné návyky.

5.2.1 Menné konvencie

5.2.1.1 Veľké a malé písmená

Existujú dva základné štýly použitia veľkých a malých písmen.

Pascal Casing

Pascal Casing dáva veľké písmeno na začiatok každého logického slova.

```
PrepertyDescriptor
IOStream
HtmlTag
```

Používajte Pascal Casing pre všetky verejné rozhrania (viď ďalej).

Camel Casing

Camel Casing dáva veľké písmeno na začiatok každého logického slova s výnimkou prvého.

```
propertyDescriptor
ioStream
htmlTag
```

Používajte Camel Casing pre názvy parametrov.

5.2.1.1.1 Pravidla použitia

Nasledujúca tabuľka definuje použitie štýlov pre jednotlivé prvky kódu

Tabuľka 5-2 použitie štýlov pre prvky kódu

Typ	Casing	Príklad
Class	Pascal	`public class StreamReader`
Enumeration type	Pascal	`public enum FileMode`
Enumeration value	Pascal	`FileMode.Append`

Event	Pascal	`public class Form`
		`{`
		`  public event EventHandler Closed`
		`}`

Exception class	Pascal	`public class FileNotFoundException`

Fields	Pascal	`private messageQueue`
		`{`
		`  public static readonly TimeSpan`

		`InfiniteTimeout;` `}`
Interface	Pascal	`public interface IEnumerable`

Method	Pascal	`public class Object`
		`{`
		`  public string ToString()`
		`}`

Namespace	Pascal	`namespace System.IO`

Property	Pascal	`public class String`
		`{`
		`  public int Length { get; }`
		`}`

Parameter	Camel	`public class Convert`
		`{`
		`  public static int ToInt32(string value);`
		`}`

Nezneužívajte veľké a malé písmená na vytváranie metód, ktoré sa navzájom líšia iba vo veľkosti písmen. Napríklad:

```
namespace ee.cummings;
namespace Ee.Cummings;

void foo(string a, string A)

int Foo {get, set};
int FOO {get, set}
```

5.2.1.1.2 Výber slov

Používajte pre názvy verejného rozhrania anglický jazyk. Používajte plný rozsah slov ,z názvu metódy by malo byť zrejmé aký je jej účel.

Nepoužívajte podtržníky , pomlčky ani žiadne iné pomocné znaky.

Nepoužívajte kľúčové slová štandardne používané v programovacích jazykoch.

5.2.1.1.3 Skratky

Nepoužívajte skrátené verzie slov. Lepšie je napísať GetWindow ako GetWin.

Nepoužívajte akronymy ktoré nie sú všeobecne známe. Pre akronymy dlhšie ako 2 znaky použite Pascal Casing, príklad HtmlButton ale napr. System.IO (2 znaky).

5.2.1.1.4 Oddelenie názvu typu a názvu premenné

Pre názvy premenných (parametrov) používajte meno, ktoré vystihuje jej účel a nie meno odvodené od typu premennej.

V prípade že u premennej nepoznáte jej sémantický význam použite generické meno. Príklad:

```
void Write(double value);
void Write(float value);
```

5.2.1.2 Názvy pre Namespace

Namespace alebo menný priestor slúži na hierarchické rozčlenenie typov do logickej štruktúry. Šablóna pre názov menného priestoru má vzor:

<Company>.<Technológie/Produkt>.<interné logické členenie>

Príklad :

```
Microsoft.VisualStudio.Design
```

Ako názov namespace používajte Pascal Casing a tam, kde je to možné (mimo skratiek a obchodných mien) množné číslo.

Príklad:

System.Collections je lepší ako System.Collection

Nepoužívajte rovnaké meno pre triedy a menné priestory.

5.2.1.3 Názvy typov

Typ je základný stavebný kameň .Net Frameworku. Pojem typ zahrňuje jednoduché dátové štruktúry ako napr. číslo, dátum , ako aj zložitejšie konštrukcie ako napr. . triedy a štruktúry.

Pre názvy typov odvodených od štandardných systémových typov platí nasledujúca tabuľka.

Tabuľka 5-3 Dekorovanie mien systémových typov

Základné Typy	Názov odvozeného typu
System.Attribute	suffix "Attribute"
System.Collections.Icollection	suffix "Collection"
System.Delegate	suffix "EventHandler" pre obsluhu událaostí a "Callback" pre callback. nepoužívajte suffix pre čistý "Delegate"
System.EventArgs	suffix "EventArgs"
System.Exception	suffix "Exception"

Príklad, ClickedEventArgs je typ odvodený od System.EventArgs.

5.2.1.3.1 Názvy tried a štruktúr

Triedy a štruktúry reprezentujú dátové typy, ktoré obaľujú dáta a metódy, ktoré s nimi pracujú. Pre názvy používajte podstatné mená v Pascal Casing a nepoužívajte prefixy (ako napr. C pre class). Pre odvodené triedy je možné použiť zložené slová.

Príklad:

```
public class FileStream : Stream
{…
}
```

5.2.1.3.2 Názvy pre interface

Interface je špeciálny typ , ktorý predstavuje definíciu kontraktu , ktorý musí byť pri implementácii dodržaný. Pre názvy interface sa opäť použije Pascal Casing a striktne prefix I pred názvom každého interface.

5.2.1.3.3 Názvy enumeračných typov

Enumeračný typ (enum) je špeciálny typ , ktorý dáva alternatívne pomenovanie hodnotám primitívneho dátového typu (typicky integer).

Pre názvy enumeračných typov sa používa Pascal Casing tak pre názov, ako aj pre hodnoty enumeračného typu. Nepoužívajte prefixy ani sufixy (napríklad adXXX pre ADO enumeračné typy)

Príklad:

Nesprávne:

```
public enum PhotoshopMode
{
   PhotoshopModeBitmap = 0,
   PhotoshopModeGrayscale = 1,
   PhotoshopModeIndexed = 2,
}
```

Správne:

```
public enum PhotoshopMode
{
   Bitmap = 0,
   Grayscale = 1,
   Indexed = 2,
}
```

5.2.1.4 Názvy členov

Triedy, štruktúry , interface reprezentujú typy s vnútornou štruktúrou. Vnútorný prvok sa nazýva člen (member) a môže to byť :

- Metóda
- Udalosť
- Vlastnosť
- Pole (field, nie array)

5.2.1.4.1 Názvy metód

Pre názvy metód používajte slovesá a Pascal Casing, nepoužívajte maďarskú notáciu. Príklad:

```
RemoveAll()
GetCharArray()
```

5.2.1.4.2 Názvy vlastností (property)

Pre názvy vlastností používajte podstatná mená a Pascal Casing, nepoužívajte maďarskú notáciu.

