Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Podpora vývoje webové hry pro více hráčů

Plzeň, 2012 Bc. Petr Vogl

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 28. června 2012

Bc. Petr Vogl

Abstract

Master's thesis deals with the project support for multiplayer webgame development. The purpose of this thesis is to design a development process and determine a suitable supporting tools for developers and players. This thesis reviews and analyses the principles of the web-based and educational games and methodologies for managing projects. Using the Space Traffic webgame – the real project, this paper evaluates the success of the chosen development process and supporting tools in the academic environment. Another purpose of this thesis is to create sufficient support for the ecosystem of the said project. At the end of this paper are presented the author's experience with project management and supporting tools.

Obsah

Seznam obrázků

Poděkování

Na tomto místě bych rád poděkoval rodině a přátelům za podporu při studiu i psaní této diplomové práce. Speciální poděkování bych chtěl věnovat jak vedoucímu mé práce Doc. Ing. Přemyslu Bradovi, MSc., Ph.D. za cenné rady a připomínky, tak celému týmu projektu Space Traffic za jeho skvělou spolupráci.

1 Úvod

Tato diplomová práce se věnuje tématu podpora vývoje webové hry. Bere si za cíl sloužit jako pomůcka studentům při řízení softwarových projektů v průběhu zpracování jejich diplomových, bakalářských a semestrálních prací. Má čtenáře seznámit s postupy při řízení softwarových projektů, sdělit, jak a podle čeho vybrat vhodnou vývojovou metodiku, pomoci určit vhodnou sadu podpůrných nástrojů jak pro řízení, tak pro vývojáře i uživatele a neopomene přiblížit ani témata jako jsou vedení lidí či jejich motivace, která jsou při řízení projektů neméně důležitá.

Semestrální a jiné práce, jejichž náplní je vývoj softwarového produktu, zpracovávají studenti ve všech ročnících studia. Počínaje prvním ročníkem se student setkává se softwarovými projekty, z nichž mnohé sám řídí. Tak je tomu i v případě této diplomové práce, jejímž předmětem zájmu se stal reálný projekt vývoje webové hry Space Traffic. Všechny tyto práce resp. projekty by měly v přiděleném čase dosáhnout stanovených cílů, měly by dodržovat alespoň základní vžité a obecně přijaté postupy řízení[1].

Přínos této diplomové práce spočívá zejména v aplikaci poznatků získaných v průběhu studia na projekt vývoje webové hry, který se díky vynaloženému úsilí přiblížil ke svým cílům. Téma této práce – vytvoření dostatečné podpory pro ekosystém projektu je klíčovou činností prováděnou nejen při zahájení projektu, ale během celého jeho životního cyklu. Cílem této činnosti, a tedy i této diplomové práce, je zajištění zdrojů a technického zázemí projektu, vytvoření dostatečné podpory pro komunitu projektu, usnadnit vývoj produktu a napomoci dosáhnout kvalitního výsledku. Podle toho, jak dobře nebo špatně je prováděna, se odvíjí také zdraví projektu.

Aby čtenář mohl snadněji zjistit směr, jakým se bude zbytek práce ubírat, a pochopit pozadí probíraného tématu, je hned na počátku práce seznámen se zásadami tvorby počítačových her. Následuje teoreticko-metodologická část předkládající teoretické poznatky, vztahující se k řešenému problému a charakteristiku metod a postupů, které by mohly být pro řešení problému použity. Praktická část se věnuje aplikaci získaných poznatků na projekt hry Space Traffic a tedy využitím nabytých informací při zajišťování dostatečné podpory projektu, jeho vedení a řízení. Závěr práce je pak rozborem získaných zkušeností a doporučením pro další vývoj.

Poznatky shrnuté v diplomové práci mohou využít nejen studenti angažující se v projektu Space Traffic nebo jiných softwarových projektech, ale mohou být přínosem také pro projekty realizované v prostředí s vlastnostmi podobnými prostředí akademickému. Mohou to být např. prostředí některých open-source projektů nebo mladých softwarových firem.

[1] V souvislosti s tímto tématem je nejčastěji doporučováno PMBOK® Guide [1] nebo PRINCE2™ [2].

2 Tvorba počítačových her

Programování her, zejména programování webových her, je pro mnohé lidi jak koníčkem, tak i profesní kariérou. Programování her se také mnohdy používá jako praktický nástroj pro studenty učící se základy programování. Důvodem je to, že programování hry je pro studenty daleko zajímavější než programování čehokoli jiného.

Účelem této kapitoly je seznámit čtenáře s tvorbou počítačových her. Kapitola na úvod velmi stručně popisuje jejich historii, dále podává obecné informace a vysvětluje základní pojmy spojené s herním průmyslem a seznamuje čtenáře s procesem tvorby počítačových her.

Většina lidí má představu o tom, co je to hra a nějakou již určitě hrála. Formulace její definice však není ustálená. Podle názoru Katie Salen a Erica Zimmermana [3] je hra systém, v němž jsou hráči zapojeni do umělého konfliktu vymezeného pravidly, který vede ke kvantifikovatelnému výsledku. Greg Costikyan [4] hru povyšuje dokonce na formu umění, ve kterém zúčasnění hráči činí rozhodnutí za účelem ovládnutí herních zdrojů ve snaze dosáhnout cíle, přičemž jejich rozhodování a cíle, kterých mohou dosáhnout, jsou omezeny herními pravidly.

Přehled historie počítačových her

Za první grafickou počítačovou hru je považována hra OXO vytvořená v roce 1952 Alexanderem S. Douglasem jako názorný příklad v jeho doktorské práci. Jedná se o hru piškvorky na hrací ploše o velikosti 3×3 políček.

V sedmdesátých letech dvacátého století přispěl rozmach osobních počítačů pro obyčejné uživatele k rozvoji počítačových her a vznikly tak hry jako Arkanoid nebo Frogger.

V průběhu osmdesátých let se stal důležitým herním prvkem příběh, který se ve hrách objevoval nejprve v podobě textu např. ve hře Prince of Persia. Dnes je příběh vyprávěn v průběhu hraní samotnými postavami formou dialogů nebo animace.

S rozvojem internetu začátkem devadesátých let přišly na svět online hry nabízející společnou hru mnoha hráčů – nejprve prostřednictvím podpory vzájemného propojení v nativních počítačových hrách a poté prostřednictvím veřejných herních serverů. Současně začaly vznikat i webové hry reagující na rozvoj webových technologií.

2.1 Kategorizace her

Her, nejen počítačových, je obrovské množství a z různých úhlů pohledu mohou být tříděny do mnoha kategorií. Je mnoho pohledů na to, jakým způsobem a podle jakých parametrů lze hry dělit. Podle provedení jsou rozděleny na počítačové[1], konzolové, stolní, karetní, halové nebo terénní. Podle svého účelu se dělí na odpočinkové, výukové, rehabilitační atp. Zařazení hry do určité kategorie hráči poskytuje první představu právě o jejím provedení a stylu hraní. Tato část kapitoly o tvorbě her se soustředí pouze na hry počítačové.

U počítačových her je základním hlediskem rozdělení platforma. V takovém případě třídíme hry v zásadě podle a operačního systému, který podporují pro svůj běh. Jedná se tedy nejčastěji o hry pro Windows, Linux a OS X. Přikloníme-li se k názoru Tima O'Reillyho, uvedeného v jeho definici webu 2.0 [5], který označuje web 2.0 jako novou platformu počítačového průmyslu, můžeme mezi kategorie dle platformy zahrnout i kategorii webových her.

Webové hry jsou založeny na principu *server – webový prohlížeč*. Proti jiným hrám mají tedy tu výhodu, že uživatel nemusí z internetu stahovat a instalovat žádného klienta, ale otevře prohlížeč a hned zobrazí webovou stránku hry. Pokud webové hry vyžadují pro svůj běh pluginy, jedná se většinou pouze o ty standardní, které už uživatel zpravidla má ve svém prohlížeči nainstalovány (např. Adobe Flash Player, Microsoft Silverlight nebo Java Runtime Environment). Navíc díky rozvinutému širokopásmovému připojení k internetu jsou webové hry snadno a plně dostupné kdykoliv a kdekoliv a nejsou závislé na hardwaru ani softwaru kromě zmíněných pluginů.

Některé počítačové hry nabízí hru pro jednoho hráče (tzv. singleplayer), jiné umožňují hru i pro více hráčů (tzv. multiplayer). Pokud je hra určena pro obrovské množství spolu interagujících hráčů propojených prostřednictvím internetu, je označována pojmem *Massively-Multiplayer Online Game* (MMOG). Tyto hry většinou budují virtuální společenství a tvoří rozsáhlé virtuální světy, ve kterých se hráči potkávají a v závislosti na žánru hry spolu interagují.

Hráči však nejvíce řekne informace o žánru hry. Žánrové kategorie rozdělují hry podle cílů a také podle typu zážitku, který hry hráči mohou poskytnout. Skupina her stejného žánru definuje díky této společné vlastnosti také podobný způsob a styl, jakým hráč hru ovládá a tudíž nabízí i podobné prostředky pro interakci s hráčem. V dnešní době se stále častěji objevují hry, které v sobě kombinují více žánrů a často jim to bývá přínosem. V následujícím textu naleznete stručný popis základních žánrových kategorií podle studie Thomase H. Apperleyho [6].

Adventury (Adventure) jsou dobrodružné hry bez akcí, které by vyžadovaly reflexivní reakce. Obvykle hra vyžaduje, aby hráč řešil s využitím vzájemné interakce s lidmi nebo herním prostředím různé hlavolamy a to ve většině případů nekonfliktním způsobem.

Akční hry (Action games) po hráči vyžadují, aby se zapojil do extrémních netrivi-

[1]Pojmem *počítačová hra*, je v této práci označována hra určená pro hraní na osobním počítači.

álních činností (akcí), při kterých musí reflexivně reagovat na vzniklé události a přesnost a načasování své reakce pro překonávání překážek. V souvislosti s tímto žánrem se často hovoří o dělení her podle pohledu hráče. Pokud se hráč prostřednictvím obrazovky na hru dívá z pohledu avatara, za kterého hraje, jedná se o tzv. *first-person shooters* (FPS). Pokud je hráčův avatar na obrazovce vidět, jde se o tzv. *third-person shooters* (TPS).

RPG (Role-playing game) je žánr úzce spjatý s literárním žánrem fantasy. Hráč se ve hrách tohoto žánru ocitá v roli nějaké fantasy postavy a hraním buduje její charakter. Současně se potýká s řadou výzev a úkolů a jeho úspěch je měřen hromaděním odměn, získáváním zkušenostních bodů a postupem postavy do vyšších úrovní (levelů).

Simulátory (Simulation) se snaží co nejlépe a poměrně přesně simulovat reálné činnosti a zároveň zcela nepodřizují věrohodnost simulace požadavkům na zábavu. Podle Thomase Apperleyho [6] zahrnuje žánr simulátorů hry, které simulují sporty, létání, řízení, a hry, které simulují dynamiku měst a malých obcí.

Strategické hry (Strategy) po hráči vyžadují kombinovat různé dostupné možnosti a schopnost správně je vyhodnotit v souvislosti s událostmi ve hře. Žánr je tvořen hrami, které kladou důraz na hraní zaměřené na kalkulaci s hodnotami proměnných v závislosti na aktuálním kontextu. Tento žánr se dále dělí ještě na dvě podkategorie *turn-based strategy* (TBS) a *real time strategy* (RTS). V případě TBS se jednotliví hráči střídají tah po tahu a v případě RTS hrají současně.

Jak již bylo zmíněno v úvodu této podkapitoly, podle svého účelu je možné hry dělit na odpočinkové, výukové, rehabilitační atp. Vzhledem k náplni vytvářené hry Space Traffic (viz kapitola 4.2 Game design) se předmětem pozornosti této práce stávají výukové hry. Podle výzkumů (např. M. Klement [7]) se výukový software postupně dostává jak do výuky, tak do téměř všech forem vzdělávání. Hry totiž mají stimulační náboj a podle obsahu mohou podněcovat tvořivost, rozvíjet koncepční a strategické myšlení, učit sociálním dovednostem a v neposlední řadě pozitivně působit na rozvoj počítačové gramotnosti, jak tvrdí Jiří Dostál [8, 9].

Pokud je výuková hra určena pro více hráčů, pak podle studie *Webgame Based Collaborative Learning Design: A Case Study* [10] může u hráčů rozvíjet týmovou spolupráci a schopnost učit se od svých spoluhráčů nebo protihráčů. Hráči společně tvoří taktické plány, řídí svoji činnost a koordinují se. Sledují praktiky své i svých protihráčů a vyhodnocují jejich úspěšnost. Hry tedy u hráčů mohou podle výše zmíněné studie rozvíjet jejich vyšší myšlenkové dovednosti[2].

Nesmí se však zapomenout na to, že i výukové hry by měly být stejně tak zábavné jako výukové. Účelem by tedy mělo být poskytnout hráči obsah cílený na zábavu ve stejné míře jako obsah cílený na výuku.

Stále častěji se objevuje také trend tzv. *gamifikace*. Tento „nástroj“ se naopak místo využití her jako pomůcek, např. při učení, snaží využít herní principy a aplikovat je v reálném světě na činnosti nebo procesy tak, aby zvýšil motivaci lidí k jejich vykonávání. Herní vývojáři již více než 30 let sbírají zkušenosti, jak hráče motivovat. Základní myšlenkou

[2]Jedná se o dovednosti kritického myšlení, řešení problémů, kreativního myšlení a rozhodování [10].

gamifikace je využít těchto zkušeností a začlenit do činností motivační prvky jako např.:

- získávání bodového ohodnocení a zvyšování levelu,
- porovnávání v žebříčcích s ostatními
- a získávání achievementů[3] nebo dárků.

2.2 Game design

Základem každé hry je její obsah, způsob a možnosti interakce hráče s hrou samotnou a pravidla. Ve spoustě případů bývá téma hry vsazeno do virtuálního herního prostředí se smyšlenými a velmi často nereálnými vlastnostmi. Děj hry také mnohdy bývá součástí více či méně propracovaného příběhu v němž může být hráč ztotožněn se svojí postavou a hrát tak v příběhu nějakou roli.

Návrhem hry od tématu a příběhu, přes pravidla až po návrh vzájemné interakce hráče a hry se zabývá game design – podmnožina oboru vývoje her. Je označován stejným anglickým výrazem i v českém prostředí. Dobré zvládnutí této činnosti je klíčem k zaujetí hráče, jeho vtažení do hry a de facto i k úspěchu celé hry. Toto téma je velmi dobře zpracováno v publikaci *The Art of Game Design: A Book of Lenses* [11] od Jesseho Schella. Základním předpokladem úspěchu je podle Schella abstrakce hry od její skutečné realizace. Zvolené téma, motiv a pravidla je pak možné použít jako předloha nejen pro počítačovou, ale třeba také pro stolní společenskou nebo pohybovou hru. Kdokoli i bez technických znalostí tedy může zvládnout základy game designu.

2.3 Proces tvorby her

Jak je zřejmé z názvu této podkapitoly, následující text se bude týkat procesu tvorby her a bude se snažit proces popsat co možná nejobecněji. V souvislosti s tímto tématem budou vysvětleny běžně používané pojmy v tomto oboru. Dále popsané informace byly čerpány z přednášky *Co je game design a jak vlastně vzniká* vedoucího game-designera firmy Cauldron Vladimíra Geršla [12].

2.3.1 Přípravná fáze

Celý proces začíná u zrodu nápadu na novou hru. Zdroj nápadu je záležitostí lidské kreativity a neexistují žádné metody pro vytvoření dokonalé hry. Dříve než je možné se opřít do vývoje hry, je nutné si položit několik níže uvedených otázek.

[3] *Achievement* je běžně užívaný výraz pro ocenění hráče za dosažení nějakého milníku ve hře.

Jaké jsou hlavní pilíře hry?

Pro zodpovězení této otázky je potřeba se zamyslet nad hlavními nápady, které vytvářenou hru odlišují od ostatních, nápady, které přitáhnou hráče, a nápady, na kterých bude hra postavena. Výsledkem je pak dokument velmi podobný dokumentu game designu. Tento dokument by neměl přesahovat deset stran a je označován anglickým názvem *10 Pager*. Z důvodu maximální stručnosti bývají na základě tohoto dokumentu při vývoji vytvářeny prototypy hry.

Jaké je cílové „publikum“ a platforma?

Vědět, jakou cílovou skupinu hráčů má hra oslovit a pro jaké platformy bude vyvíjena, je jednou z nejdůležitějších věcí, které by měly být zcela vyjasněné, protože cílové skupině je dále podřízena velká část návrhu. Je třeba si uvědomit, že vzhledem k předpokládané počítačové gramotnosti cílové skupiny musí být přizpůsobeno ovládání a vzhledem k věku skupiny musí být přizpůsoben obsah. Cílová platforma pak nepochybně určuje jisté možnosti nebo omezení pro využití některých technologií a hardwarových prostředků.

Jaká konkurence se pohybuje na trhu a jak je silná?

V případě komerčního projektu je vždy dobré před započetím vývoje provést analýzu trhu a konkurence. Zjišťují se reference na existující hry zabývající se podobným tématem a zdůrazňují se odlišnosti od vlastního konceptu. Dále se stanovuje zařazení vytvářené hry v současném herním prostředí. Výsledkem provedené analýzy je dokument zvaný *Competetive analysis*, který shrnuje výše uvedené poznatky.

Kolik bude vývoj hry stát?

Podmínkou úspěšné realizace nápadu schopnost zajistit pro vývoj produktu nejen lidské ale i finanční zdroje. Je více možností jak tuto situaci řešit a záleží vždy na rozsahu projektu a jeho předpokládané době realizace[4]. Provádí se tedy analýza financí a určují se předpokládané výdaje, na jejichž základě musí být učiněno rozhodnutí – jakým způsobem bude vývoj financován.

Buď je možné projekt financovat s výpomocí rodiny popř. přátel, nebo je možné oslovit investory či herní vydavatele. Ve všech případech se však stejně dříve nebo později dostane slovo právě na herního vydavatele (angl. video game publisher). Jde o společnost, která publikuje hry a je zodpovědná za jejich výrobu, reklamu a uvedení na trh, kromě toho vývoj her ve většině případů také sponzoruje.

