SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Imagine Cup 2011: Game Design

(Dokumentácia riadenia)

Tím: CPO Entertainment (č. 1)
Vedúci TP: Ing. Michal Tvarožek
Kontakt: icgd2011fiit@live.com
Dátum: 16. novembra 2010

Autori: Bc. Michal Barát
Bc. Anton Benčič
Bc. Peter Svorada
Bc. Márius Šajgalík

Obsah

1 Úvod

V rámci predmetu tímový projekt sme si vybrali prácu na hre s účasťou na súťaži Imagine Cup 2011 v kategórii herný návrh a tento dokument opisuje proces tvorby našej hry. Keďže v tejto súťaži ide o akési riešenie alebo aspoň vzdelávanie či informovanie v oblasti najväčších svetových problémoch, pre tento rok konkrétne miléniových cieľov OSN, tak to kladie určité dodatočné požiadavky hlavne na obsah našej hry, ktoré sa nemusia v klasických projektoch veľmi často zvažovať. Konkrétne ide napríklad o naviazanie témy a obsahu hry na jeden z týchto miléniových cieľov a obmedzenie obsahu pre kategóriu ESRB E(veryone). Keďže ide o súťaž predstavivosti, tak je ďalším kľúčovým prvkom úspešných hier ich inovatívnosť. Kvôli tomuto faktu sme sa aj my sústredili podstatnú časť prvej fázy na analýzu predchádzajúcich ročníkov súťaže a účastníckych projektov v nich, analýzu inovatívnych a populárnych hier v žánroch blízkych našej koncepcii a úvahám o tom, akú stratégiu by sme mali počas projektu viesť tak, aby sme sa úspešne dostali až do finále súťaže.

Druhá kapitola s názvom *Členovia tímu a ich kompetencie* sa venuje zoznámeniu s členmi tímu, kde je uvedená krátka informácia o predchádzajúcom štúdiu, získaných vedomostiach či skúsenostiach, Takisto má v tejto kapitole každý člen uvedené svoje roly v tíme a opis svojich kompetencií.

Tretia kapitola s názvom *Autorstvo jednotlivých častí dokumentácie* poskytuje výpis jednotlivých častí dokumentácie a autorov, ktorí sa na nich podieľali.

Štvrtá kapitola s názvom *Kolaborácia a komunikácia* sa venuje tomu ako spolu v rámci projektu komunikujeme a aké prostriedky používame pri spoločnej práci s dokumentmi a celkovo pri kolaborácii.

Piata kapitola s názvom *Manažment plánovania* pojednáva o tom ako v tíme plánujeme a ponúka pohľad ako na rámcový plán celého projektu, tak aj podrobný plán a stav splnenia úloh pre prvú fázu projektu, ktorú práve dobiehame.

Šiesta kapitola s názvom *Manažment kvality* určuje spôsoby písania zdrojového kódu tak, aby sa zabezpečili jednotné výstupy a tým aj celková kvalita riešenia.

2 Členovia tímu a ich kompetencie

2.1 Michal Barát

Bakalárske štúdium absolvoval na fakulte informatiky a informačných technológií na Slovenskej technickej univerzite v Bratislave v študijnom programe Informatika. Vo svojej záverečnej práci sa venoval problematike spracovania obrazu, konkrétne zmene rozmerov obrázku s prihliadnutím na jeho obsah. Absolvoval druhý stupeň štúdia na základnej umeleckej škole v odbore Výtvarná výchova. Vo voľnom čase sa venuje tvorbe a editácií obrazu, grafiky a fotografií.

Roly:

- Manažér rizík
- *Grafik*

Náplňou práce Michala je z pohľadu manažmentu identifikácia a riadenie rizík. Pod identifikáciou rizík sa rozumie riadenie množiny možných problémov, ktoré by mohli počas projektu nastať. Ďalej je jeho úlohou tieto možné komplikácie analyzovať z hľadiska pravdepodobnosti ich výskytu a potenciálnych dopadov. O týchto rizikách a zmenách v súvislosti s nimi bude tím pravidelne informovať.

Michal takisto zastáva rolu grafika, v ktorej je jeho úlohou pripravovať pre hru multimediálny obsah, teda jednotlivé modely, animácie, obrázky a používateľské rozhranie. Okrem spomenutých vecí má takisto na starosti prezentáciu tímu a projektu z pohľadu marketingu.

2.2 Anton Benčič

Bakalárske štúdium absolvoval na FIIT STU v Bratislave v študijnom programe Informatika a na FM UK v Bratislave v študijnom programe Manažment. Na FIIT STU vypracoval svoju bakalársku prácu v rámci súťaže Imagine Cup, kde sa dostal s projektom na zdieľanie vecí do svetového finále. Na FM UK vypracoval bakalársku prácu na tému Moderné prístupy v manažmente softvérových projektov, pričom sa v rámci nej zameral na agilnú metodiku vývoja podľa čŕt (FDD - Feature Driven Development). Popri škole pracoval na vývoji Geografického IS na vizuálnu reprezentáciu existujúcich regionálnych dát a získal skúsenosti s vývojom servisne orientovaných aplikácií ako aj klientskych častí pre tučných, tenkých aj mobilných klientov.