Príklad:

```
public class Button
{
    public Color BackColor
    {
    }
}
```

Nepoužívajte názvy, ku ktorým existuje obdobná metóda s prefixom Get, príklad:

Nesprávne

```
public string TextWriter
{
get {...}
set {...}
}
public string GetTextWriter()
{
...
}
```

Pre vlastnosti typu boolean používajte pozitívne názvy

Príklad:

- *CanSeek* je lepší ako *CantSeek*.

Voliteľne môžete použiť prefix Is, Can, Has, ale len tam, kde to zvyšuje čitateľnosť.

Príklad:

CanRead je zrozumiteľnejší ako Readable ale naopak Created je vhodnejší ako IsCreated.

5.2.1.4.3 Názvy udalostí

Pre názov udalostí používajte EventHandler sufix a Pascal Casing, nepoužívajte maďarskú notáciu.

Príklad:

```
public delegate void ClickedEventHandler(object sender,
ClickedEventArgs e);
```

```
```

Vždy používajte 2 parametre - sender a e.

Príklad:

```
public delegate void <Some>EventHandler(object sender, <Some>EventArgs
e);
```

Pre triedy argumentov používajte sufix EventArgs.

Príklad:

```
public class ClickedEventArgs : EventArgs
{
   int x;
   int y;
   public ClickedEventArgs (int x, int y)
{
      this.x = x;
      this.y = y;
   }

   public int X { get { return x; } }
   public int Y { get { return y; } }
}
```

Pre názvy udalostí reprezentujúce koncept pred-po akcii, používajte priebehový a minulý čas (a nie BeforeXxx\AfterXxx vzor). Napríklad akcia close bude vystavovať udalosti Closing a Closed.

5.2.1.4.4 Názvy polí (fields)

Pre názvy polí používajte podstatná mená a camel casing s prefixom „_“. Neexistujú public fields, všetko musí byť zabalené, pomocou vlastnosti alebo metódy.

5.2.1.5 Názvy argumentov

Metódy typicky majú argumenty. Pre názvy argumentov používajte camel Casing a nie maďarskú notáciu. Používajte popisné a výstižné názvy parametrov založené na význame a nie na jeho type. Nepoužívajte vyhradené(reserved) parametre pre potreby budúcich verzií. V prípade potreby je možné metódu jednoducho preťažiť.

5.2.1.6 Názvy Assembly/DLL

Assembly reprezentuje základnú distribučnú jednotku aplikácie v .Net Frameworku

Názov assembly je tvorený podľa nasledujúceho vzoru:

```
<Company>.<Component>.dll
```

Príklad:

```
    Manawydan.Framework.dll
```

5.2.2 Pravidlá dizajnu

Nasledujúca kapitola obsahuje pravidlá a odporučenia , ktoré je vhodné aplikovať pri návrhu typov a ich vnútornej implementácii.

5.2.2.1 Návrh typov

Pri návrhu všetkých typov dodržujte nasledujúce obecné pravidlá:

- Definície typu musia byť vo vnútri namespace.
- Používajte častejšie triedy ako interface
- Každá trieda má obsahovať explicitní konštruktor. V prípade, že trieda nie je verejne inštancovateľná , deklarujte konštruktor ako privátny.

5.2.2.1.1 Návrh tried pre dedičnosť

Základné triedy (predkovia pri implementácii dedičnosti) reprezentujú vhodný spôsob zaobalenia spoločnej funkcionality medzi niekoľkými triedami s možnosťou špecializácie v potomkoch. Z pohľadu verzií , je dedičnosť tried flexibilnejšia ako použitie interface

Interface preto používajte keď:

Triedy, ktoré navzájom nesúvisia majú poskytovať rovnaké rozhranie.

Triedy dedia od iných tried.

V ostatných prípadoch je implementácia dedičnosti lepšia.

Implementujte chránené (protected) virtuálne metódy na základných triedach, ktoré umožnia potomkom vlastné rozšírenie základnej funkcionality.

Verejné rozhranie (public metódy a vlastnosti) má poskytovať funkcionalitu pre používateľa triedy a jeho metódy nesmú byť virtuálne.

Príklad:

```
public Control
{
   //…
   public void SetBounds(int x, int y, int width, int height)
{

      …
      SetBoundsImpl (…);
   }
   public void SetBounds(int x, int y, int width, int height,
 BoundsSpecified specified)
{

       …
       SetBoundsImpl (…);
   }

   protected virtual void SetBoundsImpl(int x, int y, int width,
int height, BoundsSpecified specified)
{
         // Do the real work here.
```

```
    }
}
```

Definujte chránený (protected) konštruktor na všetkých abstraktných triedach, a zároveň nesmie existovať konštruktor verejný.

V potomkoch nemeňte prístupový modifikátor (public,protected,private) k metódam.

Nepoužívajte "sealing" tried, pokiaľ to nie je naozaj nutné.

Obmedzte počet a komplexnosť virtuálnych metód.

V prípade preťažovaných metód implementujte ako virtuálne tie s najkomplexnejším rozhraním, príklad:

```
public class Foo
{
   private const string defaultForA = "a default";
   private const int defaultForB = 42;
   protected void Bar()
{
      Bar(defaultForA, defaultForB);
   }
   protected void Bar (string a)
{
      Bar(a, defaultForB);
   }
   protected virtual void Bar (string a, int b)
{
      // core implementation here
   }
}
```

5.2.2.1.2 Návrh interface

Pred samotnou implementáciou interface zvážte, či namiesto interface nepoužiť triedu alebo abstraktní triedu.

Je vhodné poskytnúť triedu, ktorá reprezentuje základnú implementácii interface.