[4]Jedná se o tzv. *trojimperativ projektu*, který je probrán v kapitole 3.2.

2.3.2 Vývojová fáze

Jsou-li provedeny všechny přípravy, vytvořen dokument game designu, zajištěny všechny zdroje, tak je možné začít hru vyvíjet. Během vývoje hra dosáhne několika milníků, které zde budou přiblíženy.

First playable

Hra je ve stavu první hratelné verze a je spustitelná mimo vývojové prostředí. Poskytuje jednu úroveň (kolo, level) nebo např. dvě minuty hraní. Může obsahovat placeholdery[5], ale musí ukázat potenciál hry. Jedná se o významný milník, ve kterém je hra schopna předvést gameplay[6].

Vertical slice

Jak název napovídá[7], tohoto milníku hra dosáhne, když je schopná prokázat pokrok ve všech částech. To znamená, že jsou implementovány všechny důležité funkce, vlastnosti i módy, které představují výsledný produkt v kvalitě blížící se požadované kvalitě výsledku a je tedy možné všechny tyto části předvést. Od podoby hry dosahující tohoto milníku je možné odvodit demo verzi.

Pojem „vertical slice“ se v tomto případě vztahuje ke všem vrstvám nebo jednotlivým komponentám (funkcím aplikace), které ji tvoří, jak je demonstrováno na obrázku 2.1. Šířka řezu představuje rozsah demonstrované části každé komponenty.

Alfa verze

Tento milník označuje stav hry, kdy je implementována kompletní funkčnost, všechna menu a všechny levely, přičemž hra může obsahovat placeholdery. V tomto stavu hra bývá nestabilní, ale může ukázat cílové skupině, jakou veškerou funkčnost bude výsledek obsahovat.

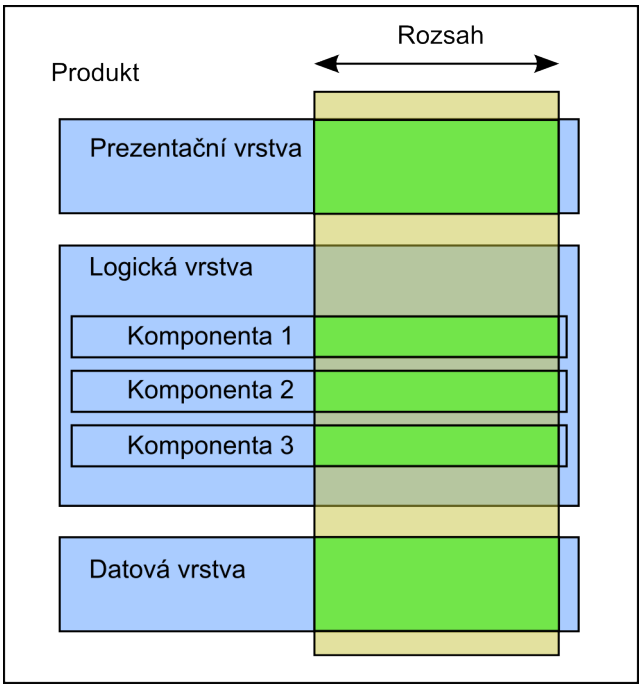
Beta verze

Beta verze hry již obsahuje kompletní obsah. To znamená, že nechybí žádné dialogy ani textury a neobsahuje žádné placeholdery. Jediným nedostatkem hry, která dosáhne tohoto milníku, jsou chyby.

[5] *Placeholder* je objekt, který dočasně zastupuje jiný ještě nevytvořený finální objekt. Má podobný tvar a vlastnosti a demonstruje chování finálního objektu, za který bude později vyměněn.

[6] *Gameplay* představuje specifický způsob, jakým hráči interagují se hrou; je to vzorec hraní vymezený pravidly hry. Zdroj [3].

[7] *Vertical slice* – svislý řez



Obrázek 2.1: Vertical slice – svislý řez softwarovým produktem.

Release

Hra je ve stavu neobsahujícím závažné chyby, které by znemožňovaly její hraní, a její kvalita je dostatečná pro distribuci. Tato verze hry bývá často označována *Gold version*.

3 Řízení softwarových projektů

Řízením pracovních činností se na vědecké úrovni začal jako první věnovat Frederick Winslow Taylor. Na přelomu devatenáctého a dvacátého století začal s podrobnou studií práce [13], v níž vědecky odůvodnil, že je možné pracovní činnosti analyzovat a zlepšovat. Dnes je označován za otce vědeckého řízení (angl. scientific management). Další předkové řízení projektů, jak popisuje Morgen Witzel [14], jsou Henry Gantt, který vyvinul techniky plánování a kontrolování, a Henri Fayol, který vyvinul základní teorii podnikové administrativy a formuloval pět funkcí řízení, které tvoří základ souboru znalostí projektového řízení – plánovat, organizovat, přikazovat, koordinovat a kontrolovat.

Cílem kapitoly je seznámit čtenáře s metodikami pro řízení softwarových projektů. Budou zde kromě konkrétních vývojových metodik, jako jsou například vodopádový model, Scrum nebo RUP, popsány základní principy platící obecně pro řízení projektů, bez ohledu na to, postupuje-li se při vývoji softwaru sekvenčně nebo iterativně. Nejprve budou vysvětleny základní pojmy a obecné postupy, dále budou představeny konkrétní metody řízení vývoje softwaru a způsob jakým lze postupovat při výběru vhodné metodiky.

3.1 Projekt

Pro správné pochopení problematiky řízení softwarových projektů je nejprve nutné vysvětlit samotný termín projekt. „Projekt je dočasné úsilí vynaložené za účelem vytvoření jedinečného produktu, služby nebo jiného výsledku. Dočasnost projektů naznačuje určený začátek a konec. Konce projektu je dosaženo až ve chvíli, kdy je dosaženo jeho cílů, nebo když je projekt ukončen z důvodu nedosažitelnosti stanovených cílů, a nebo když už není samotná existence projektu potřebná.“ Uvádí PMI (Project Management Institute), nezisková organizace zabývající se řízením projektů již od roku 1969, v publikaci PMBOK® Guide [1] uznané institutem ANSI za americkou národní normu.

Ve stejném duchu je i formulace Zdenka Staníčka [15], který v ní zdůrazňuje důležitost stanovení cílů a poukazuje na zdroje, které projekt spotřebovává: „...jedinečná soustava činností směřujících k předem stanovenému a jasně definovanému cíli, která má určený začátek a konec, která vyžaduje spolupráci různých profesí, váže jejich kapacity a jejich úsilí a využívá (případně spotřebovává) pro vytvoření cílových výstupů informace, materiál, peníze, schopnosti a dovednosti zúčastněných lidí“.

Ze své podstaty je každý projekt jedinečný, protože i jeho výsledek je jedinečný. I když jsou dva projekty zpracovávány stejným týmem, jsou spotřebovávány stejné nebo podobné zdroje a jsou stanoveny podobné cíle, kontext projektu je jedinečný s různými okolnostmi, různými dodavateli atp. „Přestože mohou být a často také jsou v projektech spatřovány opakující se prvky a mohou být na podobných projektech prokázány vzory vykonávaných

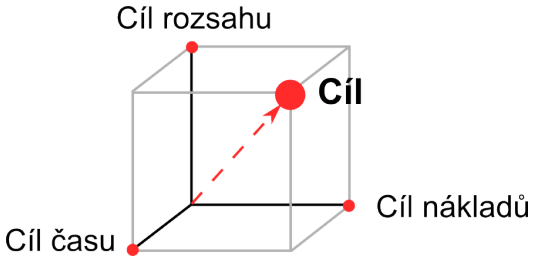
činností, nemění toto opakování základní jedinečnost projektu.“ Uvádí PRINCE2™ [2] (Projects in controlled environments).

3.2 Omezení projektu

Každý projekt je jistým způsobem omezen, a to svým rozsahem, časem a náklady, stejně jako svými cíli [15, 16]. Rozhodujeme-li o jeho úspěchu či neúspěchu, nepochybně sledujeme, jestli byly naplněny jeho stanovené cíle. Pokud ovšem nebyly dosaženy včas, ale se zpožděním, nebo byl-li překročen stanovený rozpočet a nebo kvalita výsledku patřičně neodpovídá očekáváním zákazníka, jen těžko může být projekt prohlášen za úspěšný.

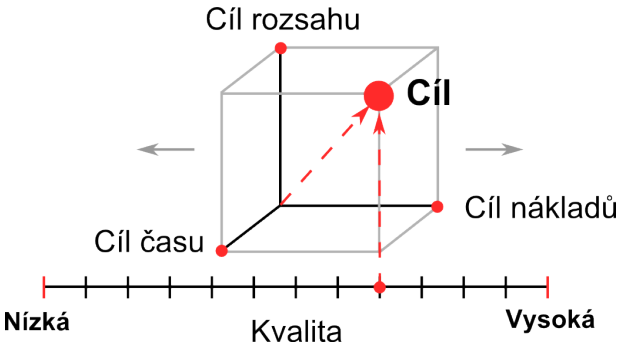
Úspěch projektu proto spočívá ve splnění cílů projektu ve všech těchto rozměrech. V české terminologii jsou tato omezení často nazývána trojimperativ projektu, v anglicky psané literatuře se setkáme s termínem *triple constraint*. Vztah mezi specifikací rozsahu, přípustnou dobou trvání a náklady projektu je možné schematicky znázornit v prostoru, jak je tomu na obrázku 3.1 převzatého z [16]. Kim Heldman [17] uvedl: „as a project manager, one of your biggest jobs is to balance the triple constraints while meeting or exceeding the expectations of your stakeholders“. Řešením trojimperativu je tedy nalezení vhodného vztahu mezi specifikací rozsahu, časovou lhůtou a náklady projektu, tak aby byla naplněna očekávání všech subjektů, kteří mají na projektu nějaký zájem (tj. stakeholders).

Správné vyvážení mezi jednotlivými omezeními je obvykle velmi nelehký úkol, protože každá z veličin (rozsah, doba trvání a náklady) má na začátku projektu stanoven nějaký cíl. Tyto cíle většinou vedou k protichůdným požadavkům, protože je vždy snaha o naplnění maximálního rozsahu při minimálních nákladech a v co nejkratší době. U většiny projektů je tedy třeba hledat vhodný kompromis.



Obrázek 3.1: Trojí omezení projektu.

Má-li projekt uspokojit zákazníka či zadavatele (potažmo všechny stakeholdery), musí být kromě výše uvedených rozměrů zvládnut ještě jeden – kvalita produktu. Mnohdy se tak setkáváme s termínem *quadruple constraint* (viz obr. 3.2). Kim Heldman [17] vidí specifikovaný rozsah jako neměnný a projekt je podle něj omezen trojicí čas, náklady a kvalita. Katy Schwalbe [16] předkládá názor, že otázky kvality, a tedy i spokojenosti zákazníka, musí být nedílnou součástí rozsahu, času i nákladů na projekt a projektový tým se musí těmto otázkám odpovídajícím způsobem věnovat.



Obrázek 3.2: Čtvero omezení projektu.

U každého projektu je na začátku nutné určit, která omezení jsou rozhodující. Obvykle je projekt vyrovnáván pomocí jednoho či dvou. Výstižně to na příkladu popsal Kim Heldman [17]: „if the project goal is a high-quality product, the saying goes, *I can give it to you fast or I can give to you cheap, but I can't give it to you fast and cheap*".

3.3 Obecně o řízení projektů

Obecně lze říci, že cílem projektového řízení je zajistit naplánování a realizaci projektu tak, aby v plánované době a s plánovanými náklady bylo dosaženo stanovených cílů projektu s optimálním využitím přidělených zdrojů. Ve stejném duchu se nese definice uvedená v PMBOK® Guide [1]: „Řízení projektu je uplatnění veškerých poznatků, dovedností, nástrojů a technik na aktivity projektu takovým způsobem, aby byly splněny požadavky na projekt".

Za tuto činnost je pak zodpovědný projektový manažer. Jeho role je tímto v projektu velmi významná. Musí komunikovat se zadavatelem během celé doby řešení, aby zajistil splnění všech jeho očekávání, musí spolupracovat s vlastním projektovým týmem a musí umět pracovat s nástroji a technikami pro řízení projektu. Kromě toho, musí dobře zvládnout níže popsané oblasti řízení, důkladně rozebrané v publikacích PMBOK® Guide [1] a Kate Schwalbe [16], zde je uvedeno jen stručné shrnutí.

Řízení rozsahu projektu znamená definování a řízení veškerých prací, nezbytných k úspěšnému dokončení projektu.

Řízení času v projektu představuje odhad doby potřebné k dokončení prací, návrh přijatelného časového plánu a zajištění včasného dokončení projektu.

Řízení nákladů se v projektu skládá z přípravy a řízení rozpočtu projektu.

Řízení kvality projektu zajišťuje, že projekt naplní stanovené nebo předpokládané potřeby, pro jejichž splnění je řešen.

Řízení lidských zdrojů v projektu se zabývá efektivním nasazením lidí do realizovaného

projektu.

Řízení komunikace znamená generování, shromažďování, distribuci a ukládání informací o projektu.

Řízení rizik se skládá z rozpoznávání rizik, jejich analýzy a odpovídající reakce na ně.

Řízení zakázek (procurement) znamená obstarávání zboží a služeb potřebných pro řešení projektu, které pocházejí z vnějších zdrojů.

Organizace PMI doporučuje prostřednictvím publikace PMBOK® Guide, kterým oblastem by měl projektový manažer rozumět a měl by v nich mít nějaké zkušenosti:

- základní znalosti v oblastech pro řízení projektů (viz výše),
- znalosti, standardy a předpisy pro příslušnou oblast aplikace,
- poznatky ohledně prostředí projektu,
- obecné znalosti a dovednosti řízení,
- sociální dovednosti neboli umění vycházet s lidmi.

Jak je vidět a jak uvádí i mnohé publikace [1, 2, 16, 18], na úspěchu se vždy velkou mírou podílí projektový manager, který musí jít celému týmu příkladem, aby dobře motivoval. Projektový manager je zodpovědný za vývoj produktu a musí se všemi účastníky projektu docílit naplnění jejich potřeb a očekávání. Úspěšný projektový manager si dokáže s účastníky projektu vybudovat dobré vztahy, tím dokáže správně pochopit jejich potřeby a očekávání a během projektu je naplnit.

3.4 Analýza metodik vývoje softwaru

V souladu s výše uvedenými definicemi projektu je možné říci, jak uvádí Zdenko Staníček [15], že „projekt je jednorázová transformace vstupů (informace, prostředí, materiál, peníze, schopnosti a dovednosti zúčastněných lidí) na výstupy — cílové produkty — za pomocí vývojových činností, uspořádaných do etap, kroků a úkonů a koordinovaných řídícími činnostmi“. A právě různými způsoby, jak uspořádat činnosti vývoje softwaru do etap a jak je řídit, se zabývá tato část kapitoly o řízení softwarových projektů.

Jsou rozlišovány dva přístupy k řízení softwarového vývoje [19] – prediktivní a adaptivní. Prediktivní přístup se vyznačuje tím, že rozsah projektu a požadavky na výsledný produkt jsou formulovány před zahájením vývoje softwarového produktu. Výhodou je možnost relativně přesně předvídat časový plán i náklady na projekt. Tento přístup podporuje také optimalizaci vývoje, ale toto všechno je vykoupeno zhoršenou, někdy i znemožněnou adaptabilitou na možné změny. Mezi prediktivní modely podle Desaulniers a Andersona [19] patří vodopádový model, V-model, spirálový model a také prototypování.

Adaptivní přístup vývoje softwaru se zaměřuje především na stanovené cíle projektu. Je vhodný u takových projektů, u kterých není možné na začátku přesně určit požadavky. Ty jsou sbírány a sestavovány iterativní metodou často i během samotného vývoje softwarového produktu. Vývoj je přizpůsobován aktuálním podmínkám a proto je tolerantní ke změnám. Z tohoto důvodu vyžaduje úzkou spolupráci mezi projektovým týmem a zadavatelem. Tento přístup zahrnuje modely extrémní programování, Feature Driven Development a Scrum.

Metodiky vývoje softwaru lze dělit také z hlediska formalizace na rigorózní metodiky a agilní metodiky. Rigorózní metodiky přesně popisují, plánují, řídí a měří procesy, činnosti a vytvářené produkty (obvykle bývají velmi objemné). Většinou jsou založeny na vodopádovém modelu, najdou se však i takové, které jsou postaveny na iterativním vývoji. Nejznámějším reprezentantem této skupiny je metodika Rational Unified Process.

Naopak agilní metodiky vývoje softwaru, jejichž hlavním cílem je vytvoření funkčního produktu, který přináší maximální hodnotu zákazníkovi, se soustředí především na přímou komunikaci mezi lidmi, spolupráci se zákazníkem a reakci na změny. Přitom odsouvají právě procesy, dokumentaci a přesné dodržování plánu. Mezi nejznámější zástupce patří extrémní programování či Scrum.