Roly:

- Vedúci tímu
- Manažér podporných prostriedkov
- *Grafik*

- *Programátor*

Náplňou práce Antona z pohľadu manažmentu je zabezpečenie celkovej orchestrácie tímu a všetkých záležitostí, ktoré sú nevyhnutné pre plynulú prácu každého člena. To zahŕňa napríklad zabezpečenie podporných a vývojových prostriedkov, informovanie o spôsobe ich používania a riešenie problémov s nimi spojených. Ďalej je jeho úlohou sledovať, či a ako si všetci členovia tímu plnia svoje úlohy, ktoré im vyplývajú z ich manažérskych aj nemanažérskych rol a navrhovať spôsoby práce, ktoré môžu prispieť k riešeniu vzniknutých problémov a k zvýšeniu celkovej efektívnosti práce.

Anton takisto zastáva v tíme pozíciu programátora, v ktorej je jeho úlohou pracovať na implementácii jednotlivých dohodnutých častí výsledného produktu. V role grafika je Antonovou úlohou kolaborácia s Michalom hlavne na úrovni diskusie potrieb a postupov tvorby multimediálneho obsahu, pomoci pri jeho práci a pri riešení problémov, keďže už s týmito nástrojmi pracoval a vie čo je potrebné pre ich hladkú integráciu s ostatnými časťami.

2.3 Peter Svorada

Bakalárske štúdium absolvoval na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave v študijnom programe Informatika. Bakalársku prácu vypracoval na tému Podpora správy výpočtovej techniky, so zameraním na rozšírenie existujúce systému o správu používateľských práv a účtov. Na pokročilej úrovni ovláda programovacie jazyky C++ a Java. Taktiež ovláda jazyky HTML, CSS, JavaScript, Visual FoxPro, Prolog a Lisp. Má skúsenosti s prostredím MS Visual Studio 2008 a Eclipse. Poznatky pre prácu v tíme nadobudol v spoločnosti zabezpečujúcej technickú podporu pre sieť maloobchodných predajní.

Roly:

- Manažér plánovania
- *Programátor*

Náplňou práce Petra z pohľadu manažmentu je tvorba rámcových a krátkodobých plánov a sledovanie ich plnenia. Rámcové plánovanie vychádza hlavne z dátumov, ku ktorým je potrebné dodať jednotlivé výstupy a krátkodobé plánovanie vychádza z jednotlivých stretnutí, kde sa vždy zistí aktuálny stav otvorených úloh a prípadne sa pridelia nové. Peter bude následne medzi jednotlivými stretnutiami sledovať stav splnenia týchto úloh, ktorý bude reportovaný práve tými, ktorí na nich budú pracovať.

Peter takisto zastáva v tíme pozíciu programátora, v ktorej je jeho úlohou pracovať na implementácii jednotlivých dohodnutých častí výsledného produktu.

2.4 Márius Šajgalík

Bakalárske štúdium absolvoval na FIIT STU v Bratislave v študijnom programe Informatika. V rámci bakalárskej práce pracoval v tíme na projekte Present, s ktorým nadobudol skúsenosti aj v celosvetovom finále súťaže Imagine Cup. Úspešne reprezentoval fakultu na viacerých

medzinárodných programátorských súťažiach ako IPSC, Google Code Jam a Challenge 24 na finále v Budapešti. Každý rok reprezentuje fakultu na regionálnom kole programátorskej súťaže ACM ICPC a podieľa sa na príprave súťaže ProFIIT. Aktívne sa venuje súťaži TopCoder v kategórii algoritmov. Má praktické skúsenosti napr. s jazykom symbolických inštrukcií, C, C++, sieťovým programovaním, DirectX, C#, XAML, Silverlight, Java, JavaScript, či PHP.

Roly:

- Manažér vývoja
- Manažér kvality
- *Programátor*

Náplňou práce Máriusa z pohľadu manažmentu je riadiť vývoj a jeho kvalitu. Čo sa týka riadenia vývoja, tak je jeho úlohou konzultovanie a návrh rozloženia implementačných úloh do rámcových a krátkodobých plánov s Peťom a následný návrh rozdelenia týchto úloh medzi jednotlivých členov tímu. Okrem samotnej implementácie je jeho úlohou držanie dohľadu nad architektúrou riešenia a riadenie zmien, testov, chýb a celkovej kvality.

Márius takisto zastáva v tíme pozíciu programátora, v ktorej je jeho úlohou pracovať na implementácii jednotlivých dohodnutých častí výsledného produktu.

3 Autorstvo jednotlivých častí dokumentácie

V tabuľke 1 sa nachádza výpis autorstva jednotlivých častí dokumentácie.