Príklad:

System.Collections.DictionaryBase je default implementácia pre System.Collections.IDictionary interface.

Nepoužívajte interface len na označenie triedy (interface bez jediného člena). Na označenie triedy používajte vlastné atribúty.

Príklad:

Nesprávne

```
public interface IImmutable {} // empty interface
public class String : IImmutable { …}
```

Správne

```
[Immutable]
public class Key {…}
```

```

```

Výnimkou môže byť nutnosť kontroly označenia v okamihu kompilácie (atribúty sa kontrolujú za behu). Príklad:

```
public interface ITextSerializable {} // empty interface
public void Serialize(ITextSerializable item)
{
    // use reflection to serialize all public properties
    …
}
```

5.2.2.2 Návrh hodnotových typov (štruktúr)

Používajte štruktúry pre implementáciu typu v nasledujúcich prípadoch:

Typ sa chová ako primitívny typ (int, date).

Veľkosť inštancie typov je <= 16 bytov

Typ je nemenný

Chcete aby sa choval ako "value type" (napr. z dôvodov správy pamäti).

5.2.2.2.1 Návrh enumeračných typov

- Používajte enumeračné typy (enum) na podporu silného typovania parametrov, vlastností a návratových hodnôt.

Príklad:

```
public enum TypeCode
{
    Boolean,
    Byte,
    Char,
    DateTime,
    …
}


Convert.ChangeType(object value, TypeCode typeCode);
```

- Preferujte enumeračné typy pred statickými konštantami.
- Nepoužívajte enumeračné typy pre neuzavreté zoznamy (napr. verzie OS a pod.).
- Používajte System.FlagsAttribute atribút pre enumerácie len v prípade, keď ohodnotenia typov reprezentujú bitovú masku. Pre bitové masky je vhodné poskytnúť hodnoty, ktoré reprezentujú časté kombinácie bitov

Príklad:

```
[Flags()]
public enum WatcherChangeTypes
{
```

```
    Created = 1,
    Deleted = 2,
    Changed = 4,
    Renamed = 8,
    All = Created | Deleted | Changed | Renamed
}
```

- Vykonávajte kontrolu argumentu i u enumeračných typov.

Príklad:

```
public void PickColor(Color color)
{
    switch (color)
{
           case Red:
                  ...
                  break;
           case Blue:
                  ...
                  break;
           case Green:
                  ...
                  break;
           //repeat for all known values of Color
           default:
                  throw new ArgumentOutOfRangeException();
                  break;
   }
}
```

- Nepoužívajte enumeračné typy s jedinou hodnotou.

5.2.2.3 Návrh členov (members)

5.2.2.3.1 Návrh vlastností (property)

- Implementujte zmysluplné implicitné hodnoty.
- Zachovávajte pôvodnú hodnotu v prípade, keď kód pre "set" vyvolá výnimku. Vlastnosti môžu byť nastavované v rôznom poradí, vnútorná implementácia by mala byť bezstavcová voči ostatným vlastnostiam.
- Je vhodné vystaviť informáciu o zmene vlastnosti formou udalostí, typicky sa implementujú udalostí pred a po zmene.

Implementácia udalostí pred zmenou môže zmenu vlastnosti prerušiť vyvolaním výnimky. Meno udalostí sa skladá z mená vlastnosti a sufixu "ing" (napr. TextChanging).
Príklad:

```
public class TextBox
{

    public event TextChangingEventHandler TextChanging;

    public string Text
 {
        get { return text; }
        set
 {
            if (text != value)
  {
                OnTextChanging(Event.Empty);
                text = value;
            }
        }
    }
}
```

Implementácia udalostí po zmene neumožňuje zrušenie nastavenia hodnoty vlastnosti, umožňuje ale implementáciu "aplikačného triggeru". Meno udalostí sa skladá z mená vlastnosti a sufixu "ed" (napr. TextChanged).

Príklad:

```
class TextBox
{
    public event TextChangedEventHandler TextChanged;
    public event TextChangingEventHandler TextChanging;

    public string Text
 {
        get { return text; }
        set
 {
            if (text != value)
  {
                OnTextChanging(Event.Empty);
                text = value;
                OnTextChanged(…);
            }
        }
    }

    protected virtual void OnTextChanged(…)
```

```
 {
        TextChanged(this,…);
    }
    protected virtual void OnTextChanging(…)
 {
        TextChanging(this,…);
    }

}
```

Nevyvolávajte výnimky v metóde “get” vlastnosti.

5.2.2.3.2 Vlastnosti vs. Metódy

Vlastnosti použite keď:

- Člen reprezentuje logické zhromaždisko dát

```
string Name { get, set }
// Name reprezentuje logický atribut triedy, proto by měl být
// implementován ako property

Guid Guid.GetNext()
// Guid obvykle nemá atribut Next, takže je vhodné implementovat
// ako metodu
```

Metódu použite keď:

- Operácia reprezentuje konverziu (Object.ToString())
- Operácia je časovo náročná (nastavení vlastnosti by nemalo byť náročné)
- Opakované volanie nemusí skončiť rovnakým výsledkom.
- Člen reprezentuje pole.

5.2.2.3.3 Read-Only a Write-Only vlastnosti

- Používajte read-only vlastnosti keď nie je povolená zmena vlastnosti po inicializácii.
- Nepoužívajte set-only vlastnosti. V tomto prípade implementujte metódu.

5.2.2.3.4 Návrh indexerov

- Používajte indexery na sprístupnenie informácie v poli.

```
public char this[int index] {get;}
```

- Používajte iba System.Int32, System.Int64, System.String, a System.Object ako index indexeru. V prípade iného alebo zložitejšieho indexu je vhodné implementovať namiesto indexeru metódu.

5.2.2.3.5 Návrh udalostí

- Návratová hodnota udalostí musí byť void.

Príklad:

```
public delegate void ClickedEventHandler(object sender,
ClickedEventArgs e);
```

- V prípade, že udalosť zaobaľuje špecifické dáta, je nutné implementovať dátovú triedu pre argumenty udalostí. Tato trieda musí byť zdedená od System.EventArgs.