3.4.1 Vodopádový model

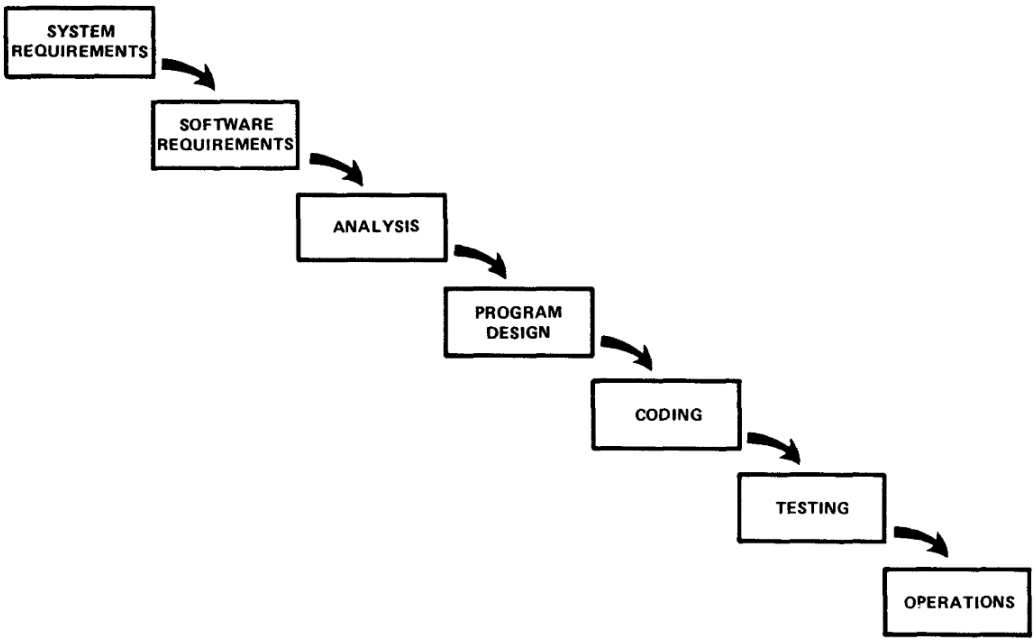
Tento model je jednou z nejstarších metod řízení projektů vývoje softwarových produktů, která staví jednotlivé etapy vývoje sekvenčně za sebou od sběru požadavků až po testování a předání produktu zadavateli (Winston Royce [20]). Etapy vývoje se neopakují ani nepřekrývají (viz obr. 3.3). Zjevným problémem vodopádu je, že v něm neexistují žádné zpětné vazby. Chyby budou vždy nalezeny až v etapě testování, a pokud opravy budou vyžadovat změnu už na úrovni návrhu, znamená to opakovat velkou část vývojového cyklu znova a to často s velkými zásahy.

Vodopádový model předpokládá, že požadavky definované na začátku vývoje zůstanou neměnné. Může být tedy použit jen v případě, kdy je očekávaný produkt jasně popsán.

3.4.2 V-model

V-model je upravenou verzí vodopádového modelu. Na rozdíl od něj však není navržen v lineární ose. Jak je vidět na obrázku 3.4, jeho jednotlivé testovací etapy jsou po zlomu v etapě kódování zakresleny vzestupně a vytvářejí tak pro něj typický tvar písmene „V“. Tento model zavádí vazbu mezi každou etapou vývojového cyklu a její související testovací etapou. Výhodou proti vodopádovému modelu je, že každá etapa je otestována.

Také v případě V-modelu se předpokládá, že požadavky budou během vývoje stabilní. Protože v každé etapě je možnost vzniku chyb a první testování se provádí až po etapě kódování a to je relativně pozdě, jsou opravy také velmi nákladné.



Obrázek 3.3: Vodopádový model řízení projektu. Zdroj: Winston Royce [20].

3.4.3 Prototypování

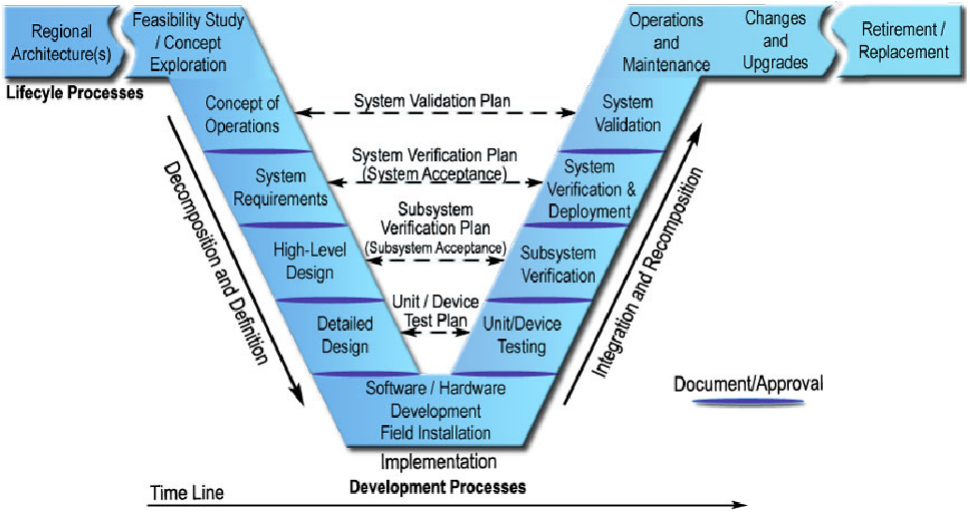
Model prototypování je iterativní metodikou, při které jsou funkční požadavky získávány současně s vytvářením specifikace návrhu. Nejprve jsou vytvářeny prototypy – reálné modely aplikací, na kterých zadavatel upřesňuje své požadavky na výsledný produkt. Opakují se zde tedy etapy návrhu, vývoje prototypu a jeho evaluace zadavatelem. Vytvořené prototypy mohou být jak zahozeny, tak ponechány pro další implementaci.

Prototypovat je možné jak uživatelské rozhraní, tak funkci aplikace. Prototypy uživatelských rozhraní jsou obvykle vytvářeny v různých grafických editorech či návrhářích. Při prototypování funkce aplikace se využívá syntetizovaných funkčností.

Prototypování se velmi hodí při analýze požadavků. Prototypy totiž pomáhají uživatelům (zadavateli) získat představu, jak bude výsledný systém vypadat a vedou tak k menším pozdějším změnám v aplikaci. Tento model vyžaduje výrazné zapojení zadavatele, který vyhodnocuje prototypy. Vývojáři pak na základě jeho hodnocení a připomínek sepisují požadavky.

3.4.4 Spirálový model

Tento model je spojením modelu prototypování a vodopádového modelu (Barry Boehm [22]). Byl vyvinut podle zkušeností s různými vylepšeními vodopádového modelu a zkušeností s řešením velkých státních zakázek na vývoj softwaru. Podle tohoto modelu je software vyvíjen ve spirálovitých iteracích, nikoliv lineárním způsobem. Opakují se zde,



Obrázek 3.4: V-model řízení projektu. Zdroj: [21]

podobně jako v případě prototypování, etapy určení cílů, analýza a vyřešení rizik, vývoj a plánování následující iterace. Na obrázku 3.5 je vidět, jaké činnosti jsou v rámci každé etapy prováděny v závislosti na „stáří“ projektu.

Spirálový model byl vyvinut pro použití při velkých, drahých a komplikovaných projektech.

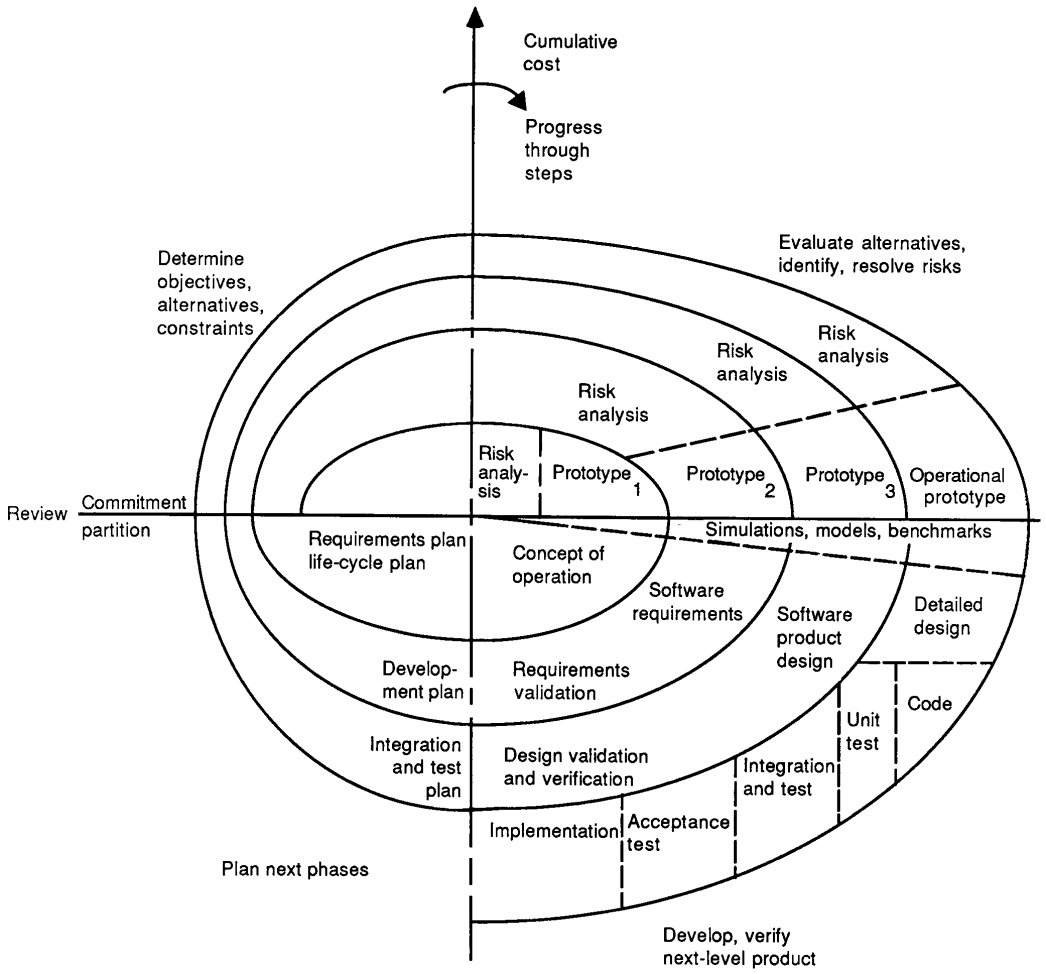
3.4.5 Extrémní programování

Extrémní programování (XP) se vyznačuje následujícími charakteristickými znaky: programování je prováděno ve dvojicích, iterativní proces, kolektivní odpovědnost a vlastnictví kódu. Součástí vývojového týmu jsou jak vývojáři a manažeři, tak i zástupci uživatelů. Tato metodika zajišťuje při vývoji v měnícím se prostředí snadněji reagovat na postupně se objevující potřeby uživatelů. Programování ve dvojicích posiluje týmovou spolupráci, zvyšuje produktivitu a kolektivní vlastnictví zdrojového kódu.

Tato metodika je vhodná pro týmy do dvaceti lidí [16] a zvyšuje spokojenost cílového uživatele. Vyžaduje však zapojení zástupce uživatelů na plný úvazek a nelze na začátku projektu stanovit závazný časový plán a náklady.

3.4.6 Feature Driven Development

Metodika Feature Driven Development (FDD) je agilní metodikou, která ve zjednodušené podobě zachovává procesní řízení a klade důraz na doménové objektové modelování. FDD je iterativní metodika s krátkými dvoutýdenními iteracemi. Klíčovými činnostmi vyznačujícími FDD, jak je popisují Palmer a Felsing [23], jsou: doménové objektové modelování, vývoj podle tzv. užitných vlastností, individuální vlastnictví tříd, týmy pro jednotlivé



Obrázek 3.5: Spirálový model řízení projektu. Zdroj: Barry Boehm [22].

užitné vlastnosti (tzv. Feature Teams), inspekce, pravidelné buildy, řízení konfigurací a vykazování a viditelnost výsledků.

Každý vývojář je zodpovědný za některé třídy doménového modelu a každý tým je zodpovědný za nějakou užitnou vlastnost. Užitné vlastnosti jsou požadavky na funkce s hodnotou pro zákazníka popsané v řeči uživatele. Každá užitná vlastnost je nanejvýš tak velká, aby ji bylo možné implementovat v rámci jedné iterace.

Díky zachování formalismu procesů a modelování je možné metodiku FDD aplikovat i na větších projektech a projektech kritických pro poslání organizace [23].

3.4.7 Scrum

Při metodice Scrum probíhá vývoj také iterativně a je přizpůsoben pro snadnou reakci na měnící se požadavky. Tato metodika používá vlastní terminologii a definuje některé

speciální role lidí v projektovém týmu. Role *product owner* je osoba, která zodpovídá za určování priorit, tudíž i za plánování úkolů a mnohdy určuje implementační detaily. Není to člověk od zákazníka. Další rolí je *scrum master*, který řídí vývojáře a všemožně jim pomáhá při vývoji. Stará se tedy i o to, aby jim fungovaly počítače, měli dostupný potřebný software a řeší spory. Pak zde také existují stakeholdeři a manažeři. Mezi stakeholdery patří lidé od zákazníka, testeři a další osoby, které mají na výsledku nějaký zájem. Manažeři jsou pak osoby, které nejsou vývojáři, product owneři, ani scrum masteři, ale pomáhají nastavit a řídit pracovní prostředí.

Scrum zavádí pojem tzv. *product backlog*, který označuje jakýsi zásobník obsahující všechny scénáře a vlastnosti, které systém zahrnuje. Product backlog je pruduct ownerem řazen podle priorit jeho jednotlivých položek. Iterace jsou nazývány *sprinty*, jsou time-boxované[1] a trvají obvykle třicet dní. Každý den se provádí tzv. *daily scrum*. Jedná se o krátkou schůzku se scrum masterem (často i ve stoje), během které všichni členové týmu popíšou, co dělali předešlý den, co budou dělat dnes a jestli při plnění úkolů narazili na nějaké překážky, které jim brání v práci.

Tato metodika vyžaduje silnou vůdčí pozici scrum mastera, který musí koordinovat práce jednotlivých menších týmů [16].

3.4.8 Rational Unified Process

Rational Unified Process (RUP) vznikl z metodiky Unified Process (UP) a je její nejznámějším derivátem. Díky svému rozsahu a podrobné specifikaci vývojových procesů není jen metodikou, ale spíše frameworkem, který je možné přizpůsobit potřebám realizovaného projektu a procesnímu prostředí konkrétní organizace. Z důvodu své přizpůsobivosti je často těžké říci, zda aplikovaný postup byl odvozen z UP nebo RUP. Jejich názvy jsou tak proto často zaměňovány.

Vývoj produktu probíhá v time-boxovaných iteracích, jejichž výsledkem je interní *release produktu*, jehož přidaná nebo vylepšená funkcionalita je porovnatelná s předchozím releasem produktu. V závislosti na tom, v jaké fázi vývoje se produkt nachází, jsou v různé míře během iterace prováděny činnosti business modelování, sběr požadavků, analýza a návrh, implementace, testování a nasazení produktu (viz obr. 3.6). RUP definuje čtyři fáze vývoje [24]:

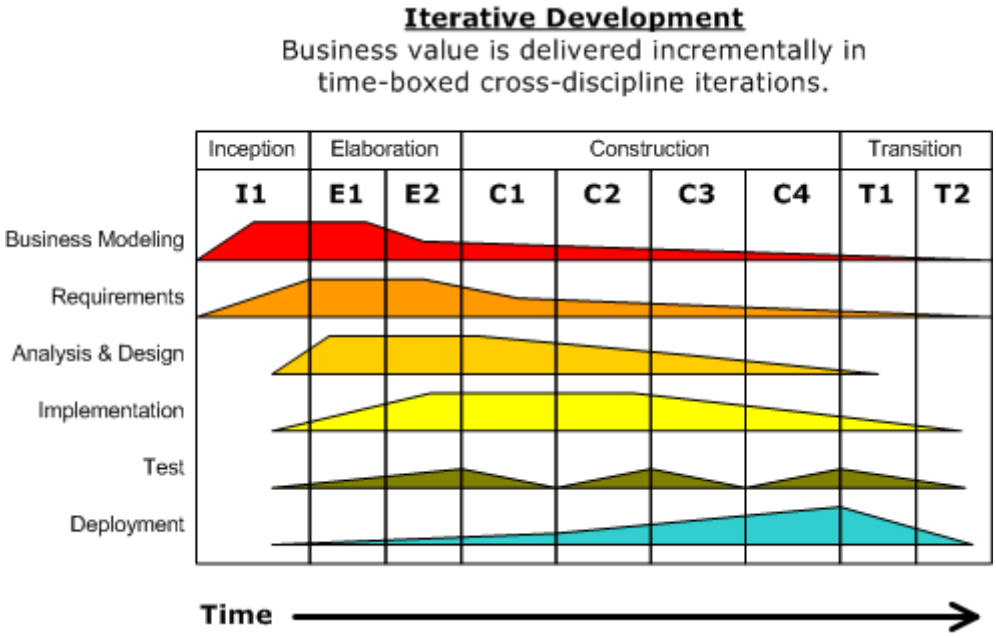
Zahájení (*Inception*) – se zákazníkem je sestavena a validována vize, je stanoven rozsah projektu, klíčové požadavky, jsou zjištěna rizika a je připraven předběžný plán a cena.

Projektování (*Elaboration*) – jsou zjištěny „životně důležité“ požadavky, všechna rizika, je navržena a ověřena architektura. Provádí se tedy především analytické a návrhové činnosti, ověřují se prototypy a implementuje se řešení.

[1] Angl. *time-boxed iterations* – na začátku projektu je stanovena délka iterace a ta je po celou dobu realizace projektu pro všechny iterace dodržena.

Konstrukce (*Construction*) – jde o nejdelší fázi, ve které je vytvořena zbývající část produktu. Jsou prováděny hlavně návrhové a implementační činnosti, testují a ověřují se mezivýsledky, jsou řízeny změny a nové požadavky zákazníka.

Nasazení (*Transition*) – výsledný produkt je u zákazníka integrován do provozu. Během této fáze jsou zaškolováni uživatelé, provozem je získávána zpětná vazba, ověřována správná funkce produktu a spouští se uživatelská podpora.



Obrázek 3.6: Činnosti v jednotlivých fázích metodiky RUP. Zdroj: IBM® RUP [24]

3.5 Rozhodování při výběru metodiky

Projekty jsou jedinečné (viz kapitola 3.1) – liší se svými vstupy a výstupy, řešitelským týmem, schopnostmi a znalostmi jeho členů. V neposlední řadě se také liší cíli projektu, svým prostředím či doménou. Z tohoto důvodu je pro každý projekt vhodná také jiná metodika řízení. Tato část uvede jednoduchý postup, podle kterého bude čtenář schopen zvolit metodiku vhodnou pro řízení projektu.

Pro výběr metodiky je potřeba definovat způsob porovnávání jednotlivých metodik. To znamená určit taková kritéria výběru, která jsou pro řešený projekt nějakým způsobem rozhodující a kritická [25]. Například, je-li pro výsledný produkt kriticky důležitá správná a bezchybná funkce, rozhodujícími kritérii výběru metodiky bude zřejmě rozsah, podrobnost a propracovanost procesů testování a ověřování. Jsou-li stanovena kritéria výběru, dalším krokem je číselné ohodnocení kandidátů pro každé z těchto kritérií. Obvykle se volí

stupnice od 1 do 10, kde 1 představuje naprostou nevhodnost metodiky a 10 znamená úplné splnění požadavků projektu [25].