Časť dokumentácie	Kapitola	Autor
Dokumentácia riadenia	Úvod	Anton Benčič
Dokumentácia riadenia	Členovia tímu a ich kompetencie[1]	Anton Benčič
Dokumentácia riadenia	Autorstvo jednotlivých častí dokumentácie	Anton Benčič
Dokumentácia riadenia	Kolaborácia a komunikácia	Márius Šajgalík
Dokumentácia riadenia	Manažment plánovania	Peter Svorada
Dokumentácia riadenia	Manažment kvality	Márius Šajgalík
Dokumentácia projektu	Úvod	Anton Benčič
Dokumentácia projektu	Analýza súťaže, hier a ich prvkov • O súťaži Imagine Cup • Imagine Cup 2010 • Hlavolamy v RPG hrách a adventúrach	Michal Barát
Dokumentácia projektu	Analýza súťaže, hier a ich prvkov • Inventár • Pohľad na postavu • Pohľad na interiéry • Zlepšovanie atribútov • Súboje • Cestovanie v čase	Peter Svorada
Dokumentácia projektu	Koncepcia RPG ôsmych svetov	Anton Benčič
Dokumentácia projektu	Špecifikácia hry a editoru • Hra	Anton Benčič
Dokumentácia projektu	Špecifikácia hry a editoru • Herný editor	Márius Šajgalík
Dokumentácia projektu	Hrubý návrh enginu	Márius Šajgalík

Tabuľka 1 Autorstvo jednotlivých častí dokumentácie

[1] Anton Benčič spisoval kompetencie a úlohy jednotlivých členov tímu. Časti, v ktorých je skrátená verzia skúseností a vedomostí je prebratá z ponuky tímu.

4 Kolaborácia a komunikácia

Hlavnými prostriedkami, ktoré podporujú kolaboráciu a komunikáciu v našom tíme, sú:

- Windows Live skupina
- Microsoft Visual Studio 2010 Team Foundation Server (TFS)

Windows Live skupina

Od začiatku projektu a teda aj fungovania tímu bola hlavným kolaboračným prostriedkom Windows Live skupina. Windows Live skupina ako jedna zo služieb Windows Live, je prepojená s viacerými ostatnými službami, ktoré tím využíva:

- **úložný priestor Sky drive** – poskytuje 25 GB úložného priestoru, kde sa momentálne nachádzajú všetky dokumenty vytvorené v rámci projektu.
- **Office Live** – vďaka tejto službe je umožnené vytvorenie a simultánna editácia dokumentov Microsoft Office viacerým členom tímu. Je to relatívne nová služba, ktorá má značné výhody oproti ostatným riešeniam. Podporuje nielen online editovanie dokumentov, ale prakticky v sebe zahŕňa aj manažment verziovania súborov. Je možné vidieť, ktoré zmeny v dokumente kto vykonal a je možné vytvárať viaceré verzie súboru, takže možno potom obnoviť existujúce predchádzajúce verzie. Online editácia je možná prostredníctvom webovej stránky, kde je dostupná v podstate viac než základná funkcionalita, alebo je možné otvoriť dokument priamo v klientskej aplikácii. Podporované sú momentálne Microsoft Word, Microsoft Excel, Microsoft PowerPoint a Microsoft OneNote. Keď sa dokument nachádza v zozname posledne otvorených, alebo je pripnutý v zozname obľúbených dokumentov, je možná aj jeho offline editácia. Takto je možná práca aj mimo internetu a zmeny je možné jednoducho pri najbližšej príležitosti zosynchronizovať s online verziou.
- **Windows Live Messenger** – toto je samostatná aplikácia v rámci aplikačného balíka Windows Live, pričom existuje aj jeho zjednodušená webová verzia. Toto je jeden z primárnych komunikačných nástrojov v tíme a komunikácia členov tímu je združená na základe príslušnosti k Windows Live skupine.
- **Windows Live Calendar** – ďalšia služba Windows Live, vďaka ktorej je možné mať prehľad o udalostiach týkajúcich sa všetkých členov tímu.

Windows Live skupina automaticky poskytuje mailovú adresu pre skupinovú komunikáciu, pričom je zvonku uzavretá, takže nie je možné na ňu posielať maily niekomu, kto do skupiny nepatrí. Preto sme zriadili tímový Windows Live účet, ktorého mailovú adresu používame ako verejný mail.

Microsoft Visual Studio 2010 Team Foundation Server (TFS)

Keďže hlavné vývojové prostredie v tíme je Visual Studio 2010, rozhodli sme sa použiť serverové rozšírenie pre tímové projekty v podobe Team Foundation Server. Jeho podpora je priamo integrovaná v rámci Visual Studia a poskytuje viacero podporných prostriedkov:

- Podpora plánovania projektu
- Podpora vytvárania reportov
- Podpora projektových dokumentov
- Manažment verzií súborov (tzv. source control)
- Podpora automatizovaného kompilovania projektu (automated build)
- Podpora automatizovaného testovania
- Podpora manažmentu chýb (tzv. bug tracking)

Pred tým, než sme mali k dispozícii server pre náš tímový projekt a teda aj nasadením TFS, plánovali sme využiť aj Project Server. Ten mal slúžiť najmä na tímové plánovanie. Po nasadení TFS, a aj kvôli viacerým technickým problémom, sme sa nakoniec rozhodli použiť plánovanie integrované v TFS. Toto riešenie predstavuje prakticky plnohodnotnú alternatívu, ktorú sme identifikovali ako vhodnejšie riešenie. TFS, okrem webového riešenia pre projektové plánovanie, navyše podporuje vytváranie plánov v Microsoft Project a zároveň dokáže vyexportovať rozličné časti plánov ako plán pre Microsoft Project.