Príklad:

```
public class ClickedEventArgs : EventArgs
{
}
```

- Vyvolávajte (raise) udalostí v chránených virtuálnych metódach triedy.
- Implementujte udalosť s podporou možnosti zrušení akcie.

5.2.2.3.6 Metódy

- Implicitne používajte nevirtuálne metódy.
- Nenavrhujte metódy, ktoré nemôžu byť volané na inštancii triedy inicializované základným konštruktorom. Ak metóda aj tak vyžaduje explicitnú inicializáciu, ošetrite v implementácii metódy prípad volanie pred explicitnou inicializáciou - vyvolajte výnimku ktorá jednoznačne deklaruje čo má byť inicializované.

5.2.2.3.7 Preťažovanie metód

- Používajte preťažovanie, keď implementujete rôzne metódy z rovnakým sémantickým významom.
- Používajte korektné implicitné hodnoty. V skupine preťažených metód, metóda s najväčším počtom parametrov by mala nazývať parametre ktoré indikujú zmenu oproti implicitnej hodnote v metóde s menším počtom parametrov. Typicky sa to týka parametrov typu boolean. V nasledujúcom príklade bude 1. metóda vyhľadávať v case sensitive módu. V metóde 2 je druhý parameter nazývaný ignoreCase namiesto caseSensitive pretože lepšie indikuje ako sa mení implicitné správanie.
- Príklad:

```
1: MethodInfo Type.GetMethod(String name);  //ignoreCase = false
2: MethodInfo Type.GetMethod (String name, boolean ignoreCase);
```

- Typicky sa používa pre implicitné hodnoty nulový stav (0, 0.0, false, "", atď.).
- Neimplementujte skupiny metód, kde metóda s najmenším počtom parametrov má viac ako 3 parametre. V prípade konštruktora je možné povoliť 5 parametrov.

- Dodržujte konzistenciu v radení a pomenovávaní parametrov.

Príklad:

```
public class Foo
{
   readonly string defaultForA = "default value for a";
   readonly int defaultForB = 42;
   readonly double defaultForC = 68.90;

   public void Bar()
{
      Bar(defaultForA, defaultForB, defaultForC);
   }

   public void Bar(string a)
{
      Bar(a, defaultForB, defaultForC);
   }

   public void Bar(string a, int b)
{
      Bar(a, b, defaultForC);
   }

   public void Bar(string a, int b, double c)
{
      // core implementation here
   }
}
```

5.2.2.3.8 Premenlivý počet argumentov

Metódy, ktoré môžu byť volané s neobmedzeným počtom parametrov sú implementované s posledným argumentom ako

Pole

```
string String.Format(string format, object[] parameters);
       String.Format("{0} {1} {2} {3}", new object[] {1, 2, 3, 4});
```

- Za pomoci kľúčového slova params

```
    string String.Format(string format, params object[] parameters);
    String.Format("{0} {1} {2} {3}", 1, 2, 3, 4);
```

Používajte params namiesto množiny preťažovaných metód s parametrami rovnakého významu.

5.2.2.4 Návrh konštruktorov

- Implementujte jednoduché, v optimálnom prípade bezparametrické konštruktory

V prípade, že trieda slúži ako kontajner statických metód implementujte iba privátny konštruktor

```
public sealed class Enviroment
{
   private Enviroment(); // Prevents the class from being created.
   //…
}
```

- Implementujte statické konštruktory ako privátne.
- Nevolajte virtuálne metódy rovnakej triedy z konštruktorov.
- Používajte výnimky pre chybové stavy i v konštruktoroch.
- Poskytujte parametre v konštruktoroch pre explicitné nastavenia niekoľkých vlastností jedným volaním.

5.2.2.4.1 Dátové pole

- Nevystavujte inštančné dátové premenné(fields), dáta vystavte vždy cez vlastnosti.
- Používajte konštantné pole pre dáta, ktoré sa nikdy nezmenia.

Príklad:

```
public struct Int32
{
   public const int MaxValue = 0x7fffffff;
   public const int MinValue = unchecked((int)0x80000000);
}
```

- Používajte static readonly pole pre preddefinované inštancie objektov.

Príklad:

```
public struct Color{
   public static readonly Color Red = new Color(0x0000FF);
   public static readonly Color Green = new Color(0x00FF00);
   public static readonly Color Blue = new Color(0xFF0000);
   public static readonly Color Black = new Color(0x000000);
   public static readonly Color White = new Color(0xFFFFFF);

   public Color(int rgb)
{ }

   public Color(byte r, byte g, byte b)
{ }
```

```
   public byte R
{
get {...}
}
   public byte G
{
get {...}
}
   public byte B
{
get {...}
}
}
```

- Nepoužívajte prefixy, ani pre rozlíšenie statický - inštančný.

5.2.2.5 Návrh parametrov

5.2.2.5.1 Argumenty

- Kontrolujte platnosť argumentov v každej public alebo protected metóde a prípadne vyvolajte patričné výnimky. Typicky ide o potomka System.ArgumentException.

Príklad:

```
class Foo
{
public int Count
{
           get
{
           return count;
       }
       set
{
           if (value < 0 || value >= MaxValue)
throw new ArgumentOutOfRangeException(Sys.GetString(

           "InvalidArgument",
"value",
value.ToString()
));
                  count = value;
}
}
```

```
public void Select(int start, int end)
{
if (start < 0)
throw new ArgumentException(Sys.GetString(

    "InvalidArgument",

    "start",

    start.ToString()

    ));
        if (end < start)
            throw new ArgumentException(Sys.GetString(

    "InvalidArgument",
                                                                "end",

    end.ToString()

    ));
}
}
```

5.2.2.5.2 Odovzdávanie parametrov

- Vyhýbajte sa odovzdávaniu "reference types" referencií, odovzdávajte radšej hodnotou. Aj v tomto prípade sa totiž odovzdáva len odkaz na odovzdávanú inštanciu a to je vo väčšine prípadov vyhovujúce.
- Nevystavujte metódy, ktoré majú ako parametre ukazovateľ alebo viacrozmerné pole.

5.2.2.5.3 Oddelenie parametrov a vnútorných členov

- Používajte kľúčové slovo "this" keď referencujete inštančné členy.