Jelikož jsou vždy některá kritéria pro projekt významnější než jiná, jsou jim přiřazeny váhy. Tyto váhy jsou aplikovány na ohodnocení kandidátů metodiky. Tím jsou získána data, na jejichž základě je již možné metodiku rovnou vybrat a nebo pro rozhodnutí použít nějakou další metodu vícekriteriální analýzy variant např. metodu TOPSIS (Technique for Order Preference by Similarity to Ideal Solution).

Při rozhodování o výběru metodiky se také ptáme, zda je v daném případě konkrétního projektu vhodnější sekvenční nebo iterativní způsob vývoje. I v tomto případě je jasně viditelná závislost na výše uvedených vlastnostech projektu. Steve McConnell ve své publikaci *Code Complete* [26] uvádí, že pro sekvenční přístup se můžeme rozhodnout v případě, když:

- požadavky jsou poměrně stálé,
- návrh je přímočarý a snadno srozumitelný,
- vývojový tým dobře zná oblast působnosti připravované aplikace,
- projekt není příliš rizikový,
- je důležitá dlouhodobá předvídatelnost.

Musí se však vzít v potaz, že náklady na změny požadavků, návrhu nebo kódu by mohly být poměrně vysoké, protože většina chyb se objeví až při testování. Práce na projektu jsou tak výrazně zpomaleny až ke konci projektu. Naopak iterační přístup může být podle McConnella zvolen v případě, že:

- požadavky nejsou příliš srozumitelné nebo očekáváte, že budou nestálé,
- návrh je poměrně složitý, odvážný nebo obojí,
- vývojový tým nezná dobře oblast, v níž se bude projekt pohybovat,
- projekt obsahuje mnoho rizik,
- dlouhá předvídatelnost není důležitá,
- náklady na změny požadavků, návrhu nebo kódu by neměly být vysoké.

Při výběru způsobu řízení projektu nesmí být opomenuta „míra zralosti“ všech členů týmu, jak uvádí František Bělohlávek ve své publikaci *Jak řídit a vést lidi* [18]. Termín „zralost člena týmu“ Bělohlávek formuluje jako dovednosti a technické znalosti potřebné ke splnění úkolu a připravenost přijmout odpovědnost za jeho splnění. Ve zmíněné publikaci k tomuto tématu uvedl: „Čím vyspělejší a zralejší jsou pracovníci, tím náročnějšího způsobu vedení může manažer použít.“

4 Projekt hry Space Traffic

Účelem kapitoly je seznámit čtenáře s projektem vývoje webové hry Space Traffic se zvláštním zaměřením na používané postupy a nástroje pro řízení projektu v minulých letech. Kapitola seznámí také s historií projektu a uvede technologie použité pro vývoj hry v minulosti i v současnosti. Čtenář bude po přečtení schopen utvořit si obraz o stavu projektu před zahájením této práce a o podstatných znacích akademického prostředí, ve kterém je projekt realizován. Nakonec bude podle vlastního názoru autora vyhodnocen vypozorovaný vliv prostředí realizace na průběh projektu.

Pro zpracování této části bylo čerpáno převážně z diplomové práce Zbyňka Neuderta [27], který měl na starost analýzu a návrh webové hry Space Traffic a vedení vývojového týmu v počátcích projektu, tedy v akademickém roce 2009-2010. Protože se jedná o jedinou v současnosti dostupnou práci věnující se tématu řízení projektu vývoje hry Space Traffic, další informace byly čerpány od studentů zapojených do projektu v posledních letech a z vlastní zkušenosti autora.

4.1 Účel projektu vývoje hry

Záměrem Katedry informatiky a výpočetní techniky (dále jen KIV) Fakulty aplikovaných věd Západočeské univerzity v Plzni je prezentovat veřejnosti činnost svých studentů a zvýšit u středoškolských studentů zájem o studium ve studijních programech, které katedra nabízí. V případě tohoto projektu je požadovaným výsledkem větší podíl přijatých studentů, kteří mají zájmem nejen o programování, ale i o účast na katedrou realizovaných projektech.

Účelem projektu, kterým se zabývá tato práce, a zároveň prostředkem pro uskutečnění záměru katedry je vytvoření hry Space Traffic – vesmírné obchodní strategie typu MMORTS s výukovými prvky. Hlavním pilířem této hry je programování objektů ve hře (vesmírné lodě, hlavní kontrolní počítač). Game design hry je více rozveden v následující podkapitole 4.2, v úplném znění je uveden v diplomové práci Martina Štěpánka [28].

Účelem hry Space Traffic je mimo prezentaci studentských aktivit a zvýšení zájmu o studium dát studentům katedry KIV rovněž příležitost ke spolupráci na reálném zajímavém projektu. Katedra se tak snaží zlepšit jejich praktické zkušenosti s týmovou spoluprací a připravit je lépe do zaměstnání.

4.1.1 Cíle pro rok 2011-2012

Hlavními cíli bylo navržení postupu vývoje hry Space Traffic, zvolení vhodných nástrojů pro zajištění dostatečné podpory ekosystému projektu a dovedení projektu k vytvoření funkčního základu hry.

4.2 Game design

Game design hry Space Traffic je detailně popsán v diplomové práci Martina Štěpánka [28] a vychází z diplomové práce Zbyňka Neuderta [27]. Na tomto místě budou uvedeny a stručně popsány pouze hlavní myšlenky, aby měl čtenář dostatečný přehled i o vytvářeném produktu.

Jak již bylo zmíněno dříve, Space Traffic bude webovou hrou pro více hráčů a žánrově bude spadat do kategorie Massively-Multiplayer Online Real-Time Strategy (MMORTS). Hra je usazena do vesmírného prostředí o rozsahu jedné galaxie. Herním plánem je galaktická mapa složená z dynamických hvězdných soustav, které jsou mezi sebou různě propojeny systémem tzv. červích děr (tj. tunely zkracujícími cestovní vzdálenost)[1].

Hráč ve hře vlastní vesmírné lodě, se kterými se může po herním plánu pohybovat a cestovat z planety na planetu buď v rámci jednoho hvězdného systému, nebo mezi různými hvězdnými systémy právě prostřednictvím červích děr. Kromě lodí hráč manipuluje ještě s dalšími objekty: základnami, továrnami, a sklady. Objektů bude ve hře do budoucna možná mnohem více, neboť game design postupem času takzvaně „zraje“ s nově přicházejícími nápady nebo požadavky i ze strany katedry.

Námětem hry je mezihvězdný obchod s nejrůznějším zbožím. Úkolem hráče je nalézt v okolí svých lodí nejvýhodnější zboží pro nákup, dopravit ho na planetu, na které je ho nedostatek a za co nejvýhodnější cenu prodat. Hráč by se v současné době měl při této činnosti snažit co nejlevněji nakoupit, minimalizovat náklady na převoz zboží a provoz lodí, nakoupené zboží co nejdráže prodat a maximalizovat tak svůj zisk.

Ve hře se budou objevovat prvky programování. Hráč bude své lodě ovládat pomocí svých programů a bude tak moci automatizovat jejich činnost. Pro manipulaci s vesmírnými loděmi, továrnami a sklady na vyšší úrovni si bude hráč moci pronajmou Master Control Computer (MMC). Jeho prostřednictvím bude hráči umožněno globální řízení jím vytvářeného impéria.

Původně byla hra určena pouze pro hráče znalé programování, ale nakonec se uchytila myšlenka vytvoření hry jako výukové. V současnosti se tedy od hráče nepředpokládá žádná vstupní znalost programování a bude do této činnosti postupně uveden nenásilnou formou ukázek podpořenou achievementy. Do budoucna je otázkou, bude-li prostřednictvím hry hráčům umožněno obchodovat i s vlastními vytvořenými programy a pokud ano, jakým

[1] Věcně je herním plánem neorientovaný graf, ve kterém jsou hvězdné soustavy reprezentovány jako uzly a červí díry tvoří hrany.

způsobem bude takový obchod realizován.

Hra je cílena především na hráče ve věku od 15 do 19 let, kteří jsou potenciálními uchazeči o studium v některém ze studijních programů nabízených katedrou KIV. Samozřejmě nevadí, pokud hra zaujme i jiné věkové skupiny.

4.3 Technologie realizace

V této krátké podkapitole je pouze pro úplnost základních informací o vytvářeném produktu stručně uveden výčet technologií, na kterých je založena programová implementace hry. Podrobnosti k výběru a využití jednotlivých technologií je možné získat z diplomové práce Martina Štěpánka [28].

Space Traffic je ASP.NET MVC 3 webovou aplikací. Serverová část je psána v jazyce C# a je nasazena na webovém serveru IIS 7. Perzistentní vrstva je založena na ADO.NET Entity Frameworku a je nasazena na databázovém enginu MS SQL. Aplikační vrstva na klientské straně je vytvářena v JavaScriptu a je spouštěna na straně klienta ve webovém prohlížeči. Dalšími technologiemi použitými ve větší míře jsou jQuery, JSON a SVG pro vykreslování dynamiky hvězdných systémů.

4.4 Historie projektu

Před třemi lety, v roce 2009, vznikl díky vlastní iniciativě studenta Zbyňka Neuderta a podpoře katedry KIV projekt vývoje webové hry pro více hráčů s názvem Space Traffic. Tento student vytvořil scénář hry, stanovil její téma, určil herní možnosti a omezení a definoval vizi projektu. Poté s velkými obtížemi, jak popisuje ve své práci [27] na straně 36, sestavil vývojový tým čítající prakticky dva členy – sebe a jednoho studenta v bakalářském studiu. Tento tým pak řídil podle zvolené metodiky, čímž projekt dovedl k vytvoření funkčního základu hry. Z důvodu nedostatečné velikosti vývojového týmu hra neobsahovala veškerou plánovanou funkcionalitu a nemohla být použita k prezentaci katedry v plánovaném rozsahu.

Pod Neudertovým vedením, byla pro řízení a vývoj projektu zvolena kombinace dvou metodik: Rational Unified Process [24] a Feature Driven Development [23]. Vývojové aktivity na projektu byly rozplánovány do pěti iterací dlouhých dva až tři týdny. Na začátku každé iterace určil Zbyněk Neudert její cíle a na základě jejich analýzy vytvořil v nástroji IBM Rational Team Concert (dále jen RTC) sadu úkolů, které rozdělil mezi členy týmu. Po splnění úkolu byla ověřena správná funkce výsledku daného úkolu testy, které k němu byly přiřazeny.

Z důvodů popsaných v Neudertově diplomové práci [27] na straně 53 vybral její autor pro vývoj technologii PHP a JavaScript. Programová realizace nevyužívala žádný framework, všechen programový kód byl autorskou prací spolupracujících studentů.

Bohužel se následující akademický rok 2010-2011 ukázalo, že architektura hry byla nedostatečně zdokumentována, nebyla snadno rozšiřitelná o další funkcionalitu a velké množství kódu by bylo potřeba zcela přepsat. Nový čtyřčlenný tým pod vedením Richarda Kocmana tedy podal návrh na přepracování stávající implementace tak, aby využívala nějaký framework. Dalším návrhem byl přechod na některý silně typovaný programovací jazyk např. Javu nebo C#, který by umožnil snadnější debug programu. Richard Kocman pak předal návrhy garantovi projektu. Nakonec bylo rozhodnuto o pokračování v implementaci v programovacím jazyce PHP s využitím frameworku Nette.

Jak popisuje Richard Kocman ve své diplomové práci [29], v tomto akademickém roce se práce příliš daleko neposunula, protože na začátku se dlouze rozhodovalo o dalším postupu pro vývoj, dále byly provedeny převážně jen analytické a návrhové práce a v únoru přišel projekt o své vedení. Bez vedoucího tým rychle přestal pracovat a prakticky se samovolně rozpustil. Na konci letního semestru byl projekt „na pokraji smrti“.

I přes svůj neúspěch byl tento akademický rok přínosem. Během vývoje se totiž ukázalo, že použití frameworku při vývoji hry přineslo zjednodušení, ale kvůli slabé typové kontrole jazyka PHP bylo stále obtížné hledání vzniklých chyb.

Na konci tohoto akademického roku 2010-2011 převzali autor této práce a Martin Štěpánek projekt ve výše uvedeném stavu. Práce na projektu byly v takovém stádiu, že nebylo problémem na ně navázat, ale co se týká vůle garanta a zákazníků tento projekt dále podporovat, byla situace výrazně odlišná. Po značném úsilí studentů pokračovatelů se na konec podařilo projekt oživit (viz tato práce a práce Martina Štěpánka [28]).

4.5 Rozsah projektu

Rozsah projektu je v našem případě určen především dokumentem game designu vymezujícím funkčnost, kterou bude hra Space Traffic ve výsledku nabízet. Dokument game design je, jak již bylo uvedeno, součástí diplomové práce Martina Štěpánka [28]. Stejně jako je postupně upřesňován game design, zpřesňuje se i rozsah projektu. Rozsah projektu se však neomezuje pouze na podrobný popis funkčnosti produktu, ale jeho součástí je podle PMBOK® [1] i popis projektu samotného. Autoři PRINCE2™ [2] pak konkrétně uvádějí, že mezi související parametry patří velikost projektu (často měřená z pohledu doby trvání, nákladů a lidí), rizik a významu. Těmto parametrům se věnuje zbylý text této podkapitoly.

Principiálně není možné v prostředí realizace projektu (viz kapitola 4.7) přesně určit (ani spolehlivě odhadnout) pracnost projektu udanou v tzv. studento-měsících[2]. Při výpočtu hodnoty jednoho studento-měsíce se vycházelo z celkového počtu hodin odvedeného během celého akademického roku a počtu studentů podílejících se v tomto roce na realizaci projektu[3]. Výsledkem je, že jeden studento-měsíc má přibližně 13,75 produktivních hodin. Na základě množství plánovaných funkcionalit hry popsaných v dokumentu game

[2]Studento-měsíc – alternativa měrné jednotky člověko-měsíc (angl. man-month); udává přibližný produktivní čas jednoho studenta v jednom z jeho projektů za měsíc.

[3]Všechny hodnoty včetně postupu výpočtu jsou uvedeny v kapitole 7.5 Výsledky vedoucích projektu

designu je odhadováno, že pro dokončení projektu by mohlo být potřeba zhruba 350 ± 100 studento-měsíců.

Jediným zdrojem projektu jsou v současnosti pouze lidé (studenti). Finanční podpora nebyla dosud potřeba a není předpokládána ani do budoucna. Počet studentů zapojených do projektu i celá jeho organizační struktura se vlivem prostředí projektu (viz kapitola 4.7) neustále mění.

Co se týká rizik, mezi největší patří v projektu vývoje hry Space Traffic selhání při sestavování vývojového týmu, nezvládnutí role vedoucího projektu a případ, kdy se nové vedení začne pozdě seznamovat s projektem. Další rizika představí následující podkapitola 4.6.

Význam projektu spočívá především v účelu vytvářeného produktu, který je pro katedru KIV hodnotným přínosem. Význam má také pro studenty, kteří si vyzkoušejí spolupráci ve větším týmu, než jaký tvoří pro řešení některých semestrálních prací. V tomto směru má projekt hodnotu opět i pro katedru, která může jeho prostřednictvím lépe připravit studenty do praxe.

4.6 Rizika projektu

V každém projektu se vyskytují různě závažná rizika, neboli hrozby a jevy, kterým je projekt vystaven. Analýzou těchto rizik je zjišťována míra zranitelnosti projektu vůči těmto hrozbám, pravděpodobnost, že nastanou a jejich dopad na realizaci projektu. Pomocí různých metod a technik prevence rizik jsou odhalovány a eliminovány existující i budoucí faktory zvyšující tato rizika. V této oblasti je možné se setkat s množstvím standardů (ISO 16085:2006, ISO 31000, EIC/ISO 31010, atd), řídících rámců (RMF, M_o_R nebo RiskIT), metod a metodik (CRI, CRAMM, EWRM atd.) – ty jsou však již mimo rozsah této práce.

Existuje mnoho typů a členění rizik. V základu je možné rozlišovat mezi externími a interními riziky. Následující výčet uvádí externí rizika ohrožující dosažení cílů projektu.

1. Selžou nebo budou nedostupné dodávané služby.
2. Vzniknou problémy při komunikaci s dodavateli služeb.
3. Katedra KIV přestane projekt podporovat.
4. Vznikne konkurenční pro studenty zajímavější projekt.
5. Studentům budou zvýšeny nároky na studium.
6. O produkt bude nedostatečný zájem ze strany uživatelů.

Z uvedených rizik mají nejvyšší pravděpodobnost vzniku první dvě. Katedra KIV poskytuje v projektu služby pro provoz a správu nástrojů Redmine a Subversion (viz podkapitola 6.1 Technická podpora pro vývoj). Selhání při zajišťování provozu těchto nástrojů by mohlo v krajním případě způsobit ztrátu dat projektu.

Rizika 5. a 6. z výše uvedeného seznamu mají negativní vliv na zájem studentů o účastnění se projektu vývoje hry Space Traffic, tudíž i na složitost sestavování realizačního týmu a jeho udržení. Poslední uvedené riziko ohrožuje dosažení cílů přímo. Nezíská-li si hra své příznivce, přestane mít její vývoj význam.

Druhou skupinou rizik jsou rizika interní, tedy hrozby pocházející přímo z vnitřku projektu. Následující seznam obsahuje jejich výčet.

1. Nezkušenost manažera projektu.
2. Neúspěch při sestavování vývojového týmu.
3. Nedostatek pracovníků.
4. Nedostatečná kvalifikace pracovníků.
5. Pozdní seznámení nového vedení projektu s projektem samotným.
6. Odchod vedení projektu během realizace.
7. Nedostatečné testování.
8. Špatná komunikace mezi členy týmu.
9. Nezájem členů týmu na výsledku projektu.

Vzhledem k tomu, že katedra KIV nevychovává v žádném ze svých studijních programů vedoucí projektů, mají její studenti s touto činností jen velmi málo zkušeností. Je tedy vysoká pravděpodobnost, že student, který na období jednoho akademického roku zastává roli manažera projektu v rámci své diplomové práce, tuto roli nezvládne dostatečně zastat. Realizace projektu tím může být částečně ochromena, ne-li zcela pozastavena.