V súčasnosti taktiež uvažujeme o prechode na TFS riešenie pre tímové dokumenty, aby mohla byť tímová práca sústredená na jednom mieste.

5 Manažment plánovania

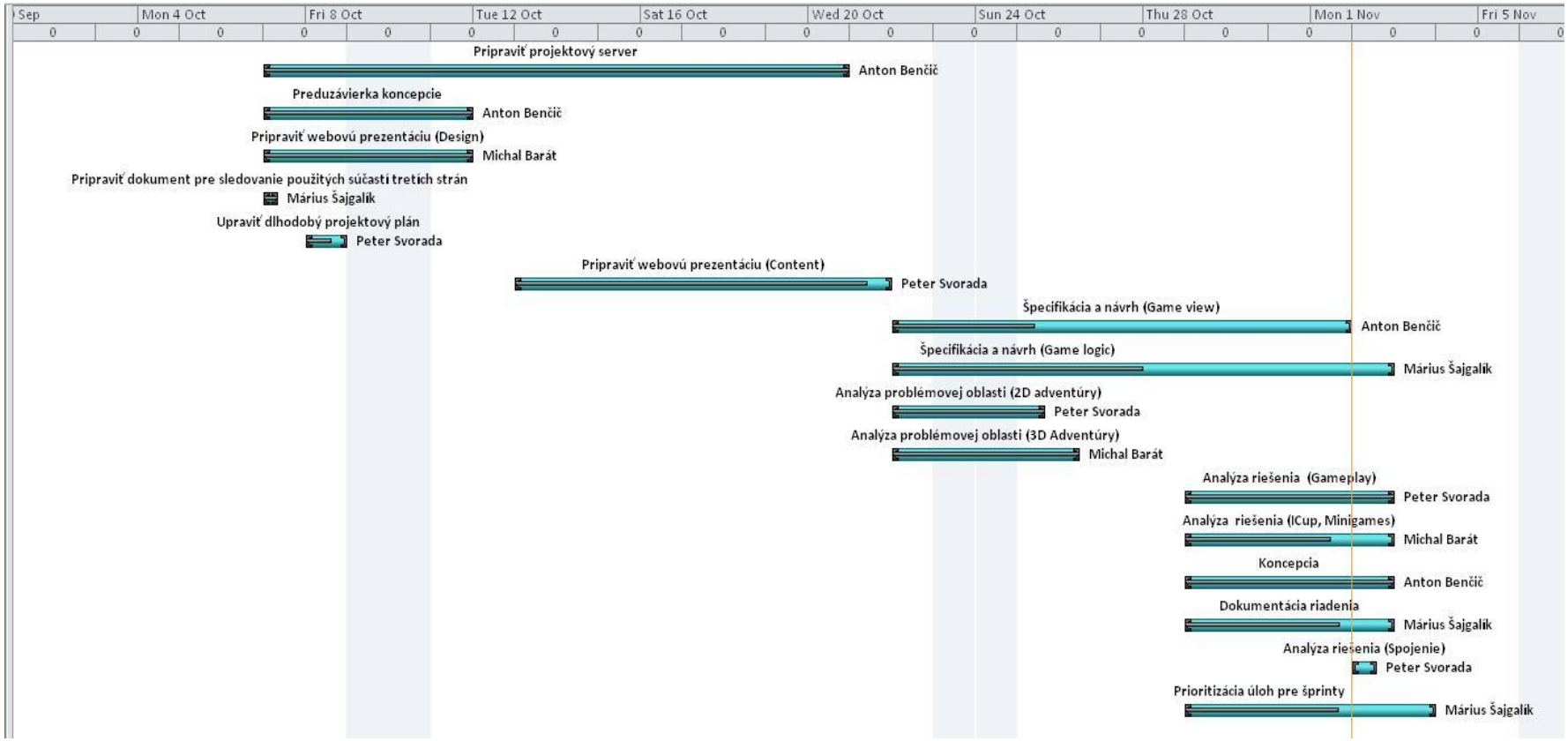
V rámci plánovania bol najskôr vytvorený dlhodobý plán vychádzajúci z dátumov odovzdania, respektíve prezentácií jednotlivých výstupov potrebných pre predmety Tímový projekt 1 a 2, ako aj súťaže Imagine Cup a TP Cup, Tieto dátumy je možné vidieť v tabuľke 2.