Príklad:

Nesprávne

```
public Employee (string myName, int myExtension)
{
name = myName;
extension = myExtension;
}
```

Správne

```
public Employee (string name, int extension)
{
this.name = name;
```

```
this.extension = extension;
}
```

5.2.3 Rozvrhnutie kódu

5.2.3.1 Zátvorky

Používajte zátvorky vo vnútri výrazu keď je použitých niekoľko operátorov (+, -, *, /, <, =, >, <>, alebo logický operátor). Eliminuje sa tak diskusia o prednosti operátorov.

Príklad:

Nesprávne

```
if (isOpen && isAvailable || isRequired)
    total = fees + subtotal * tax;
```

Správne

```
if (isOpen && (isAvailable || isRequired))
    total = (fees + subtotal) * tax;
```

Otváracia zátvorka bloku kódu má byť na novom riadku ako začiatok bloku. (dá sa nastaviť vo Visual Studiu) .

Príklad:

Dohodnuté použitie zátvoriek

```
if (a > 2)
{
    b = 3;
    c = 4;

    if (d < 5)
    {
        e = 6;
    }

    if (f < 5)
    {
        g = 6;
        h = 7;
 }
 i = 8;
}
```

Nevhodné použití zátvoriek

```
if (a > 2) {
    b = 3;
    c = 4;
```

```
    if (d < 5) {
        e = 6;
    }

    if (f < 5) {
        g = 6;
        h = 7;
    }
    i = 8;
}
```

Jedinou výnimkou , keď sa pripúšťa použitie otváracej zátvorky na tom istom riadku je použitie v definícii vlastnosti pri get a set. Ak obsahuje len jeden príkaz. potom môže vyzerať takto:

```
public string Width
{
    get{ return this._width;}
    set{ this._width = value;}
}
```

5.2.3.2 Medzery

Používajte medzery na zvýšenie čitateľnosti logických výrazov. Oddeľujte kľúčové slova od identifikátorov, operátorov a ostatných reťazcov.

Pridajte medzeru k vonkajšej strane zátvoriek s výnimkou parametrov metódy, nie je dôvod pridávať medzeru z vnútornej strany zátvoriek.

Príklad:

Nesprávne

```
if( (subtotal<total )&&( ( total<credit ))
    tax=invoice.CalculateTax () ;
```

Správne

```
if ((subtotal < total) && (total < credit))
    tax=invoice.CalculateTax();
```

5.2.3.3 Riadok kódu

Na jednom riadku kódu umiestnite len jednu dátovú deklaráciu.

Príklad:

Nesprávne

```
int customerCount; int vendorCount;
   string customerName, vendorName;
```

Správne

```
int customerCount;
int vendorCount;
```

```
string customerName;
string vendorName;
```

Parametre a výrazy sa snažte udržať v jednom riadku

Príklad:

Nesprávne

```
if ((total > creditLimit) &&
    (requireAuthorization == true))
    order.CalcTotal(subtotal,
        miscellaneous,
        fees,
        freight);
```

Správne

```
if ((total > creditLimit) && (requireAuthorization == true))
    order.CalcTotal(subtotal, miscellaneous, fees, freight);
```

Ak už musíte výraz rozdeliť, umiestnite logický operátor spojujúci výrazy na začiatok nového riadku.

Nesprávne

```
if ((total > creditLimit) &&
(requireAuthorization == true))
```

Správne

```
if ((total > creditLimit)
&& (requireAuthorization == true))
```

5.2.3.4 Tabulátory

Používajte tabulátor.

Tabulátory je treba zachovávať a nenahradzovať ich výskyt používaním medzier..

5.3 Metodika pre programovanie komponentov aktivít na serveri

5.3.1 Úvod

Tento dokument predstavuje návod, ako programovať komponenty serverových aktivít vo výukovom hernom systéme. Je určená pre členov tímu Resharpers, ktorí implementujú komponenty aktivít na strane servera.

5.3.1.1 Súvisiace dokumenty

- Technická dokumentácia – komunikačný rámec
- Technická dokumentácia – rámec aktivít
- Štandardy programovania v jazyku C#

5.3.1.2 Použité skratky

VS – Microsoft Visual Studio 2005

5.3.1.3 Pojmy

Aktivita – typicky hra, implementovaná ako služba na serveri

Trieda aktivít – niekoľko inštancií danej aktivity. Riadi vytváranie inštancií aktivity a registrovanie a prideľovanie hráčov k inštanciám aktivity.

Trieda implementujúca triedu aktivít – implementácia triedy aktivít vo forme triedy v jazyku C#

5.3.2 Tvorba aktivity

Táto kapitola popisuje postup pri vytváraní novej aktivity. Na príklade aktivity Hangman ukazuje použitie rámca aktivít na vytvorenie aktivity. Zároveň je predpisom, ako organizovať súbory a pomenúvať jednotlivé typy pri vytváraní aktivity.

5.3.2.1.1.1 Adresáre

Pre novú aktivitu sa vytvorí vo VS v projekte Server. World podadresár v adresári Games s názvom zhodným s názvom aktivity. Na tento adresár sa dokument ďalej odkazuje ako na adresár aktivity.

5.3.2.1.1.2 Príklad:

V adresári Games sa vytvorí podadresár s názvom Hangman.

5.3.2.1.1.3 Triedy

Do adresára aktivity sa pridá trieda s názvom <Názov aktivity>Game. V prípade, že nejde o hru, použije sa sufix Activity. Na súbor, ktorý VS vytvorí pre túto triedu, referuje tento dokument ďalej ako na súbor aktivity. Na názov triedy sa referuje ako názov triedy aktivity.

5.3.2.1.1.4 Príklad:

Do adresára **\Games\Hangman** pridáme triedu s názvom **HangmanGame**.

5.3.3 Implementácia

Všetky typy pre danú aktivitu sú zapísané v súbore aktivity. Pre vytvorenie aktivity je potrebné deklarovať rozhrania a triedy implementujúce rozhrania, popísané v technickej dokumentácii k rámcu aktivít, odvodené z tried GameBase a GameInstance.