Stejný dopad na projekt má také další významné riziko, kterým je neúspěch při sestavování vývojového týmu. Protože není předpokládána dlouhodobá spolupráce studentů, nové vedení projektu sestavuje vývojový tým na začátku každého semestru. Musí být proto schopné správně zvolit způsob, jak oslovit jiné studenty a zaujmout je tak, aby se přihlásili o spolupráci.

S tímto rizikem pak souvisí další dvě – rizika 3. a 4. výše uvedeného seznamu. Nedostatek pracovníků i jejich nedostatečná kvalifikace je příčinou nesplnění stanoveného plánu. Maximální počet členů týmu je však omezen počtem a schopnostmi vedoucích projektu.

Každý rok je vedení projektu vystřídáno novým vedením, které se musí vždy po svém nástupu důkladně seznámit s projektem. Dalším velkým rizikem je, že tak učiní příliš pozdě a nebude schopné na začátku zimního semestru vhodně a v čas oslovit studenty, nabídnout jim spolupráci, sestavit z nich vývojový tým a ten pak vhodně vést. Takovýto scénář by jistě opět vedl k výraznému zpomalení vývoje. Toto riziko je ovlivněno postupně se rozvíjejícím produktem projektu – hrou Space Traffic. Se zvětšujícím se množstvím

zdrojových kódů se zvyšuje jak složitost řešení, tak i množství dokumentace a hlavně čas potřebný pro dostatečné seznámení s jeho stávajícím stavem.

Zbylá čtyři rizika mají buď relativně nízkou pravděpodobnost, že nastanou, nebo jejich dopad na cíle projektu není tak významný. Jejich interpretace je navíc zcela zřejmá, proto zde nebudou dále rozvedeny.

4.7 Charakteristika prostředí

Je třeba si uvědomit, že vývoj softwarového produktu probíhá v kontextu organizace, její organizační struktury, kulturních, společenských i politických vztahů; jak uvedl Jim McCarthy ve své publikaci *Softwarové projekty* [30]. Charakteristické znaky a z nich vyplývající problémy akademického prostředí, v němž je realizován projekt vývoje hry Space Traffic, budou předmětem této části.

4.7.1 Lidské zdroje

Jediným zdrojem v projektu jsou studenti, jejichž získávání pro spolupráci není jednoduchým úkolem. Studenti často nemají zájem se na projektu podílet, jak popisuje Zbyněk Neudert ve své diplomové práci [27]. Pokud se je podaří získat, nelze předpokládat, že budou ve spolupráci pokračovat i v následujícím semestru či akademickém roce.

Zbyněk Neudert věnoval získávání studentů pro projekt velké úsilí a vyzkoušel čtyři způsoby jejich oslovení:

- krátké prezentace na přednáškách,
- informativní letáky,
- zadání speciální semestrálních prací na předmětech KIV,
- vypsání stipendia,

nepodařilo se mu zajistit tak velký tým, jaký očekával. Ve své práci uvádí, že jím požadovaný stav by bylo devět členů řešitelského týmu, podařilo se mu však získat pouze jednoho, který s ním spolupracoval po celou dobu vývoje a dva další, kteří se na projektu podíleli méně než polovinu jednoho semestru.

Při hledání nových členů vývojového týmu v tomto akademickém roce bylo zjištěno, že studenti často zastávají následující názory:

- semestrální práce pro projekt Space Traffic bude obtížnější než standardní,
- pokud si vyberou nestandardní práci, nebudou se moci poradit s kamarády, protože ti budou řešit jiné zadání,

- řešení standardní semestrální práce mohou najít na internetu,
- aby se mohli efektivně zapojit do týmu, musí ovládat jazyk C#.

Studenti si také mnohdy nevěří nebo nemají odvahu se do nestandardního zadání pustit a proto argumentují slovy:

- *„Potřeboval bych mít nějaké znalosti a zkušenosti s vývojem her.“*
- *„Moje řešení nebude dostatečně kvalitní a tak se stejně zahodí.“*
- *„Když se mi práce nepovede, nebo ji vůbec nezvládnu, můžu tím ohrozit studium jiných, kteří v rámci projektu dělají svou bakalářskou nebo diplomovou práci.“*

Toto jsou velmi závažná zjištění komplikující zajišťování lidských zdrojů a je nutné řešit (viz kapitoly 5.5 Motivování studentů a 7.2 Získávání studentů).

Zaměříme-li se blíže na tento zdroj projektu, zjistíme, že nejdůležitější na tomto zdroji je čas studentů, který je z hlediska jednotlivého studenta omezen studentovým studijními povinnostmi. Množství času, jež student do projektu vkládá, je tak určeno jednak tím, kolik času věnuje běžné semestrální práci a jednak množstvím ostatních prací v daném semestru. Studenti se musí nejprve seznámit s projektem a ve většině případů také s technologií a naučit se s ní pracovat, než začnou být produktivní. Zbývající čas, který projektu věnují musí být co nejlépe využit.

4.7.2 Nestálý projektový tým

Studenti se na realizaci projektu podílejí většinou prostřednictvím semestrálních prací z různých předmětů. Předměty, a tudíž i semestrální práce, mají trvání jednoho semestru. Projektový tým je tak nutné nalézt, sestavit a zaučit dvakrát během jednoho akademického roku. Tuto činnost má za úkol vedení projektového týmu. Jeho zapojení do projektu je však také omezeno a to na jeden akademický rok. Z důvodu neustále se měnícího týmu hraje v projektu velkou roli dokumentace. Ta musí být každému členovi snadno přístupná, ale i zde se objevuje další podstatný znak, a tím je fakt, že studenti ji neradi čtou.

4.7.3 Studentský syndrom

Je léty prověřeno, že výkon studentů je na počátku výrazně nižší nebo žádný a postupem času, jak se blíží termín odevzdání, se exponenciálně zvyšuje. Tento efekt je nazýván *studentský syndrom*. Studenti mají během semestru obvykle více praktických semestrálních prací než jednu a zátěž na konci semestru je tudíž úměrná tomuto počtu. Důsledkem pak bývá pozdní odevzdání některé z prací. Studentský syndrom však není nic neobvyklého i ve firemním prostředí.

V prostředí projektu Space Traffic je tento syndrom pro studenty, kteří projekt řídí a kontrolují kvalitu odvedené práce, obzvlášť nepříjemný. I tito studenti totiž mají své studijní závazky určené jejich studijními plány a nedodržení termínů jejich kolegů z týmu je může dostat do stejných problémů. Potíž se studentským syndromem je možné řešit hned několika zcela odlišnými způsoby. Buď je možné se s ním smířit a využít ho ve svůj prospěch (viz 6.3 Podpora a vedení vývojového týmu), nebo se mu bránit, aby vůbec nevznikl (viz 5.5 Motivování studentů).

4.7.4 Nestálá pracovní doba

Na rozdíl od běžného firemního prostředí, kde je obvyklá osmihodinová pracovní doba, která je v průběhu realizace projektu stálá, tak v akademickém prostředí je všechno přesně naopak. Studenti na svých semestrálních pracích obvykle výrazně pracují až ke konci semestru. V celém průběhu semestru je jejich čas věnovaný projektu navíc značně nestálý, protože je ovlivněný jednak zápočtovými týdny a jednak termíny odevzdání průběžných zápočtových úloh z ostatních studovaných předmětů.

Jednotliví studenti mají sestaveny odlišné studijní plány tvořené různými předměty. Zaměřit se na každého jednoho studenta zvlášť a na základě jeho vlastního rozvrhu ho efektivně plánovat, je pro vedoucího studenta (popř. studenty), který projekt řídí časově nezvládnutelné.

5 Řízení projektu Space Traffic

V předchozí kapitole bylo ukázáno, jak je projekt strukturován a s jakými problémy se setkává. V této kapitole bude představen postup řízení projektu, který byl postupně vytvořen za účelem snížení rizik a problémů prostředí realizace projektu. Kapitola je tedy pojata jako popis aplikace dříve uvedených poznatků na reálném projektu vývoje hry Space Traffic. Volba postupů a nástrojů, použitých během realizace pro podporu projektu Space Traffic, byla podřízena možnostem výpomoci ze strany katedry, dále charakteristickým znakům prostředí projektu, účelu hry a stanoveným cílům.

Po přečtení následujícího textu bude čtenář vědět, jakým způsobem byl projekt řízen a jaké nástroje byly pro podporu jeho řízení použity. Autor zde nemůže z důvodu jedinečnosti každého projektu čtenáři poskytnout přesný návod, jak řídit obdobné softwarové projekty v prostředí podobného charakteru. Čtenář se však bude moci na základě uvedených informací inspirovat při volbě metodiky pro řízení vlastního softwarového projektu, stanovení způsobu jakým podchytit jednotlivé oblasti řízení a při řešení problémů plynoucích ze specifických znaků prostředí svého projektu.

5.1 Přípravná opatření projektu

V okamžiku, kdy autor této práce převzal společně s Martinem Štěpánkem projekt vývoje hry Space Traffic do svých rukou (viz 4.4 Historie projektu), byly zahájeny činnosti příprav projektu pro následující akademický rok 2011-2012. Účelem těchto příprav bylo nalézt a minimalizovat existující rizika (viz 4.6 Rizika projektu) a pracovat s nimi nebo je řešit co nejdříve, aby následná realizace probíhala co nejplynuleji. Z tohoto důvodu bylo usilováno o pokrytí přípravnými aktivitami co nejvíce rizikových oblastí projektu. Během příprav byl kladen důraz také na kvalitu vykonávaných činností, neboť ta je podle McConnella [26] příčinou úspěchu nebo selhání.

Ať se jedná o jakýkoli softwarový projekt, je důležité nejprve pochopit, k čemu bude jeho výsledný produkt určen a jak ho implementovat. Na počátku by tedy měl být jasně definován problém, který má produkt řešit. Poté by se měl projektový tým zaměřit na požadavky na produkt, což je prvním krokem k řešení problému, protože požadavky podrobně popisují, co má systém vlastně dělat. Dalším logickým krokem je návrh na nejvyšší úrovni abstrakce – specifikace architektury softwaru. Její kvalita pak určuje soudržnost celého systému.

Jak již bylo několikrát uvedeno, každý projekt je jedinečný. Různé projekty tedy vyžadují odlišný přístup i ke své přípravě. V rámci přípravných činností projektu vývoje hry Space Traffic byly definovány klíčové požadavky, vytvořen kvalitní základ architektury hry, připraveno provozní prostředí pro vývoj, analyzovány možnosti sestavení realizačního týmu

atd. Všechny kritické aspekty projektu, kterým byla v rámci příprav věnována pozornost, jsou uvedeny v následujícím textu.

Seznámení se s projektem

Na úplném začátku bylo nutné seznámit se s projektem. To znamenalo zjistit, v jakém stavu se nachází vytvářená hra, jaký je vůbec její účel, kteří členové katedry jsou s projektem spojeni, jaká je v projektu jejich úloha, jaké jsou cíle projektu a jeho historie. Takovéto seznámení umožní hlubší pochopení kontextu projektu.

Zahájení projektu

V okamžiku, kdy již byla známá základní specifika projektu, mohly být zjištěny klíčové požadavky zainteresovaných členů katedry. Tyto požadavky byly převážně technického charakteru, a to vytvořit kvalitní modulární[1] základ hry, pro implementaci serverové části použít programovací jazyk C# a zachovat programovatelné prvky hry.

Na základě těchto požadavků a dokumentu game designu byly stanoveny cíle (viz kapitola 4.1.1), rozsah (viz kapitola 4.5), a předběžný plán projektu pro následující akademický rok. Kromě toho byla identifikována hlavní velká rizika projektu vyplývající především ze specifických znaků akademického prostředí (viz kapitola 4.6).

Zajištění technického zázemí

Při výběru podpůrných nástrojů pro vývoj bylo nejprve zjištěno, jaké nástroje může poskytnout katedra a jaké bude muset zajistit vedoucí projektu (viz kapitola 5.3). Mezi základní nástroje patří vývojové prostředí (IDE), nástroje pro správu verzí, plánování, zadávání úkolů, reportování chyb, sdílení dokumentací a hlavně komunikaci.

V rámci těchto příprav byl také u katedry obstarán virtuální server s operačním systémem Windows Server 2008, na kterém později mohly být instalovány a provozovány zvolené nástroje. Více o technickém zázemí projektu je možné nalézt v kapitole 6.1.

Dalším předmětem příprav bylo vytvoření návodů s pokyny pro instalaci a nastavení pracovních nástrojů, které zajišťovaly snadné zprovoznění pracovního prostředí nových členů projektu.

Předběžný návrh architektury hry

V rámci příprav byl v letních měsících vytvořen koncept návrhu architektury (viz práce Martina Štěpánka [28]), který popsal samostatné subsystémy hry spolupracující na vyšší úrovni abstrakce.

[1] Modulární ve smyslu snadné rozšiřitelnosti o další funkcionalitu (viz práce M. Štěpánka [28]).

Sestavení realizačního týmu

Aby bylo možné projekt realizovat, bylo nezbytné sestavit vývojový tým. Na začátku zimního semestru tedy byla provedena analýza možnosti nabízet studentům spolupráci prostřednictvím speciálních zadání semestrálních prací v předmětech katedry. Bylo zkoumáno, jaké předměty jsou pro spolupráci svojí osnovou vhodné, co by bylo obsahem konkrétních zadání a jaká by byla jejich forma.

Na náš záměr byla od oslovených pedagogů získána kladná odezva. Dokonce jsme u jednoho z předmětů dostali nabídku k vytvoření standardního zadání pro všechny vyučované studenty. Jak se předpokládalo, vytvořené zadání musí splňovat několik podmínek, aby bylo pedagogem příslušného předmětu přijato. Základní podmínkou je, aby výsledný produkt zpracovaný v rámci speciální semestrální práce splňoval stejná kritéria jako standardní zadání. Další podmínkou je, že speciální zadání musí být formulováno ve stejném rozsahu a detailu jako zadání standardní a je studentům poskytnuto též stejnou formou.

5.2 Volba metodiky pro řízení vývoje

Účelem řízení vývoje softwaru je zajistit dosažení předpokládaného výsledku. Z tohoto důvodu se volí postupy (metodiky) řešící realizaci a řízení vykonávaných činností v projektu tak, aby byly co nejvíce přizpůsobeny jeho potřebám, prostředí i velikosti, zkušenostem a povaze týmu, dostupným technologiím i nástrojům. Těmto okolnostem je důležité při výběru věnovat pozornost zvláště proto, že použitá metodika má vždy významný dopad na následný průběhu celého projektu [16]. Výběr provádí na základě svých znalostí manažer projektu, kterému při rozhodování nejvíce napomůže dostatečné pochopení a zkušenosti s daným typem problému.

Protože každý projekt vyžaduje individuální přístup, nelze zvolit nějakou všeobecně oblíbenou metodiku, nastudovat ji, aplikovat ji na realizovaný softwarový projekt přesně tak, jak je popsána a věřit, že bude pro projekt tím nejlepším řešením. Také v případě projektu Space Traffic není tento přístup vhodný, a to už jen z toho důvodu, že celý projektový tým tvoří studenti, z nichž někteří se s řízením projektu nikdy nesetkali a někteří dokonce nemají žádné zkušenosti ani s prací v týmu. Kromě toho, studenti nemají pevně stanovenou pracovní dobu a na projektu se střídají (viz 4.7 Charakteristika prostředí). Hlavně však nemají čas a nebo se nechtějí při řešení semestrální práce učit cokoli jiného, než je nezbytně nutné pro její splnění. Studenti jsou obvykle striktně orientovaní na dosažení základních cílů, jakými jsou např. získání zápočtu nebo ušetřit co nejvíce času.

V rámci svých semestrálních prací mají studenti úvazek k projektu pouze na jeden semestr. Je tedy vhodné zvolit pro řízení vývoje jednoduchý postup, aby bylo minimalizováno úsilí (především náklady z hlediska času) vynaložené na zaučení nových členů, kteří se na projektu podílejí právě jeden semestr. Požadavek na jednoduchost zvolené metodiky posiluje také míra zralosti studentů (viz 6.3.2 Vedení týmu podle jeho zralosti) angažujících se v projektu.

Na základě výše uvedené charakteristiky prostředí a účastníků projektu Space Traffic bylo rozhodnuto, že z důvodu jejich nestandardních vlastností bude použita vhodná kombinace prvků několika jiných metodik tak, aby byla pro studenty rozumě jednoduchá a přitom neodporovala současným znalostem v oblasti moderního řízení vývoje softwaru.

Postupy, které byly pro řízení vývoje zvoleny, byly ve velké míře inspirovány všeobecně známou metodikou Scrum. Z této metodiky byl využit backlog – zásobník pro scénáře, funkční vlastnosti a úkoly, které mají být v budoucnu realizovány, ale nespadají do rozsahu plánované iterace. Ze stejné metodiky byla použita také myšlenka daily scrum schůzek. Náplň schůzek byla zachována, ale perioda jejich konání byla prodloužena na čtrnáct dní. Z metodiky FDD byla využita koncepce vývoje podle užitných vlastností. Dílčí vývojové týmy tak byly zodpovědné za jimi vytvářené přírůstky funkcionality a jejich výsledky vykazovaly viditelný pokrok.

Jelikož se nepředpokládá, že budou na první pokus sestaveny optimální postupy pro řízení projektu, budou na základě zkušeností s jejich použitím navrhována a aplikována vylepšení. Vhodná metodika se tak bude při realizaci postupně budovat.

5.2.1 Sekvenčně nebo iterativně

Již od začátku bylo jasné, že sekvenční způsob vývoje softwaru bude v případě projektu Space Traffic nevhodný (viz 3.5 Rozhodování při výběru metodiky). Jedná se totiž o dlouhodobý projekt, ve kterém se angažují studenti v rámci více různých předmětů i studijních programů. Požadavky na hru jsou nestálé a postupně se zpřesňují s rozvíjejícím se game designem. Kromě toho v projektu existuje také více závažných rizik (viz 4.6 Rizika projektu).