2.11.2010	[Tímový projekt] Odovzdanie analýzy, špecifikácie a návrhu
24.11.2010	[TP CUP] Prihláška
26.11.2010	[Prezentácia] Onto party
14.12.2010	[Tímový projekt] Odovzdanie prototypu vybraných častí systému spolu s dokumentáciou/odovzdanie dokumentácie prvých piatich šprintov spolu s popisom prototypu
15.12.2010	[Tímový projekt] Používateľská prezentácia prototypu
23.2.2011	[TP CUP] Odovzdanie priebežnej správy
7.3.2011	[Imagine Cup] Prvé kolo
7.4.2011	[Imagine Cup] Druhé kolo
11.4.2011	[Tímový projekt] Oovzdanie produktu a dokumentácie k produktu/odovzdanie produktu a documentácie k šprintom 6 až 10
10.5.2011	[Tímový projekt] Odovzdanie celkového výsledku projektu

Tabuľka 2 Dlhodobý rámcový plán

Tento plán bol postupne dopĺňaný o konkrétne, krátkodobé úlohy, ktoré boli aktualizované na týždenných základoch, zväčša na tímových stretnutiach, kde sa overil pokrok v jednotlivých úlohách, dokončené sa uzavreli a otvorili sa nové. Úlohy sa vyberali a vyberajú tak, aby speli k splneniu požiadaviek dlhodobého plánu. To znamená, že posledné týždne boli venované dokončovaniu dokumentácie pre prvé odovzdanie v predmete Tímový projekt, zatiaľ čo nasledujúce týždne budú venované programovaniu prvého prototypu, ktorý bude možné prezentovať na Onto Party. Prehľad všetkých týchto úloh je možné vidieť na obrázku 1.

Práve sa nachádzame v jednom z bodov, kedy sa dokončujú stanovené úlohy, aby sa na základe ich výstupov mohli vytvoriť ďalšie. Ide najmä o špecifikáciu požiadaviek a návrh riešenia, na základe ktorých budú navrhnuté šprinty, ktoré sa budú následne bližšie rozplánovávať.



Obrázok 1 Prehľad úloh

6 Manažment kvality

Názvoslovie

camelCase – slovo s prvým písmenom malým a prvým písmenom každej ďalšej slovnej časti veľkým, napr. typAuta.

PascalCase – slovo s prvým písmenom veľkým a prvým písmenom každej ďalšej slovnej časti veľkým, napr. TypAuta.

Konvencie pri nazývaní

Identifikátor	Public	Protected	Internal	Private	Poznámky
Zdrojový súbor	P	x	x	x	Rovnaký názov ako názov triedy v ňom.
`Namespace`	P	x	x	x	
`Class` / `Struct`	P	P	P	P	
`Interface`	P	P	P	P	Názov vždy začína s veľkým `I`.
`Method`	P	P	P	P	Sloveso, alebo kombinácia sloveso - objekt.
`Property`	P	P	P	P	Nikdy nezačínať s `Get` alebo `Set`. Názov by mal reprezentovať entitu, ktorá sa vracia.
`Field`	P	P	P	_c	Používať len na úrovni `Private` a pristupovať k nim cez `Property`.
`Constant`	P	P	P	_c	
`Enum`	P	P	P	P	Aj polia v `Enum` sú štýlom `PascalCase`.
`Delegate`	P	P	P	P	
`Event`	P	P	P	P	
`Inline` premenná	x	x	x	c	Vyhýbať sa jednoznakovým názvom okrem použitia v riadení cyklu.
Parameter	x	x	x	c	

Vysvetlivky:

„c“ = názov štýlom `camelCase`
„P“ = názov štýlom `PascalCase`
„_“ = názov začína znakom _
„x“ = nepoužiteľné

Obrázok 1 Prehľad úloh

Manažment kvality

Všeobecné pokyny

- Pri nazývaní vždy používať štýl `camelCase`, alebo `PascalCase`.
- Všetko písať a nazývať po anglicky (vrátane komentárov), napr. `IsDead`, nie `JeMrtvy`.
- Vždy používať zmysluplné názvy opisujúce entitu, nie typ a rozmer. Nepoužívať jednoznakové názvy, jedine pre riadiacu premennú v cykle. Nepoužívať maďarskú notáciu, napr. `strName` alebo `iCount`. Vyhýbať sa redundantným a bezvýznamovým predponám a príponám, napr.:

```
// Bad!
public enum ColorsEnum {…}
public class CVehicle {…}
public struct RectangleStruct {…}
```

- Vyhýbať sa používaniu skratiek pokiaľ nie je celý názov pridlhý. Vyhýbať sa použitiu skratiek dlhších ako 5 znakov. Všetky skratky musia byť dobre známe a schválené manažérom kvality.
- Používať veľké písmená na dvojpísmenovú značku a štýl `PascalCase` pre dlhšie značky.
- Nepoužívať kľúčové slová jazyka C# ako názvy. Vyhýbať sa konfliktom s existujúcim .NET Framework typom alebo `namespace`.
- Názov rodičovskej triedy sa nemôže vyskytovať v názve `Property`, napr. `Customer.Name`, nie `Customer.CustomerName`.
- Snažiť sa dávať príponu „`Can`“, „`Is`“, alebo „`Has`“ k názvu premennej, alebo `Property`.
- Tam, kde sa hodí, dávať výpočtový kvalifikátor ako príponu k názvu premennej ako `Average`, `Count`, `Sum`, `Min` a `Max`.