5.3.3.1 Rozhrania

Implementujú sa rozhrania s nasledovným významom:

Tabuľka 5-4 Použité rozhrania aktivít

Pomenovanie	Význam
Rozhranie inštancie aktivity	Deklaruje všetky metódy jednej inštancie hry na serveri
Rozhranie triedy aktivít	Deklarácia pre zabezpečenie genericity v „rámci aktivít". Implementuje ho trieda predstavujúca triedu aktivít danej aktivity
Rozhranie implementované na strane klienta (callback rozhranie)	Deklarácia metód, ktoré volá server smerom k pripojeným klientom

5.3.3.1.1 Rozhranie inštancie aktivity

5.3.3.1.1.1 Pomenovanie

Rozhranie inštancie aktivity sa pomenuje nasledovne:

I<názov triedy aktivity>Instance

5.3.3.1.1.2 Nadtriedy

Žiadne

5.3.3.1.1.3 Atribúty

Rozhranie inštancie aktivity sa označí nasledovným atribútom:

```
[ServiceContract(SessionMode=SessionMode.Required)]
```

5.3.3.1.1.4 Význam

Rozhranie deklaruje všetky metódy jednej inštancie hry na serveri.

5.3.3.1.1.5 Príklad

```
[ServiceContract(SessionMode=SessionMode.Required)]
public interface IHangmanGameInstance
{
        [OperationContract(IsOneWay = true)]
        void TryLetter(char c);
}
```

5.3.3.1.2 Rozhranie triedy aktivít

5.3.3.1.2.1 Pomenovanie

Pomenuje sa nasledovne:

I<názov triedy aktivity>

5.3.3.1.2.2 Nadtriedy

Rozhranie triedy hier sa implementuje ako prázdne rozhranie (bez deklarácií metód) implementujúce rozhranie inštancie aktivity a rozhranie IGameContract.

5.3.3.1.2.3 Atribúty

Rozhranie sa označí nasledovným atribútom:

```
[ServiceContract(SessionMode=SessionMode.Required,
CallbackContract=typeof(...))]
```

Ako hodnota parametra CallbackContract sa použije typ callback rozhrania aktivity (pozri kapitolu 5.3.3.1.3).

5.3.3.1.2.4 Význam

Rozhranie slúži ako deklarácia pre zabezpečenie genericity v „rámci aktivít“. Implementuje ho trieda zastrešujúca triedy aktivít danej aktivity.

5.3.3.1.2.5 Príklad

```
[ServiceContract(SessionMode=SessionMode.Required,
CallbackContract=typeof(IHangmanCallback))]
    public interface IHangmanGame : IHangmanGameInstance, IGameContract
    {


    }
```

5.3.3.1.3 Callback rozhranie

5.3.3.1.3.1 Pomenovanie

Callback rozhranie sa pomenuje nasledovne:

I<názov aktivity>Callback.

5.3.3.1.3.2 Nadtriedy

Rozhranie implementuje rozhranie IGameCallback.

5.3.3.1.3.3 Atribúty

Rozhranie sa označí nasledovným atribútom:

```
[ServiceContract(SessionMode=SessionMode.Required)]
```

5.3.3.1.3.4 Význam

V callback rozhraní sa deklarujú metódy, ktoré volá server smerom k pripojeným klientom.

5.3.3.1.3.5 Príklad

```
[ServiceContract(SessionMode=SessionMode.Required)]
public interface IHangmanCallback : IGameCallback
{
        [OperationContract(IsOneWay = true)]
        void Update(HangmanGameState word);
        [OperationContract(IsOneWay = true)]
        void PlayerGuessed(PlayerToken who, char guessedChar);
        [OperationContract(IsOneWay=true)]
        void YourTurn(HangmanGameState state);
}
```

5.3.3.2 Triedy

Pri implementácii aktivity sa musia implementovať aspoň nasledovné triedy:

- Trieda implementujúca triedu aktivít
- Trieda implementujúca inštanciu aktivity

5.3.3.2.1 Trieda aktivít

Pomenovanie

Trieda implementujúca triedu aktivít má názov zhodný s triedou, ktorej vytvorenie je popísané v kapitole 0.

5.3.3.2.1.1 Nadtriedy

Trieda aktivít dedí z triedy **GameBase<TGameInstance, TCallback>** a implementuje rozhranie triedy aktivít (viď 5.3.3.1.2).

Ako typový parameter **TGameInstance** sa použije trieda inštancie aktivity (viď 5.3.3.2.2).

Ako typový parameter **TCallback** sa použije callback rozhranie aktivity (viď 5.3.3.1.3).

5.3.3.2.1.2 Atribúty

Trieda aktivít sa označí nasledovným atribútom:

```
[ServiceBehavior(ConcurrencyMode=ConcurrencyMode.Multiple,
InstanceContextMode=InstanceContextMode.Single)]
```

5.3.3.2.1.3 Význam

Trieda aktivít slúži na zastrešenie viacerých inštancií danej aktivity. Na serveri prebieha paralelne niekoľko inštancií danej aktivity. Trieda aktivít riadi prihlasovanie hráčov k inštanciám aktivity, vytváranie a rušenie inštancií aktivity a registrovanie hráčov pre inštancie aktivity. Viac o význame a implementácii triedy aktivít je v technickej dokumentácii rámca aktivít.

5.3.3.2.1.4 Ďalšie poznámky

Trieda aktivít sa implementuje podľa vzoru unikát (singleton).

5.3.3.2.1.5 Príklad

```
public class HangmanGame : GameBase<HangmanGame.HangmanGameInstance,
IHangmanCallback>, IHangmanGame
{
    #region HangmanGame Singleton
    public static HangmanGame Instance
    {
           get
           {
                  return NestedHangmanGame.instance;
           }
    }
    class NestedHangmanGame
    {
           // Explicit static constructor to tell C# compiler
           // not to mark type as beforefieldinit
           static NestedHangmanGame()
           {
           }
           internal static readonly HangmanGame instance = new
HangmanGame();
    }
    #endregion

private HangmanGame()
    {
    }

#region IHangmanGame Members
    public void TryLetter(char c)
{


    }
    #endregion
}
```

5.3.3.2.2 Trieda inštancie aktivity

5.3.3.2.2.1 Pomenovanie

Trieda inštancie aktivity sa implementuje ako vnorená (nested) trieda triedy implementujúcej triedu aktivít.