Studenti při řešení semestrálních prací obvykle nevykonávají před samotným programováním žádné přípravy (sběr požadavků, doménový model, návrh architektury atd.). Vytvářejí pak produkt, aniž mají celkovou představu o řešeném problému. Ukázalo se, že z důvodu nezkušenosti často nevědí, co by měli během příprav dělat, a proto při řešení postupují více méně chaoticky, bez příprav a co nevyčtou ze zadání si domyslí.

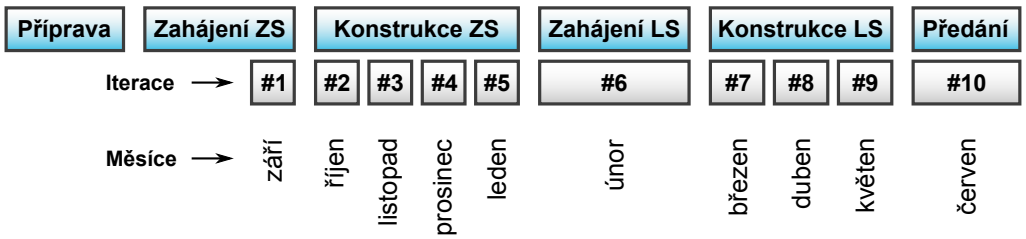
Iterativní způsob vývoje softwaru podle McConnella [26] zmírňuje dopad nedostatečných nebo nevhodných počátečních prací, jak ale McConnell dodává, zcela je eliminovat nedokáže. Bylo tedy rozhodnuto použít pro vývoj iterativní přístup. Určilo se, že iterace budou mít délku trvání jeden měsíc a budou time-boxované – délka iterace (stanovená na začátku projektu) je po celou dobu realizace projektu pro všechny iterace shodná.

5.2.2 Fáze vývojového cyklu

Během akademického roku prochází projekt několika fázemi, které byly přizpůsobeny pro spoluúčast studentů prostřednictvím jejich semestrálních prací. Těmito fázemi jsou *Příprava*, *Zahájení ZS* (zimní semestr), *Konstrukce ZS*, *Zahájení LS* (letní semestr), *Konstrukce LS* a *Předání*. Doba trvání každé fáze je striktně vymezena počtem iterací,

které ve fázi proběhnou, jak je naznačeno na obrázku 5.1.

Jak již bylo zmíněno, studenti jsou k projektu vázáni na jeden semestr na jehož konci odevzdávají řešení svého úkolu ke kontrole a ohodnocení. Tato skutečnost se ve vývojovém cyklu projevila zdvojením fází zahájení a konstrukce, protože v každém semestru jsou shodně prováděny činnosti plánování úkolů, vytváření, prezentování, vypracování, kontrolování a ohodnocování semestrálních prací. Je možné říci, že konec semestru je jakýmsi přirozeným milníkem, při jehož dosáhnutí lze sledovat nový přírůstek hry, který je otestován a zdokumentován.



Obrázek 5.1: Fáze vývojového cyklu během akademického roku.

V závislosti na fázi jsou v průběhu iterací vykonávány různé činnosti. V následujícím textu jsou fáze vývojového cyklu popsány a obrázek 5.2 ukazuje přehled vykonávaných činností s jejich přiřazením k jednotlivým fázím.

Příprava – tato fáze probíhá již v letních měsících od konce letního semestru. Jejím účelem je seznámit nové vedení s aktuálním stavem projektu tak, aby bylo možné na projektu začít pracovat. Seznámení s projektem zahrnuje především tyto úkoly:

- zjistit cíle projektu a účel hry,
- nastudovat způsob jeho řízení,
- prozkoumat používané nástroje,
- zjistit jaká část hry je již implementována.

Zahájení ZS – této fázi odpovídá jedna iterace probíhající v měsíci září. Jejím smyslem je vytvoření zadání semestrálních prací pro budoucí členy realizačního týmu a určení plánu pro nadcházejí akademický rok. Během této fáze jsou vykonány činnosti:

- určení plánu pro nadcházející akademický rok,
- stanovení cílů pro zimní semestr,
- analýza možnosti rozdělení plánované funkcionality do semestrálních prací,
- vytvoření zadání semestrálních prací.

Konstrukce ZS – této fázi odpovídají následující čtyři iterace, z nichž nejvíce produktivní jsou první dvě, zbylé se kryjí se zápočtovými týdny a zkouškovým obdobím. Jejím účelem je sestavení vývojového týmu, a naplnění cílů stanovených pro zimní semestr. Jsou prováděny činnosti:

- prezentace vytvořených zadání semestrálních prací,
- návrh, implementace, testování a dokumentování plánované funkcionality,
- ověřování správného návrhu a funkce mezivýsledků dílčích týmů,
- kontrola a ohodnocení výsledných produktů dílčích týmů.
- prezentace hry na dni otevřených dveří.

Zahájení LS – této fázi odpovídá jedna iterace probíhající v měsíci únoru a její náplň je podobná fázi *Zahájení ZS*. Jejím účelem je vytvoření zadání semestrálních prací pro budoucí členy realizačního týmu v letním semestru a určení cílů pro tento semestr. V jejím průběhu jsou provedeny následující činnosti:

- stanovení cílů pro letní semestr,
- analýza možnosti rozdělení plánované funkcionality do semestrálních prací,
- vytvoření zadání semestrálních prací.

Konstrukce LS – této fázi odpovídají následující tři iterace, z nichž nejvíce produktivní jsou opět jen první dvě, zbylá se kryje se zápočtovými týdny a zkouškovým obdobím. Jejím cílem je stejně jako v případě fáze *Konstrukce ZS* sestavení vývojového týmu, a naplnění cílů stanovených pro letní semestr. Během fáze jsou prováděny činnosti:

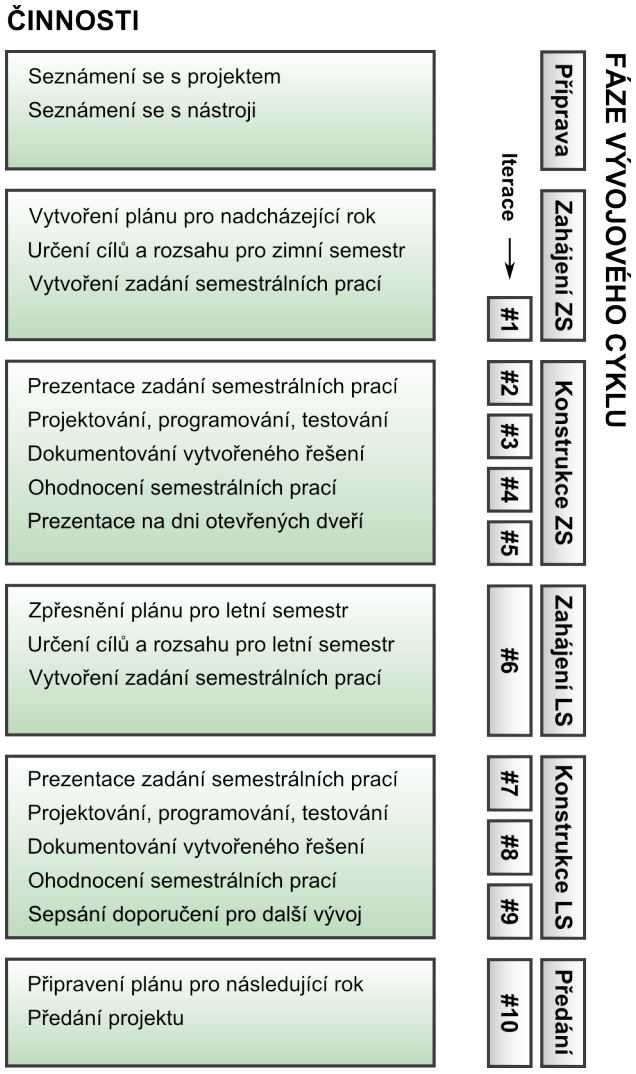
- prezentace vytvořených zadání semestrálních prací,
- návrh, implementace, testování a dokumentování plánované funkcionality,
- ověřování správného návrhu a funkce mezivýsledků dílčích týmů,
- kontrola a ohodnocení výsledných produktů dílčích týmů.
- sepsání doporučení pro další vývoj.

Předání projektu – této fázi odpovídá jedna iterace probíhající v měsíci červnu. Jejím smyslem je předání projektu následovníkům, kteří projekt povedou v následujícím akademickém roce. Během fáze jsou prováděny činnosti:

- připravení konceptu plánu pro následující rok,
- administrativa spojená s předáním projektu (předání hesel, změna vlastníka projektu v nástroji Redmine atd.)

U některých vyučovaných předmětů je v rámci semestrální práce vyžadováno použít při jejím řešení konkrétní metodiku vývoje softwaru. V takovém případě nejsou členové dílčího vývojového týmu vedeni přímo vedoucími projektu Space Traffic. Místo toho si mezi sebou rozdělí role odpovídající použité metodice a určí si svého zástupce, který je vede a zodpovídá za jimi vykonanou práci.

Spolupráce pak ve skutečnosti probíhá formou zakázky, kdy vedoucí projektu Space Traffic zadají pro celý dílčí tým úkol a termín jeho splnění. Vedoucí dílčího týmu následně úkol rozloží na individuální úkoly pro jednotlivé členy svého týmu. Podle požadavků použité



Obrázek 5.2: Činnosti prováděné během vývojového cyklu.

metodiky je také nutné, aby si vedoucí dílčího týmu přizpůsobil iterace #2 a #3 (viz obr. 5.1). Ve většině případů je přidělený čas dvou měsíců rozdělen do pěti dvoutýdenních iterací.

Díky takto volně navržené konstrukční fázi je možné řídit a vést dílčí vývojové týmy i jednotlivce v závislosti na míře jejich zralosti (viz kapitola 6.3.2 Vedení týmu podle jeho zralosti).

5.3 Výběr nástroje pro podporu řízení

Řízení projektů je doprovázeno nástroji, které manažerům projektu usnadňují jejich činnost. Tyto nástroje mají za úkol zefektivnit a zpřesnit řízení jednotlivých řídících oblastí (viz 3.3 Obecně o řízení projektů) a zpřehlednit průběh realizace projektu. Projekt je potřeba spravovat způsobem odpovídajícím rozsahu a prostředí jeho realizace. Proto se volí nástroj, který nabízí takové funkce, které vyhoví jeho potřebám.

Jednodušší nástroje pro podporu řízení obvykle umožňují evidenci projektů, tvorbu Ganttova diagramu a zjednodušenou správu zdrojů projektu. Složitější pak bývají součástí systémů ERP (Enterprise Resource Planing) a obsahují funkce koordinace soustavy projektů, řízení rizik, finančního plánování, plánování a optimalizace kapacit atd. a jsou určené pro velké organizace a podniky.

V oblasti řízení softwarových projektů je možné využít jak nástrojů, které jsou zaměřeny na obecné projektové řízení, tak nástrojů, které jsou více či méně přizpůsobeny konkrétním metodikám vývoje. Nejčastěji se setkáváme s nástroji optimalizovanými pro metodiky Scrum, UP, případně RUP. Společnou vlastností všech těchto nástrojů je, že umožňují spravovat projekty a dohlížet na plnění dílčích úkolů, což je přínosné zejména u rozsáhlejších a komplikovanějších projektů s více účastníky.

Pro podporu řízení projektu Space Traffic byl zvolen nástroj Redmine[2], který je provozován a spravován katedrou KIV. Redmine je webovou aplikací pro řízení softwarových projektů a je využíván studenty při řešení jejich týmových semestrálních prací. Jedním z hlavních důvodů výběru tohoto nástroje bylo právě to, že s ním již někteří studenti mají zkušenost a nemusejí věnovat čas k jeho zvládnutí.

Tento nástroj poskytuje uživatelské rozhraní v českém jazyce, umožňuje správu projektů v hierarchickém uspořádání, flexibilní správu přístupu uživatelů podle jejich rolí, oznamování událostí emailem a další funkce, které jsou uvedeny v následujícím seznamu.

Pro řízení činností umožňuje

- vytvářet a plánovat úkoly,
- členit úkoly do logické hierarchie s vazbami mezi nimi,

[2]Dostupný z URL http://students.kiv.zcu.cz:3000.

- vytvářet Ganttovy diagramy
- a poskytuje kalendář integrující rozpis úkolů.

Pro řízení zdrojů umožňuje

- sledovat čas strávený jednotlivými členy týmu na projektu,
- správu lidských zdrojů
- a přidělovat lidi k úkolům.

Pro podporu spolupráce umožňuje

- vkládat a sdílet dokumenty,
- integrovat nástroj pro správu verzí,
- vytvořit pro každý projekt samostatné fórum a wiki

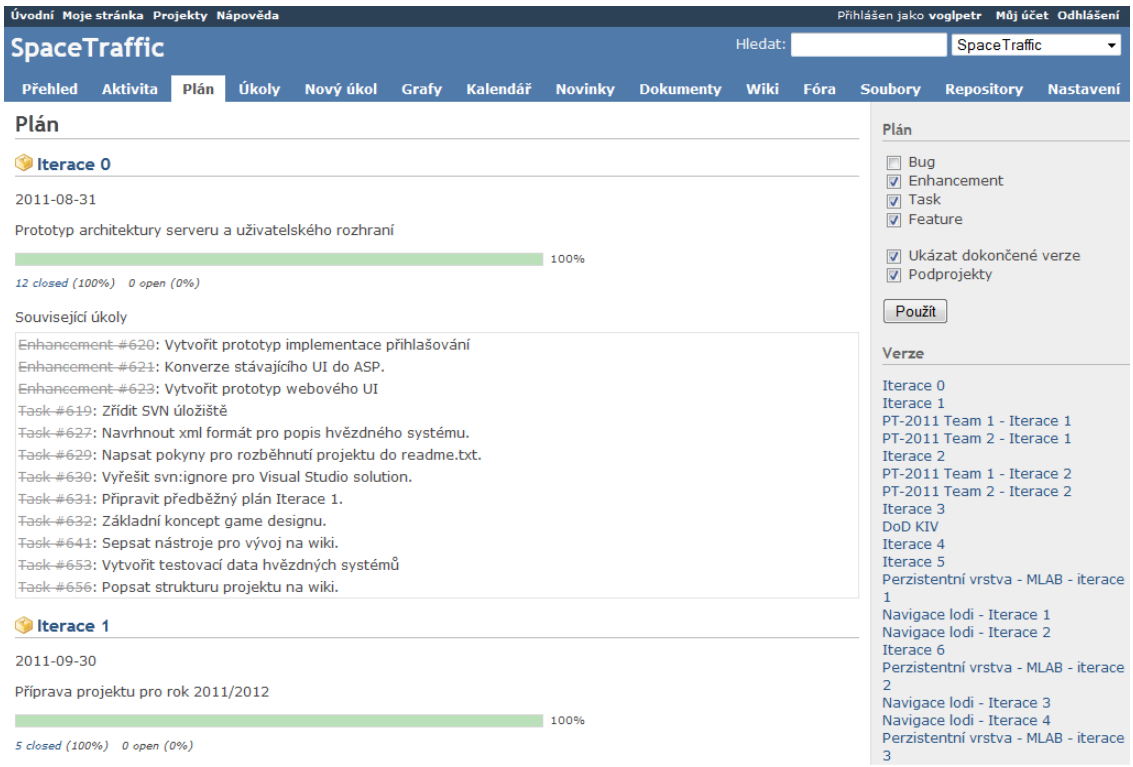
Redmine není nijak složitý na ovládání, jeho používání by měl zvládnout každý student a na práci v týmu o pár lidech poměrně bohatě dostačuje. Pro lepší představu o této aplikaci je na obrázku 5.3 ukázán screenshot jejího uživatelského rozhraní.

5.4 Plánování a zadávání úkolů

Nedílnou součástí řízení projektů je vytváření úkolů, jejich následné plánování a přidělování jednotlivým členům týmu. Před samotným vytvářením a plánováním úkolů však musí být jasné, jakých cílů se má v dané iteraci dosáhnout. Na začátku iterací byly proto stanoveny jejich cíle s následným zapsáním do wiki v nástroji Redmine. Po skončení každé iterace bylo do této wiki zaznamenáno zhodnocení splnění stanovených cílů.

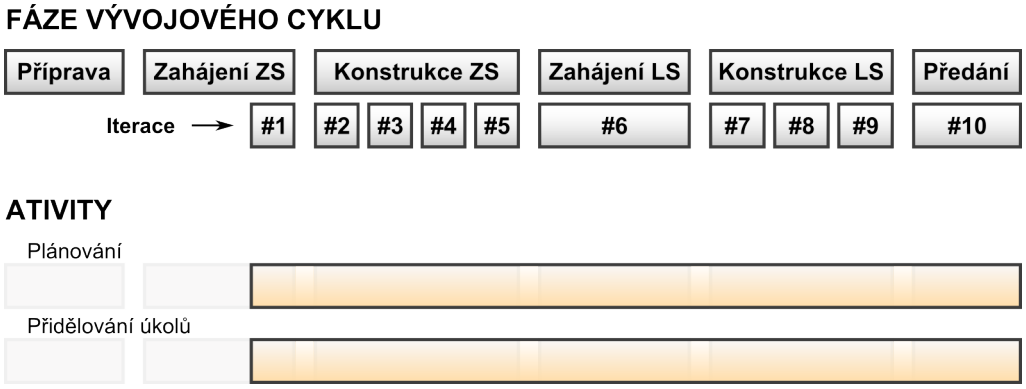
Podle určených cílů byla na začátku každé iterace vytvořena a v nástroji Redmine dané iteraci přiřazena základní sada úkolů, která byla v průběhu iterace rozšiřována o doplňující úkoly. Jednotliví členové týmu si sobě úkoly přiřazovali sami, nebo jim byly přidělovány průběžně na schůzkách a byly s nimi konzultovány. Tento přístup byl použit u jednotlivců, u nichž z důvodu malého rozsahu jejich přičinění nemělo smysl vytvářet v nástroji Redmine podřízený projekt.

V případě dílčích týmů, a to nejen těch, které při vývoji používaly z výše popsaných důvodů jinou metodiku, byl v nástroji Redmine vytvořen podřízený projekt, jehož rozsahem byla týmu zadaná semestrální práce a cílem bylo vytvoření funkčního a otestovaného produktu odpovídajícího zadání. Dílčí týmy pak byly schopné v nástroji Redmine vytvářet vlastní iterace odpovídající jimi použité metodice. Na základě konzultací svých semestrálních prací s jejich zadavateli (vedením projektu Space Traffic) týmy plánovaly vlastní



Obrázek 5.3: Screenshot aplikace Redmine – plán projektu.