Štýl písania kódu

- Nikdy nedávať viac ako 1 `namespace` do 1 súboru.
- Vyhýbať sa použitiu viacerých tried v 1 súbore.
- Vždy dávať zložené zátvorky do nového riadku. Príslušné zátvorky musia byť na rovnakej úrovni odsadenia.
- Vždy použiť zložené zátvorky, ak je to voliteľné. Vždy používať zložené zátvorky pri podmienke `if` a pri všetkých cykloch, pokiaľ nasleduje blok `else`, alebo je príkaz v tele pridlhý (dlhší ako šírka obrazovky) – ten treba rozdeliť na viac riadkov.
- Používať odsadenie s veľkosťou 4 medzery. Kód vrátane komentára musí byť vždy na rovnakej úrovni odsadenia. Pre správne formátovanie vždy použiť funkciu automatického formátovania (vo VS2010 Ctrl+K,F pre naformátovanie aktuálneho riadku, alebo výberu a Ctrl+K,D pre naformátovanie celého dokumentu).
- Používať `//` alebo `///`, nie `/* ... */` a zbytočne neskrášľovať (flowerbox). Komentáre treba písať v angličtine a gramaticky správne.
- Písať komentár všade, kde treba. Dobrý čitateľný kód však vyžaduje oveľa menej komentára. Ak majú všetky premenné a metódy zmysluplné názvy, kód je oveľa čitateľnejší a nepotrebuje veľa

Obrázok 1 Prehľad úloh

komentárov. Netreba písať komentár, ak je kód ľahko pochopiteľný aj bez neho. Nevýhoda množstva komentáru spočíva v tom, že ak sa zmení kód a zabudne sa zmeniť komentár, spôsobí to viac zmätkov.

- Menej riadkov komentára urobí kód elegantnejším. Ak však kód nie je čitateľný a má málo komentárov, to je horšie. Ak kód používa nejakú komplexnú alebo neobvyklú logiku, treba to veľmi dobre zdokumentovať s dostatočným množstvom komentáru.
- Ak sa nejaká číselná hodnota inicializuje špeciálnym číslom rôznym od 0, -1, a pod., treba uviesť dôvod pre výber tejto hodnoty.
- Snažiť sa pri komentovaní využívať značky pre zoznam úloh, napr.:

```
// TODO: Place database code here
// UNDONE: Removed due to errors
// HACK: Temporary fix until able to refactor
```

- Vždy vytvárať bloky komentára (`///`) pre deklarácie typu `public`, `protected` a `internal`.
- Vždy uvádzať komentáre s popisom v `<summary>`. Tam, kde je to možné, uviesť aj `<param>`, `<return>` a `<exception>`. Podobne tam, kde je to možné, treba využiť `<see cref=””/>` a `<seeAlso cref=””/>`.
- Deklarovať každú premennú nezávisle v samostatnom riadku. Snažiť sa inicializovať premenné tam, kde sú deklarované, resp. čo najbližšie k miestu deklarácie.
- Zoskupovať výrazy „`using`“ vzťahujúce sa na použitie `namespace` a umiestniť ich na začiatku súboru. Zoskupovať .NET názvoslovia nad vlastnými.
- Zoskupovať internú implementáciu triedy podľa typu v nasledovnom poradí:
    1. Členské premenné
    2. Konštruktory, deštruktory a finalizéry
    3. Vnorené triedy, štruktúry a konštrukcie `Enum`
    4. Konštrukcie `Property`
    5. Metódy
- Zoskupovať v rámci typových skupín podľa dostupnosti:
    1. `Public`
    2. `Protected`
    3. `Internal`
    4. `Private`
- Používať maximálne jednu medzeru na oddelenie logických skupín v kóde. Vnútri triedy používať práve jednu medzeru na oddelenie metód. Vyčleniť implementáciu rozhraní použitím výrazu `#region`.
- Deklarovať členské premenné iba na úrovni `private`. Pre prístup k nim použiť `Property` na úrovni `public`, `protected`, alebo `internal`.
- Vždy vyberať najjednoduchší možný údajový typ. Ak stačí mať konštantu, nevytvárať premennú, ak stačí mať statické pole, nepoužívať zbytočne triedu `Dictionary`.