5.3.3.2.2.2 Nadtriedy

Trieda inštancie aktivity dedí z triedy GameInstance<T> a implementuje rozhranie inštancie aktivity.

5.3.3.2.2.3 Atribúty

Žiadne

5.3.3.2.2.4 Význam

Trieda inštancie aktivity zabezpečuje realizáciu logiky inštancie danej aktivity. Metódy volané na strane klienta sú implementované v tejto triede.

5.3.3.2.2.5 Príklad

Príklad uvádza aj triedu implementujúcu triedu aktivít na ilustráciu vnorenia triedy inštancie aktivít.

```
public class HangmanGame : GameBase<HangmanGame.HangmanGameInstance,
IHangmanCallback>, IHangmanGame
{
    public class HangmanGameInstance : GameInstance<IHangmanCallback>,
IHangmanGameInstance
    {
    //implementácia všetkých metód rozhrania IHangmanGameInstance
    //implementácia zvyšku hernej logiky
    }
}
```

5.3.4 Integrácia - server

Táto kapitola popisuje kroky, ktoré treba vykonať pre správnu konfiguráciu a integráciu implementovanej aktivity, aby bežala pri každom spustení servera.

5.3.4.1 Konfigurácia ServiceHost-u

V súbore app.config v projekte Server sa pridá do sekcie <services> nasledovná podsekcia:

```
<service behaviorConfiguration="metadataSupport"
       name="Server.Core.Games.Hangman.HangmanGame">
<endpoint
binding="wsDualHttpBinding"
contract="Server.Core.Games.Hangman.IHangmanGame" />
      <endpoint
address="mex"
binding="mexHttpBinding"
contract="IMetadataExchange" />
      <host>
        <baseAddresses>
```

```
            <add baseAddress="http://localhost:8080/Hangman/" />
          </baseAddresses>
        </host>
</service>
```

Tabuľka 5-5 Typy hodnôt atribútov špecializovanej podsekcie service

Atribút	Hodnota
Name	plné meno (fully qualified name) triedy implemenetujúcej triedu aktivít
Contract (mex endpoint)	IMetadataExchange
Contract (endpoint kontraktu aktivity)	plné meno rozhrania triedy aktivít
baseAddress	„http://localhost:8080/“ + názov aktivity

5.3.4.2 Konfigurácia HostManager-a

V projekte Server, v súbore HostManager.cs, v triede HostManager, v metóde `internal static void StartServices()` sa pridajú nasledovné riadky:

```
ServiceHost gameHost = new ServiceHost(HangmanGame.Instance, new Uri[]
{ });
hosts.Add(gameHost);
```

Ako prvý argument konštruktora triedy ServiceHost sa použije referencia na singleton inštanciu implementovanej aktivity. Ako druhý argument sa použije prázdne pole inštancií URI. URI koncového bodu (endpoint) služby je konfigurovaný v app.config súbore v projekte Server (viď kapitolu 5.3.4.1)

5.3.5 Integrácia – klient

Kroky popísané v tejto kapitole sa vykonávajú v prípade, že má byť umožnené použiť danú aktivitu ako argument metódy InitService, teda umožniť serveru prikázať klientovi kontaktovať danú službu.

5.3.5.1 Doplnenie typu aktivity do enumerácie ServiceType

Tento krok zabezpečí, že server bude schopný prikázať klientovi osloviť danú aktivitu.

Do enumerácie ServiceType v súbore ICallbackContract.cs v projekte Server.World v adresári Contracts\Service\ sa pridá na koniec enumerácie hodnota s názvom aktivity.

Príklad:

```
    public enum ServiceType
    {
            Math,
            Hangman //pridaná hodnota
    }
```

5.3.5.2 Konfigurácia ServiceManager-a na klientovi

Tento krok zabezpečí, že klient bude vedieť reagovať na príkaz osloviť danú aktivitu.

Do súboru ServiceManager.cs v projekte Client.Core sa do tela konštruktora triedy ServiceManager pridá inštancia delegáta do tabuľky changers.

Príklad:

```
changers.Add(serviceType, delegate(InitServiceSolicit args)
{
    HangmanConnector.Instance.Connect(args.Token, args.Endpoint);
});
```

Kde serviceType určuje typ implementovanej aktivity (z kapitoly 5.3.5.1) a telo delegáta zabezpečí pripojenie k službe aktivity s pomocou Connectora implementovaného na strane klienta (viď kapitola 5.3.6).

5.3.6 Implementácia na strane klienta

5.3.6.1 Pridanie referencie na službu

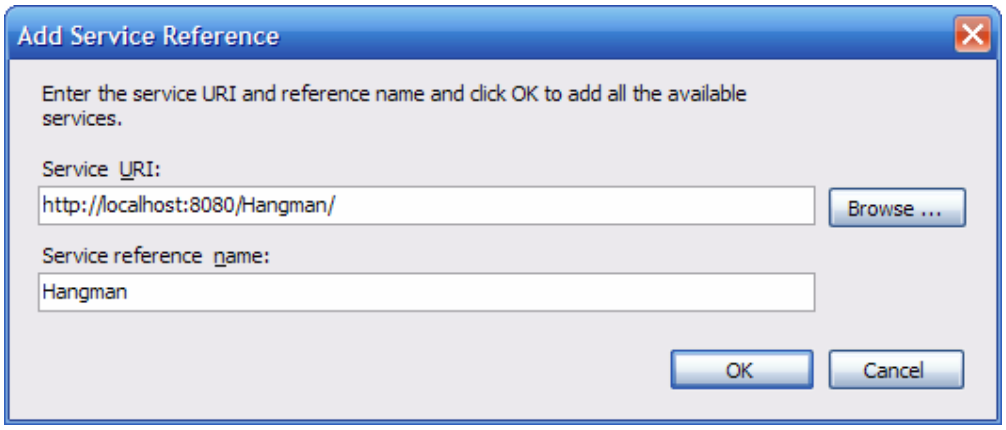
Spustí sa výstup projektu Server.