úkoly. Plánování a přidělování úkolů je prováděno během celého semestru, jak ukazuje obrázek 5.4.



Obrázek 5.4: Aktivity plánování a zadávání úkolů ve vývojovém cyklu.

Při vytváření plánů nových iterací byl použit backlog. Pracovní položky (úkoly) byly při plánování nové iterace z backlogu vybírány a přesouvány do této iterace v závislosti na tom, zdali spadaly do jejího plánovaného rozsahu a cílů. Nástroj Redmine však nemá pro backlog podporu. Byl tedy vytvořen jako samostatná iterace a nepodporuje tak řazení obsahovaných položek.

Rozsah a cíle iterací byly určeny cíli stanovenými pro celý akademický rok. Cíle pro rok 2011-2012 je možné nalézt v příloze B společně s předběžnými cíli pro následující rok 2012-2013.

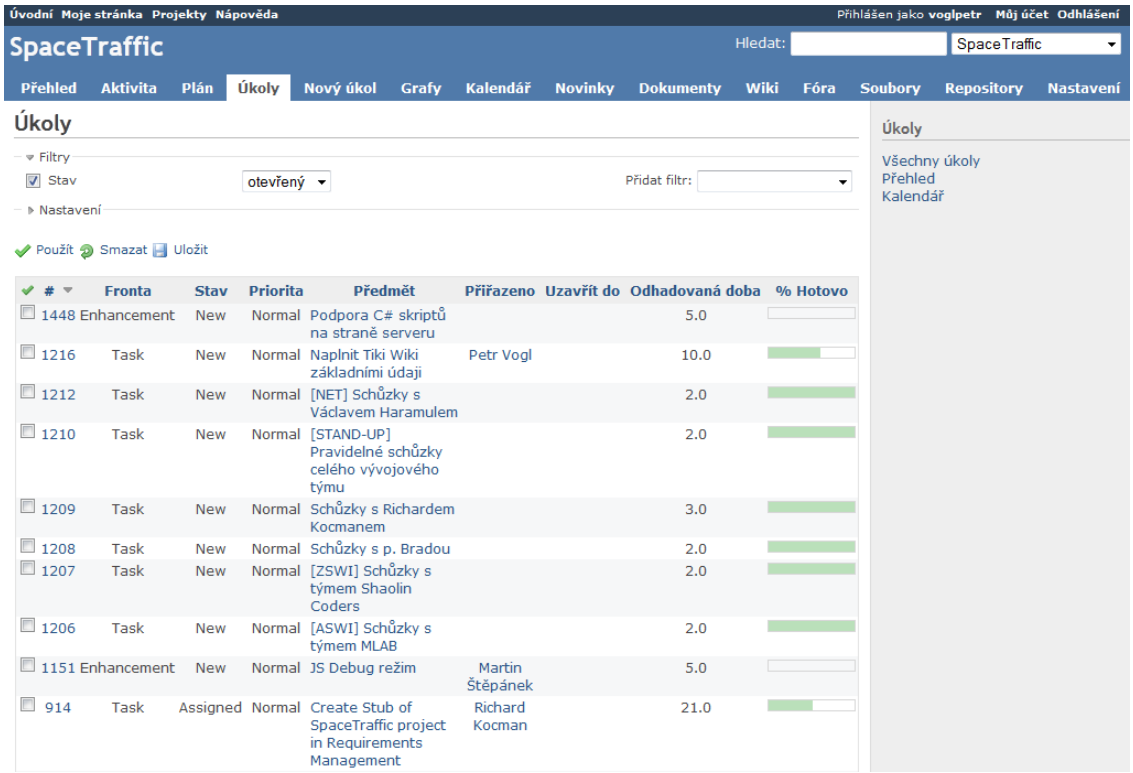
Pro ilustraci plánování úkolů je na obrázku 5.5 ukázán screenshot aplikace Redmine, na kterém je vidět seznam úkolů a k některým také přiřazení studenti.

5.5 Motivování studentů

Zřejmě nejobtížnější podpůrnou technikou pro vedení týmů je motivace. Při vedení projektu Space Traffic bylo potřeba studenty motivovat především k výběru speciálních zadání semestrálních prací nabízejících spolupráci v projektu. Jakkoli je však složité a těžké vhodně motivovat, tak zvládnutí této oblasti řeší většinu problémů týkajících se projektového týmu.

Akademické prostředí postihuje kreditním ohodnocením studentů pouze jejich hmotnou motivaci. Z důvodu vysoké časové náročnosti studia v oboru informatiky se nelze spolehnout, že studenti se budou na projektu podílet nad rámec svého studia. Proto bylo tohoto způsobu motivace využito tak, že spolupráce byla nabízena prostřednictvím speciálních semestrálních prací vyučovaných předmětů.

Byla také snaha zajistit i jejich nehmotnou motivaci, tedy vzbudit u studentů zájem o účast ve vývojovém týmu. Proto bylo budováno přátelské prostředí, aby se odstranila



Obrázek 5.5: Screenshot aplikace Redmine – seznam úkolů.

přirozená bariéra mezi vedoucími projektu (zadavateli semestrálních prací) a dílčími vývojovými týmy i jednotlivci (řešiteli zadaných prací). Bylo tak usilováno o to, aby v projektu spolupracující studenti neměli zábrany při komunikaci s vedoucími projektu, brali je také jako studenty a řešení svých prací s nimi konzultovali ve vyšší míře než v případě, kdy zadavatelem je pedagog.

Při prezentacích projektu a jednotlivých zadání bylo poukazováno hlavně na to, že vytvářeným produktem je MMORST webová hra, a že celý vývojový tým je tvořen studenty. Studenti byli dále lákáni na získání zkušeností s vývojem ve větším týmu, nástroji pro jeho podporu a s použitými technologiemi. Aby studenti chápali speciální zadání jako výhodu proti standardnímu, bylo na prezentacích zdůrazněno:

- Konzultace zadání, řešení i možných problémů lze provést kdykoliv prostřednictvím komunikátoru Skype.
- Se zadavateli práce je možné se sejít kdykoliv i mimo pravidelné schůzky.
- Při řešení práce vedoucí projektu průběžně pomáhají, aby byl výsledek kvalitní.
- Do projektu zapojení studenti získají zkušenosti od starších kolegů.

Kromě motivace k výběru zadání je vhodné motivovat i v průběhu realizace k lepšímu výkony a to jak jednotlivé členy týmu, tak i dílčí vývojové týmy. Během schůzek byli jedinci zdravě chváleni za dobře odvedenou práci. Občas jim bylo sděleno, aby byla povzbuzena jejich soutěživost a byly získány lepší výsledky, že jejich řešení je dobré, ale že by mohlo být i lepší, že druhý tým (popř. jednotlivec) je na tom lépe.

6 Podpora ekosystému projektu

Účelem této práce bylo vedle řízení projektu Space Traffic také zajištění dostatečné podpory jeho ekosystému během vývoje. Pod pojmem *ekosystém projektu* si lze v analogii s biologickým ekvivalentem představit komunitu účastníků projektu v součinnosti s technologiemi v kontextu organizace, která projekt realizuje. Cíle podpory pro ekosystém projektu pak vzhledem k uvedené definici byly: zajistit zdroje a technické zázemí projektu, vytvořit dostatečnou podporu pro komunitu projektu, usnadnit vývoj produktu a napomoci dosažení kvalitního výsledku.

Při psaní této práce bylo rozhodnuto rozdělit prostředky pro splnění cílů podpory ekosystému projektu do tří skupin podle toho, na jakou část ekosystému je zaměřen jejich podpůrný efekt. V následujících podkapitolách, odpovídajících těmto skupinám, jsou uvedeny postupy a nástroje, které byly pro splnění cílů podpory projektu použity.

6.1 Technická podpora pro vývoj

V této části budou představeny všechny nástroje, které z pohledu podpory vlastního vývoje usnadňují dosažení cílů projektu. Následující text se tak bude zabývat jak podporou řízení, plánování a úkolování, tak provozováním zvolených nástrojů, vývojovým a testovacím prostředím. Nebude opomenuta ani technická podpora pro distribuci informací a komunikaci.

Uvedený výčet nástrojů tvoří jádro technického zázemí ekosystému projektu. Každý člen vývojového týmu se s uvedenými nástroji přímo, či nepřímo setká:

- Redmine
- MS Visual Studio 2011
- MS Windows Server 2008
- Subversion
- Tiki Wiki CMS Groupware
- Google Groups
- Skype

V následujících dvou podkapitolách je uveden význam použití výše uvedených nástrojů.

6.1.1 Řízení projektu, vývoj hry a provoz nástrojů

Tato podkapitola uvádí význam použití aplikací Redmine, MS Visual Studio 2011, Subversion a MS Windows Server 2008 v projektu vývoje hry Space Taffic. Jedná se o nástroje umožňující řízení projektu, vývoj hry, správu verzí usnadňující spolupráci a prostředí umožňující testování a provoz dalších nástrojů. Sestava takovýchto nástrojů je minimální možnou technickou podporou. Z následujících odstavců je patrné, že pro úspěšnou realizaci projektu Space Taffic s vlastnostmi uvedenými v kapitole 4 musí mít každý nástroj této minimální sestavy v projektu své zastoupení.

Pro podporu řízení projektu, plánování a přidělování úkolů byla zvolena webová aplikace Redmine[1]. Redmine byl vybrán především proto, že někteří studenti z vyšších ročníků s ním již mají zkušenost. Jak již bylo uvedeno v kapitole 5.3 Výběr nástroje pro podporu řízení, Redmine je provozován katedrou informatiky a výpočetní techniky a poskytován studentům pro potřeby řízení jejich semestrálních projektů. Výhodou je, že přístup k projektu mohou novým členům týmu udělit sami vedoucí studenti. Není tedy nutné nikde žádat o autorizaci nových členů.

Redmine byl používán předně pro plánování a přidělování úkolů a pro sdílení informací týkajících se konfigurace pracovního prostředí vývojářů prostřednictvím integrované wiki. Kromě toho plnila wiki v tomto nástroji funkci rozcestníku ke zdrojům dalších informací převážně návodům a tutoriálům. S výhodou možnosti hierarchického uspořádání podprojektů dílčích vývojových týmů a tedy oddělení jejich vlastního plánování byla udržena přehlednost v globálním plánu projektu.

Z pohledu samotného vývoje hry je nejvýznamnějším nástrojem vývojové prostředí. Bylo zvoleno Visual Studio 2010, které je k dispozici všem studentům výše zmíněné katedry prostřednictvím programu MSDN (Microsoft Developer Network) Academic Alliance. Toto vývojové prostředí poskytuje nejlepší podporu pro vývoj nejen webových aplikací v jazyce C#. Kromě základních funkcí, které podporuje většina vývojových prostředí, jako jsou např. debuging nebo podpora Subversion, poskytuje také podporu pro lokální testování webových aplikací spouštěných prostřednictvím webového serveru IIS i podporu pro databázový server MS SQL Server.

Pro správu verzí nejen zdrojových kódů ale i všech digitálních zdrojů hry[2] a usnadnění spolupráce při vývoji byl zvolen nástroj Subversion[3]. V dnešní době je možné ke stejnému účelu využít i nástroje, jako jsou Git nebo Mercurial, ale jejich správa by musela být zajištěna vedoucími studenty, nebo by bylo možné využít neplacených služeb poskytovaných prostřednictvím internetu externími společnostmi. Subversion byl zvolen ze stejných důvodů, z jakých byl zvolen nástroj Redmine.

Subversion je spravován administrátory katedry KIV, repositáře jsou poskytovány pro potřeby studentů při řešení jejich semestrálních prací. Přístup k úložišti je umožněn prostřednictvím jednotného univerzitního přihlašování Orion a udělování přístupu je tak

[1]Projekt je přístupný z URL http://students.kiv.zcu.cz:3000/projects/space-traffic.

[2]Jedná se o soubory obsahu hry (angl. Game Assets) např. texty, obrázky, zvuky, videa nebo konfigurace.

[3]Repositář projektu je dostupný z URL svn+ssh://students.kiv.zcu.cz/home/subversion/diplomka/spacetraffic.

možné pouze prostřednictvím administrátorů katedry.

Vedle výše uvedených nástrojů je neméně důležité zajistit prostředí pro otestování hry samotnými hráči a pro potřeby prezentací průběžných výsledků projektu. Za tímto účelem byl katedrou poskytnut virtualizovaný Windows Server 2008 a k němu administrátorský přístup. Tento server byl použit také pro provoz všech dalších nástrojů, které nebyly poskytovány katedrou, nebo jinými subjekty.

6.1.2 Usnadnění komunikace a distribuce informací

Pro potřeby běžné komunikace během vývoje byl zvolen nástroj Skype. Jeho prostřednictvím byly řešeny jak akutní problémy, otázky návrhu a implementace řešení, tak domlouvání osobních schůzek, pomoc s ovládnutím používaných nástrojů i online konference. Skype se tak stal prostředkem pro okamžité, nebo alespoň rychlé, získání potřebných informací.

V případě potřeby sdělit informaci hromadně celému týmu nebo jeho větší části byla používána e-mailová skupina založená prostřednictvím služby Google Groups[4]. Touto cestou byly rozesílány důležité a naléhavé informace, se kterými nebylo možné počkat do následující schůzky, nebo byly určeny všem členům vývojového týmu. Tato sdělení měla většinou organizační charakter, nebo informovala o zásadních změnách v návrhu, implementaci nebo konfiguraci, které ovlivňovaly práci více členů týmu.

Pro podporu snadného vytváření a sdílení dokumentací a jiných dokumentů byl na začátku realizace projektu provizorně použit wiki nástroj DokuWiki[5]. Jeho prostřednictvím byl mezi členy týmu sdílen dokument game designu a dlouhodobý plán projektu. V průběhu realizace projektu se pak hledalo z důvodu problémového běhu DokuWiki v prostředí Windows Serveru jiné řešení.

Nové řešení přinesl semestrální projekt zadaný pro předmět Znalostní a informační management. Zadáním a současně cílem projektu bylo provést analýzu a doporučit nástroje pro sdílení znalostí a informací (tj. dokumentací a návodů) vhodných pro nasazení v projektu Space Traffic. Pro potřebu tvorby a sdílení dokumentů a dokumentací byl navržen wiki nástroj Daisy[6] vytvořený v jazyce Java společností Outerthought. Průběh jeho nasazování byl však komplikovaný mnoha problémy a to i přes přesné dodržení postupu instalace poskytovaného výrobcem[7]. Po bezúspěšném snažení bylo rozhodnuto nalézt jiné řešení.

Protože již nebyl k dispozici dostatek času na důsledný výběr alternativy, byl pomocí webové aplikace WikiMatrix [31], která umožňuje srovnávání wiki nástrojů, zvolen nástroj Tiki Wiki CMS Groupware[8]. Výhodami tohoto řešení je stálý vývoj nástroje, dobrá technická podpora a podpora jeho instalace prostřednictvím instalační služby webové plat-

[4]E-mailová adresa skupiny je spacetraffic-dev@googlegroups.com.
[5]Dočasně přístupný z URL http://spacetraffic.kiv.zcu.cz/dokuwiki.
[6]Webové stránky projektu jsou dostupné z URL http://www.daisycms.org/daisy/index.html.
[7]Návod na instalaci je dostupný z URL http://docs.outerthought.org/daisy-docs-2_1/13-daisy.html.
[8]Nasazený nástroj je dostupný z URL http://spacetraffic.kiv.zcu.cz/code.

formy Windows (Web Platform Installer). Nasazení a konfigurace tohoto nástroje trvala jeden den a během jeho používání se dosud neobjevily žádné problémy.

Tiki nabízí kromě tvorby obsahu formou wiki také fóra, blogy, souborové a obrázkové galerie, kalendář a události. Z administračního hlediska nabízí správu uživatelů, uživatelských skupin, řízení oprávnění k různým činnostem atd. Vzhledem k tomu, že tento nástroj poskytuje všechny funkce, které byly požadovány, a neobjevily se u něj zatím žádné problémy, je možné u něj zůstat.

6.2 Podpora komunity vytvořené kolem projektu

Jak se postupně vyvíjí produkt hry, tak se kolem projektu utváří komunita lidí, která má zájem získat o projektu nějaké informace. S přibývajícím počtem prezentací na přednáškách vybraných předmětů se zvyšuje množství studentů, kteří jsou s projektem blíže seznámeni. Na tyto studenty je žádoucí se dívat jako na potenciální budoucí členy vývojového týmu a podle toho jim co možná nejvíce usnadnit získání dalších informací o projektu.

Při příležitosti konání akce Den otevřených dveří jsou veřejnosti prezentovány výsledky vývoje. S projektem se tak seznámí i lidé mimo akademické prostředí (většinou studenti středních škol) a má-li být projekt jedním z důvodů, proč se přihlásí ke studiu na katedře KIV, měla by pro ně být také zajištěna dostatečná podpora.

V každém případě je vhodné se snažit o rozšiřování povědomí o projektu jak mezi lidmi z akademického prostředí, tak z veřejnosti. Následující seznam uvádí způsoby, jaké byly v projektu navrženy pro zajištění podpory vytvářející se komunity:

- prezentace projektu a hry na dni otevřených dveří,
- prezentace projektu a hry veřejnosti na sociálních sítích,
- prezentace projektu studentům na přednáškách vybraných předmětů,
- informativní plakáty v prostorách katedry,
- zpřístupnění některých částí vývojářské wiki veřejnosti,
- wiki určená pouze pro hráče,
- vývojářský blog,
- webové stránky produktu.

Vytvářející se komunita kolem projektu může být tvořena dvěma skupinami lidí s odlišnými zájmy. Jedni, kteří se zajímají o hru z pohledu hráče, a druzí, které zajímá spíše způsob řízení vývoje, použité technologie, architektonický návrh apod. Podporou obou skupin se zabývají následující podkapitoly.