Obrázok 1 Prehľad úloh

Manažment kvality

- Vždy používať vstavané kľúčové slová pre C# údajové typy, nie .NET common type system (CTS) objekty, napr.:

  `short` a nie `System.Int16`
  `int` a nie `System.Int32`
  `long` a nie `System.Int64`
  `string` a nie `System.String`

- Snažiť sa používať `int` pre akékoľvek celočíselné hodnoty, ktoré sa zmestia do rozsahu typu `int` – aj pre nezáporné čísla. Snažiť sa vyhnúť použitiu `sbyte`, `short`, `uint` a `ulong` pokiaľ sa nejedná o spoluprácu s natívnymi knižnicami.
- Tam, kde je to možné, pre zabezpečenie presnosti používať typ `double` pre desatinné číselné hodnoty, nikdy nie typ `float`. Používať typ `decimal`, ak sa pri výpočte vyžaduje zaokrúhľovanie čísla na presný počet desatinných miest (obvykle pri rátaní peňazí).
- Vyhýbať sa uvádzaniu konkrétnych číselných hodnôt priamo v kóde výpočtu. Namiesto toho používať `constant`, `enum`, konfiguračný súbor, registre, alebo iné zdroje údajov. Používať `enum` vždy, keď je to možné, spravidla na diskrétne hodnoty.
- Vyhýbať sa použitiu konkrétnych reťazcov priamo v kóde. Namiesto toho využívať tzv. „Resource“ súbory, konštanty, konfiguračné súbory, registre, alebo iné zdroje údajov.
- Používať konštanty len vtedy, ak je isté, že sa určite nebude meniť ich hodnota. Inak používať konfiguračný súbor.
- Snažiť sa používať predponu „`@`“ pre reťazce namiesto tzv. „escaped“ reťazcov.
- Snažiť sa používať `String.Format()` alebo `StringBuilder` namiesto spájania reťazcov. Nikdy nespájať reťazce vo vnútri cyklu.
- Pri porovnávaní reťazcov nikdy neporovnávať so `String.Empty` alebo `“”` na zistenie, či je reťazec prázdny. Namiesto toho porovnávať spôsobom `String.Length == 0`, `String.IsNullOrEmpty()`, alebo `String.IsNullOrWhiteSpace()`.
- Vyhýbať sa skrytému vytváraniu dočasných reťazcov, najmä v cykle. Používať String.Compare() pre porovnávanie ignorujúce veľkosť písmen. ToLower() vytvára dočasný reťazec. Napr.:

```
// Bad!
int id = -1;
string name = “lance hunt”;

for (int i = 0; i < customerList.Count; i++)
{
      if (customerList[i].Name.ToLower() == name)
      {
            id = customerList[i].ID;
      }
}
```

Obrázok 1 Prehľad úloh


```
// Good!
int id = -1;
string name = “lance hunt”;

for (int i = 0; i < customerList.Count; i++)
{
      // The “ignoreCase = true” argument performs a
      // case-insensitive compare without new allocation.
      if (String.Compare(customerList[i].Name, name, true) == 0)
      {
            id = customerList[i].ID;
      }
}
```

- Používať ternárny operátor len pri triviálnych podmienkach, napr.:

```
int result = isValid ? 9 : 4;
```

- Vyhýbať sa porovnávaniu Boolovských výrazov s konštantami `true` a `false`, napr.:

```
// Bad!
if (isValid == true) {...}

// Good!
if (isValid) {...}
```

- Vyhýbať sa priradzovaniu v rámci podmienky, napr.:

```
if ((i = 2) == 2) {...}
```

- Snažiť sa rozdeliť zložité podmienky do viacerých Boolovských premenných, napr.:

```
// Bad!
if (((value > _highScore) && (value != _highScore)) && (value <
_maxScore))
{...}

// Good!
isHighScore = (value >= _highScore);
isTiedHigh = (value == _highScore);
isValid = (value < _maxValue);
if ((isHighScore && !isTiedHigh) && isValid)
{...}
```