Vo VS v projekte Client.Core sa pridá referencia na službu (Add Service Reference...)

Do textboxu „Service URI:“ sa zadá Uri endpointu služby implementovanej aktivity (tak ako bolo nakonfigurované v kapitole 5.3.4.1).

Do textboxu „Service reference name:“ sa zadá názov aktivity.

Príklad:



Obrázok 5-2 Pridanie referencie na službu

5.3.6.2 Implementácia Connector triedy

Connector trieda sa implementuje podľa vzoru unikát (singleton). Implementuje callback rozhranie aktivity (viď kapitolu 5.3.3.1.3).

Ako členská premenná sa použije Client trieda, ktorú vygenerovalo VS po pridaní referencie.

Implementuje sa metóda Connect, ktorá inštancuje Client triedu vygenerovanú pri pridaní referencie.

Ako argument InstanceContext do konštruktora Client triedy sa použije `this`.

Príklad:

```
this.c = new Client.Hangman.Hangman.HangmanGameClient(
new System.ServiceModel.InstanceContext(this));
```

Na počítačoch s operačným systémom Windows XP Professional, kde beží služba IIS, je potrebné použiť nasledovný trik pre nastavenie iného portu ako 80 (implicitná hodnota):

```
WSDualHttpBinding b;

b = this.c.Endpoint.Binding as WSDualHttpBinding;
b.ClientBaseAddress = loopbackUri;
```

kde loopbackUri je platné Uri odkazujúce sa na voľný otvorený port na klientovi.

Posledným krokom je zavolanie metódy Open na inštancii Client:

```
this.c.Open();
```

5.3.6.2.1 Implementácia callback metód

Táto kapitola obsahuje popis implementácie callback metód v Connector triede.

V deklarácii triedy:

```
public class HangmanConnector : IHangmanGameCallback
```

Sa kurzor presunie na slovo `IHangmanGameCallback` a stlačí sa ctrl+. (control a bodka). Toto vygeneruje prázdne metódy pre všetky metódy deklarované v callback rozhraní.

Pre každú metódu sa implementuje v súlade s pravidlami popísanými v dokumente „Štandardy programovania v jazyku C#“ udalosť typu EventHandler<EventArgs<T>>, kde typový parameter T je typ, ktorý je buď typom argumentu callback metódy (v prípade jediného argumentu), alebo obaľuje argumenty callback metódy (v prípade viacerých argumentov).

6 Posudzovanie projektov

Kapitola mapuje proces posudzovania na proces vývoja softvéru. Okrem posudkov, ktoré napísali členovia tímu pre druhý projekt, zahŕňa aj posudky napísané druhým tímom na dokumentáciu a produkty v rôznych štádiách vývoja softvéru.

6.1 Posudzovanie ku 1. kontrolnému bodu (19.10.2006)

Posudzovanie ku prvému kontrolnému bodu zahŕňalo napísanie posudku ku projektovej dokumentácii druhého tímu a vyjadreniu ku posudku druhého tímu na našu projektovú dokumentáciu. Podkapitola zahŕňa nasledovný zoznam dokumentov:

- posudok na projektovú dokumentáciu tímu č. 12 (NetRollers)
- posudok na našu projektovú dokumentáciu tímom č. 12
- vyjadrenie k posudku druhého tímu

7 Preberacie protokoly

Kapitola obsahuje preberacie protokoly dokumentácie tímového projektu pre jednotlivé kontrolné body .

7.1 Preberací protokol 1. kontrolného bodu (19.10.2006)

Podkapitola obsahuje preberací protokol zo dňa 19.10.2006.

8 Práca na dokumentácii

Kapitola zhŕňa podiel práce všetkých autorov na projektovej dokumentácii. Pre každý kontrolný bod uvádza tabuľku, v ktorej popisuje kapitoly dokumentácie

8.1 Práca na dokumentácii v 1. kontrolnom bode

Podkapitola zhŕňa prácu členov tímu v 1. kontrolnom bode vo forme tabuľky. [Tabuľka 8-1]

Tabuľka 8-1Práca na dokumentácii v 1. kontrolnom bode

Kapitola	Autori	Rozsah práce [%]
Dokument riadenia projektu		
Ponuka	Marek Tomša	95
	Andrej Frlička	5
Plán projektu	Marek Tomša	100
Úlohy členov tímu	Andrej Frlička	100
Záznamy zo stretnutí	Andrej Frlička	25
	Marek Tomša	25
	Oto Vozár	25
	Richard Veselý	25
Tímové štandardy	Andrej Frlička	50
	Richard Veselý	50
Formátovanie a úprava	Andrej Frlička	100
Dokument výsledkov projektu		
Úvod	Andrej Frlička	100
Opis riešeného problému	Oto Vozár	100
Špecifikácia rámca	Andrej Frlička	50
	Marek Tomša	50
Špecifikácia častí	Andrej Frlička	100
Analýza technológií pre vývoj	Richard Veselý	100
Hrubý návrh	Marek Tomša	100
Záver	Marek Tomša	100
Formátovanie a úprava	Andrej Frlička	100

8.2 Práca na dokumentácii v 2. kontrolnom bode

Podkapitola zhŕňa prácu členov tímu v 2. kontrolnom bode vo forme tabuľky. [Tabuľka 8-2]

Tabuľka 8-2 Práca na dokumentácii v 2. kontrolnom bode

Kapitola	Autori	Rozsah práce [%]
Dokument riadenia projektu		
Plán projektu	Marek Tomša	100
Úlohy členov tímu	Andrej Frlička	100
Záznamy zo stretnutí	Andrej Frlička	25
	Marek Tomša	25
	Oto Vozár	25
	Richard Veselý	25
Tímové štandardy	Marek Tomša	100
Formátovanie a úprava	Andrej Frlička	100
Dokument výsledkov projektu		
Predslov	Oto Vozár	100
Úvod	Andrej Frlička	100
Prototyp		
• server	Marek Tomša	100
• editor máp	Oto Vozár	100
• klient	Richard Veselý	100
• PET v PDA	Andrej Frlička	100
Záver	Andrej Frlička	100
Formátovanie a úprava	Andrej Frlička	100