Obrázok 1 Prehľad úloh

Manažment kvality

- Pomocou `catch` zachytávať len tie výnimky, ktoré sú v tomto bloku ošetrené, vždy len konkrétne, nikdy nie všeobecné. Netreba sa snažiť zachytiť všeobecné výnimky, treba ich ponechať, aby aplikácia havarovala. Takto sa dopomôže nájsť väčšinu chýb počas vývojového cyklu. Potom je možné spraviť zachytávanie výnimiek celej aplikácie pre zachytenie všetkých všeobecných výnimiek spolu s ich logovaním.
- Nikdy nenechávať prázdny `catch` blok.
- V prípade výnimky treba podať priateľskú správu používateľovi, no zalogovať chybu, ktorá nastala so všetkými možnými informáciami o chybe, vrátane času, kedy nastala, názov metódy a triedy, atď.
- Nepoužívať veľmi dlhé `try/catch` bloky a nevnárať ich do seba.
- Nesnažiť sa zachytávať výnimky vo všetkých metódach, nepoužívať ich na kontrolu behu programu. Použiť ich iba vtedy, ak je tam možnosť, že nastane určitá konkrétna výnimka a nie je to možné ošetriť žiadnym iným spôsobom. Napr. pri vkladaní záznamu do databázy, nie je dobré sa ho pokúsiť najprv vložiť a ak nastane výnimka, predpokladať, že už tam existuje. Vždy treba explicitne ošetrovať chyby namiesto čakania na výnimku. Na druhej strane, treba sa snažiť zachytávať výnimky pri komunikácii s externými systémami ako napr. sieť, hardvérové zariadenia a pod. Takéto systémy môžu byť kedykoľvek príčinou zlyhania a ošetrovanie chýb zvyčajne nie je dostatočne spoľahlivé. V takýchto prípadoch treba zachytiť výnimku a snažiť sa pozviechať sa z chyby, ktorá nastala.
- Vždy volať `Close()` a `Dispose()` v triedach, ktoré to poskytujú.
- Pred vyvolaním vždy skontrolovať, či `event` a `delegate` inštancie nie sú rovné `null`.
- Metóda na ošetrenie udalosti (tzv. „`event` handler“) by nemala obsahovať kód na vykonanie požadovanej akcie. Mala by sa volať metóda, ktorá danú akciu vykoná.
- Vždy používať kľúčové slovo `lock` namiesto typu `Monitor`. Zamykať len objekty na úrovni `private`. Vyhýbať sa zamykaniu objektu `this`.
- Názov metódy by mal vyjadrovať, čo metóda robí. Metóda by mala robiť iba jednu robotu. Nekombinovať viaceré roboty v jednej metóde aj keď sú len malé.
- Vždy ošetriť neočakávané hodnoty. Napr. ak má parameter dve možné hodnoty, nikdy nemožno predpokladať, že ak nemá parameter jednu hodnotu, môže mať jedine druhú.
- Nikdy neuvádzať absolútnu cestu, alebo konkrétne označenie zväzku. Cestu treba zistiť v kóde, alebo použiť absolútnu cestu. Nikdy nepredpokladať, že kód pobeží na zväzku „C:“. Niektorí používatelia ho môžu spúšťať zo siete, alebo z inak označeného zväzku.
- Ak sa nenájde konfiguračný súbor, aplikácia musí byť schopná vytvoriť si ho s bežnými hodnotami. Ak sa nájde nesprávna hodnota v konfiguračnom súbore, aplikácia musí vyhlásiť chybu, alebo podať správu a mala by používateľovi povedať, aké sú korektné hodnoty.
- Chybové správy by mali pomôcť používateľovi vyriešiť problém. Treba podať špecifickú správu a taktiež povedať, aké kroky by mal používateľ vykonať pre vyriešenie problému.
- Používateľovi ukazovať krátke a priateľské správy, no logovať aktuálnu chybu so všetkými možnými informáciami. Dosť to pomôže diagnostikovať problém.

Obrázok 1 Prehľad úloh

Manažment kvality

- Vyhýbať sa veľmi veľkým súborom. Ak má jeden súbor viac ako 1000 riadkov kódu, je to dobrý kandidát na refaktorovanie. Treba ho logicky rozdeliť na dve a viac tried.
- Vyhýbať sa veľmi dlhým metódam. Metóda by mala mať obvykle 1~25 riadkov kódu. Ak má metóda viac ako 25 riadkov, treba zvažovať refaktorovanie na samostatné metódy.
- Počet parametrov metódy by nemal byť väčší ako 5. Inak treba zvážiť vytvorenie triedy, alebo štruktúry, ktorá bude agregovať viacero pôvodných parametrov, najmä ak ich spoločný výskyt je opakovaný, alebo častý.
- Ak metóda vracia nejakú kolekciu prvkov, treba vracať prázdnu kolekciu namiesto `null`, ak neexistujú žiadne údaje, ktoré by sme vrátili.
- Ak sa otvára nejaké databázové spojenie, sieťové spojenie, súbor a pod., vždy to treba zatvoriť v bloku `finally`. Toto zaistí, že aj keď nastane výnimka po otvorení spojenia, bude bezpečne zatvorené v bloku `finally`.

7 Prílohy

Príloha A: Ponuka

Príloha B: Zápisy z formálnych stretnutí

Príloha C: Šablóna preberacieho protokolu

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Imagine Cup 2011: Game Design

Príloha A: Ponuka

(Dokumentácia riadenia)

Tím: CPO Entertainment (č. 1)
Vedúci TP: Ing. Michal Tvarožek
Kontakt: icgd2011fiit@live.com
Dátum: 16. novembra 2010

Autori: Bc. Michal Barát
Bc. Anton Benčič
Bc. Peter Svorada
Bc. Márius Šajgalík

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Imagine Cup 2011: Game Design

Príloha B: Zápisy z formálnych stretnutí
(Dokumentácia riadenia)

Tím: CPO Entertainment (č. 1)
Vedúci TP: Ing. Michal Tvarožek
Kontakt: icgd2011fiit@live.com
Dátum: 16. novembra 2010

Autori: Bc. Michal Barát
Bc. Anton Benčič
Bc. Peter Svorada
Bc. Márius Šajgalík

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Imagine Cup 2011: Game Design

Príloha C: Šablóna preberacieho protokolu
(Dokumentácia riadenia)

Tím: CPO Entertainment (č. 1)
Vedúci TP: Ing. Michal Tvarožek
Kontakt: icgd2011fiit@live.com
Dátum: 16. novembra 2010

Autori: Bc. Michal Barát
Bc. Anton Benčič
Bc. Peter Svorada
Bc. Márius Šajgalík