6.2.1 Podpora pro komunitu hráčů

Aby bylo možné splnit hlavní cíl projektu – zvýšení zájmu středoškolských studentů o studium, musí se tato cílová skupina o projektu nejprve dozvědět. K tomuto účelu slouží prezentace projektu na akci dne otevřených dveří, jak bylo zmíněno již dříve. Tato akce se však koná pouze jednou do roka a není tak pro rozšíření povědomí o projektu dostačující. Z tohoto důvodu bylo navrženo podpořit šíření informací o projektu a hře Space Traffic prostřednictvím internetu pomocí populárních sociálních sítí Facebook a Twitter.

Tato online média jsou oblíbená především u mladých lidí, kteří jsou rádi součástí komunity s podobnými zájmy. Až se bude hra nacházet ve hratelném stavu, mohly by sociální sítě také pomoci získávat nové hráče, mezi kterými se jistě budou nacházet i lidé z cílové skupiny středoškolských studentů. Nesmí se ale zapomenout na to, že sociální sítě (resp. všechna sociální média) nejsou určena primárně k propagaci nebo reklamě, ale jejich úlohou je podpora komunikace ve vytvářených komunitách.

Jsou-li sociální sítě využívány pro marketing, základem by měla být právě komunikace s komunitou zákazníků. Proti tradičním médiím poskytují sociální média komunikaci obousměrnou (tzn. i ve směru od zákazníka k firmě). Komunita totiž sdílí hodnotné informace, zkušenosti a postřehy, které jejím členům pomáhají při rozhodování. Tato média jsou velmi oblíbená a lidé uvedeným informacím důvěřují, protože kolektivní názor je většinou pravdivý. Doporučení komunity pak šetří jejich čas. Komunita má tímto napomoci rozšířit povědomí o firmě a poskytovat zpětnou vazbu na vytvářené produkty tak, aby lépe vyhovovaly zákazníkům.

V případě hry Space Traffic, byly založeny účty na sítích Facebook[9] i Twitter[10], aby v budoucnu sloužily jako zdroj informací o projektu a jako komunikační kanály s komunitou fanoušků hry a jejího vývoje. Jejich prostřednictvím bude možné na základě mnohých hodnocení, hlasování o nejrůznějších záležitostech a uvedených komentářů dělat různá vyhodnocení či rozhodnutí. Výhodou je, že lidé mohou na tomto místě diskutovat o čem chtějí a vedení týmu se může diskusemi inspirovat pro další vývoj.

Dalším, v dnešní době již standardním podpůrným nástrojem pro hry, je wiki určená výhradně pro hráče. Hráči na tomto místě mezi sebou obvykle sdílí různé herní tipy a triky, uvádějí strategie, popisují všechny komodity a jednotky ve hře. Pokud je ve hře nalezena chyba zvýhodňující nějaký postup hráče, brzy se ve wiki objevuje v podobě cheatu. V případě vytvářené hry Space Traffic by se na tomto místě mohly objevovat také výhodné způsoby programování, programy pro řízení jednotlivých lodí, nebo celých flotil a jiné programové triky.

Součástí podpory hráčů by měl být také web produktu, ze kterého by bylo možné hru spustit a hrát. Na tomto webu by hráči měli dostupné veškeré informace o hře včetně odpovědí na často kladené dotazy. Součástí webu by byla také výše zmíněná wiki a fórum pro hráče. Na webech herních produktů se také často nacházejí nejrůznější žebříčky a statistiky, soutěže, hráčská videa, novinky, změny ve hře a neodmyslitelně také prezentace

[9]Stránka je dostupná z URL http://www.facebook.com/pages/Space-Traffic/313114725442658.
[10]Účet je dostupný z URL https://twitter.com/#!/SpaceTrafficKIV.

produktu a reklama.

Pro nasazení hráčské wiki a webu hry bylo zatím zajištěno pouze prostředí pro jejich provoz – výše uvedený Windows Server 2008. Hra zatím není ve stavu schopném hraní (je implementováno pouze její jádro a část grafického rozhraní) a během akademického roku již nezbyla kapacita pro řízení dalšího dílčího týmu, který by zajistil výběr vhodného redakčního systému a wiki, jejich integraci, konfiguraci a vytvoření obsahu.

6.2.2 Podpora pro příznivce vývoje hry

Kromě hráčů mohou být součástí komunity vytvářející se kolem projektu i lidé, které kromě hraní zajímá také způsob řízení vývoje, použité technologie nebo třeba návrh její architektury. Z hlediska podpory této skupiny bylo rozhodnuto o vytvoření dvou zdrojů informací tohoto druhu – vývojářský blog a wiki.

V dnešní době jsou stále velmi užitečným a populárním zdrojem informací blogy. Bylo tedy rozhodnuto o vybudování blogu vývojářů, který by průběžně informoval o postupech při vývoji hry. Záleží však na odhodlání a čase studentů – vývojářů, jestli budou mít zájem publikovat příspěvky.

Hlavním informačním zdrojem pro tuto skupinu příznivců projektu byla zvolena vývojářská Tiki wiki, kterou vývojový tým používá pro sdílení dokumentací a jiných informací v projektu. V této wiki byly veřejnosti zpřístupněny některé části společně s vývojářským blogem, který je její součástí.

Příznivci vývoje se předpokládají především v prostředí katedry informatiky a výpočetní techniky, proto jsou ve vývojářské wiki uvedeny také pokyny jak postupovat, pokud se chce student stát členem vývojového týmu. Získávání nových vývojářů je podpořeno také plakáty, vyvěšenými v prostorách katedry.

6.3 Podpora a vedení vývojového týmu

Předchozí dvě podkapitoly 6.1 a 6.2 se zabývali využitím nástrojů pro podporu ekosystému projektu. Tato kapitola se soustředí především na práci s lidmi – další aktivity, které ovlivňují jednak kvalitu a množství odvedené práce a jednak spokojenost členů během vývoje. Činnosti které byly v projektu Space Traffic prováděny s cílem zajistit kvalitnější výsledky a spokojenost týmu, jsou uvedeny v následujícím seznamu:

- vytváření návodů pro konfiguraci používaných nástrojů,
- zasvěcení týmu do projektu a vysvětlení práce s nástroji,
- dohled nad zapojením jednotlivých členů,
- průběžné kontrolování návrhu i programové implementace,

- revize vytvořených specifikací a dokumentací,
- poskytování rad a ukazování správného směru,
- řešení problémů,
- přijetí a ohodnocení výsledných produktů dílčích týmů,
- časté osobní schůzky,
- stálá komunikace.

Díky spolupráci na projektu získávali členové týmu cenné zkušenosti a znalosti. Jejich vyučení vždy stálo mnoho času vedoucích, kteří často zastávali roli vyučujících. Cílem podpory týmu se tak stalo udržet stávající členy i pro další vývoj v příštích semestrech. Následující podkapitoly popisují jakým způsobem byla tato podpora zajišťována a jak bylo přistupováno k vedení jednotlivců i celých dílčích týmů.

6.3.1 Podpora vývojového týmu

V této části budou uvedeny aktivity pro podporu týmu, které byly prováděny za účelem dosažení spokojenosti jak na straně vedení projektu (tzn. kvalitně odvedená práce), tak na straně vývojářů (tzn. včasné řešení problémů týkajících se vývoje a spravedlivé ohodnocení jimi vykonané práce). Postupně tak bude představen celý průběh spolupráce se členy týmu během semestru.

Předmětem příprav bylo vytvoření návodů s pokyny pro instalaci a nastavení pracovních nástrojů, které zajišťovaly snadné zprovoznění pracovního prostředí nových členů projektu. Tyto návody byly velkou výhodou, neboť vedení projektu se dále nemuselo problémy s konfigurací prostředí výrazně zabývat a navíc značně urychlily vstup a základní orientaci nových členů.

Dalšími přípravnými aktivitami byla tvorba zadání semestrálních prací, jejich prezentací a následné sestavení vývojového týmu. Jednotlivé semestrální práce rozdělují vývojový tým na dílčí skupinky. Každá skupinka (v některých případech i jednočlenná) tak vetšinou nezávisle na ostatních pracuje na jedné z částí hry.

Po tom, co se přihlásí zájemci o vypracování námi zadané speciální semestrální práce, je s nimi dohodnuta úvodní schůzka. Účelem této schůzky je seznámit studenty s projektem Space Traffic, sdělit jak do celkového konceptu zapadá jejich zadání a nastínit jeho předpokládaný postup realizace. V rámci této schůzky jsou také sdělovány pokyny – jaké kroky musí každý nový člen týmu provést, aby byl schopen pracovat jako plnohodnotný člen. Noví účastníci projektu jsou tak informováni o množině používaných nástrojů a dostupných návodech. Dále jsou od nich získány informace pro udělení přístupu k nástrojům, u kterých je přístup omezen pouze na členy týmu (Subversion, Redmine, Google Groups a Tiki Wiki). Na konci schůzky jsou pro usnadnění další komunikace mezi členy týmu a vedením vzájemně vyměněny kontakty (e-mail a Skype).

Na následující schůzce se obvykle řeší možné problémy s konfigurací pracovního prostředí, nebo s přístupem k některým nástrojům. Hlavní náplní je však detailní popsání stávající struktury implementovaného řešení, kde se nacházejí jaká vstupní a testovací data a jak provést konfiguraci hry, aby ji bylo možné úspěšně spustit. Dále je také detailněji rozebráno zadání a dílčím týmem je tak proveden první sběr požadavků.

Zatím, co dílčí týmy zpracovávají své semestrální práce, vedení projektu dohlíží na postup prací a průběžně kontroluje mezivýsledky, aby na konci jejich snažení byl takový přírůstek funkcionality hry, který je od jednotlivých dílčích týmů vedením požadovaný. Tato aktivita je prováděna z důvodu, že studenti, jak se ukázalo, nekladou z počátku spolupráce dotazy k přiděleným úkolům, neboť vztahy s vedením jsou v této době stále ještě příliš odtažité a studenti nižších ročníků nemají dostatek zkušeností. Dotazy nepokládají ani v případě, že zadání plně nepochopili s tím, že se později doptají kolegů, nebo se s úkolem nějak vypořádají. Pokud tedy u dílčích týmů nevznikají žádné otázky, je nutné, aby otázky pokládalo vedení, aby ověřilo dostatečné pochopení zadaného úkolu.

Při průběžné kontrole návrhu i programové implementace, se včas objeví možné chyby a je možné na ně rychle reagovat. Tyto chyby většinou plynou z nezkušenosti studentů a v případě bakalářských studentů z jejich neznalosti (např. studenti 2. ročníku bakalářského studia se teprve seznamují s objektovým návrhem programu). Po objevení takového nedostatku u dílčích týmů se vedení dostává do role mentorů a učitelů a vysvětluje správné způsoby a techniky nejen programování, ale velmi často také návrhu nebo testování. Udělováním rad, ukazováním správného směru a motivací (viz 5.5 Motivování studentů) je ze zkušenosti dosahováno kvalitnějšího výsledku.

Kdyby vedení průběžně nekontrolovalo činnost týmu, na uvedené problémy by se narazilo až na konci semestru. Vypracovaná řešení semestrálních prací by však nesplňovala kvalitativní požadavky, a pro následnou integraci do hlavní vývojové větve hry by bylo nutné získané programy více, či méně, přepracovat.

Na konci semestru je od týmů vyžadována dokumentace. Protože se na projektu střídá mnoho lidí včetně vedení (viz 4.7 Charakteristika prostředí), je důležité, aby byla dokumentace kvalitní. Zde se opět projevují problémy nezkušenosti a neznalosti nejen mladších studentů. Především studenti bakalářského studia často vůbec nevědí, co by měla dokumentace obsahovat. Jiní zase nevědí co je, a co není, pro zdokumentování důležité a vytvářejí „obsáhlé romány“ o tom, jak vypracovali zadanou semestrální práci. Zde opět nastupuje vedení projektu, které musí vysvětlit jakým způsobem dokumentaci vytvořit a následně ji mnohokrát kontrolovat, dokud neodpovídá požadavkům projektu.

Význam a přínos osobních schůzek

Ukázalo se, že velmi dobrým prostředkem pro zajištění kvality odevzdaných řešení, navázání přátelského vztahu dílčích týmů s vedením a lepšího využití časových možností týmu jsou časté osobní schůzky. Pode jejich povahy a cílů je možné rozlišit tři typy: pravidelné schůzky s dílčími týmy, schůzky na vyžádání dílčího týmu nebo vedoucích a schůzky společné pro celý realizační tým.

Pravidelné schůzky s dílčími týmy byly pořádány s každým týmem odděleně a konaly se jednou týdně. Účelem těchto schůzek byl dohled nad jednotlivými členy týmu, kontrola mezivýsledků řešení a zjišťování, jestli při práci nevznikl nějaký problém. Tento způsob organizace schůzek je časově nejnáročnější. Nejen, že se schůzky konají každý týden, ale vedoucí studenti se musí před schůzkou připravit. Příprava spočívá v kontrole rozpracovaného řešení, aby bylo možné na schůzce sjednat nápravu případných chyb. Tento typ schůzek výrazně převažoval v zimním semestru, ve kterém byly vedeny týmy studentů druhého ročníku bakalářského studia (viz vedení stylem přikazování v podkapitole 6.3.2).

Schůzky na vyžádání dílčím vývojovým týmem nebo vedením byly pořádány nahodile podle potřeby. Jejich cílem byl nejčastěji sběr požadavků, nebo vyřešení nějakého problému a dále pak zjišťování, jak daleko se dílčí tým dostal se svým řešením. Tyto schůzky nevyžadovaly tak důkladnou přípravu, jak tomu bylo v předchozím případě, a měly většinou pouze informativní charakter.

Schůzky celého realizačního týmu byly pořádány v zimním semestru pravidelně každých čtrnáct dní. Účelem těchto schůzek bylo dohlédnout na zapojení každého člena týmu a zároveň informovat všechny členy týmu o aktuálním dění v projektu. Tyto schůzky byly konány formou podobnou standup schůzce z metodiky Scrum. Na schůzce sdělil každý člen týmu (včetně vedení) ostatním, jakou práci od minula udělal, jestli se vyskytly nějaké problémy, popř. jak je vyřešil a co má v plánu udělat do příští schůzky. Konec schůzky byl věnován případným dotazům a podle potřeby domlouvání individuálních schůzek.

Časté scházení se s dílčími týmy a neustálý dohled nad jejich prací měl kromě získání výsledků v požadované kvalitě ještě jeden přínos. Bylo využito studentského syndromu ve prospěch projektu. Studentský syndrom, jak je popsán ve stejnojmenné podkapitole 4.7.3, je přirozeným chováním studentů, kdy studenti po získání úkolu zpočátku nevykazují téměř žádnou aktivitu vedoucí k jeho splnění a teprve pár dní před mezním termínem pro dokončení zadaného úkolu podávají maximální výkon, aby dohnali ztracený čas. Díky častým schůzkám bylo dosaženo zvýšené aktivity u členů dílčích týmů vždy před konanou schůzkou. Tím bylo získáno mnohem více času studentů, než kdyby se schůzky nekonaly a studenti by aktivně pracovali pouze dva až tři týdny před koncem semestru.

6.3.2 Vedení týmu podle jeho zralosti

Vždy je snaha o vybudování velkého realizačního týmu, aby práce rychle postupovaly a brzy se dosáhlo cílů projektu. Překročí-li velikost týmu jistou mez, stane se, že časová zátěž na vedení týmu bude tak vysoká, že vedení nebude schopné včas připravit dostatečné množství práce a tu následně průběžně kontrolovat. Z tohoto důvodu je nutné na základě svých zkušeností a schopností vybudovat tak velký tým, aby jej bylo možné z hlediska časových možností vedoucích studentů řídit.

Kromě výše popsaných aktivit, které byly prováděny pro podporu vývojového týmu, ovlivňuje zvládnutelnou velikost týmu také jeho zralost. Pojem „míra zralosti“ označuje rozsah dovedností a technických znalostí potřebných ke splnění úkolu společně s připraveností přijmout za jeho splnění odpovědnost. František Bělohlávek [18] definoval podle

zralosti lidí čtyři styly vedení:

Přikazování *„Lidem nezralým, kteří si nevědí rady, zapracovávají se do úkolu, je třeba přikazovat — vést krok za krokem, přesně určovat, co a jak se bude dělat a neustále na jejich práci dohlížet.“*

Tento styl musel být použit při vedení studentů druhého ročníku bakalářského studia v počátcích jejich spolupráce v zimním semestru. Vedení přikazováním se ukázalo jako nejvíce časově náročné a nebylo možné jím v rámci studia řídit více než čtyři takové členy týmu. Tento způsob byl použit protože členové týmu se dříve nesetkali s prací v týmu, používanými nástroji (Subversion, Redmine, Visual Studio, atd.), dokonce ani s programovacím jazykem C#.

Přesvědčování *„S tím, jak si pracovník osvojuje potřebné dovednosti, měl by se měnit i přístup vedoucího. Protože po stránce pracovní již dosáhli podřízení určité úrovně dovedností, může se vedoucí více zaměřit na interpersonální stránku úkolu. Nyní již podřízení nechtějí pouze bez rozmyslu přijímat pokyny vedoucího, protože se v mnoha věcech dokáží orientovat. Vedoucí jim bude vysvětlovat svá rozhodnutí a získávat je pro aktivní plnění úkolu.“*

Na tento styl vedení se přešlo zhruba v polovině zimního semestru u výše uvedené čtveřice lidí, protože již získali základní zkušenost s nástroji i programovacím jazykem, lépe se orientovali v zadaném úkolu a byli již schopni samostatnějšího návrhu řešení. Bylo zjištěno, že proti předchozímu stylu vedení nebylo ušetřeno výrazné množství vynaloženého času. Vývojový tým musel být stále směrován, aby se vydal správnou cestou.

Participování *„Vedoucí postupně omezuje pozornost věnovanou lidem, protože ti se učí zvládat své problémy sami. Vedoucí ponechává na nich, jaký přístup k problémům zvolí a mnohé věci rozhoduje společně s nimi.“*

Tento postup při vedení byl použit u dvou dílčích týmů v letním semestru. Použití bylo dáno povahou předmětů, na které byly zadány semestrální práce. Bylo tak vedeno osm lidí rozdělených do dvou čtyřčlenných týmů.

Delegování *„Pracovník vyspělý odborně i psychologicky, zvládá pracovní úkoly samostatně, bez vedoucího. Vedoucí spíše pomáhá při překonávání mimořádných událostí a věnuje se strategickým záležitostem.“*

Tento styl vedení vyžadoval v průběhu semestru minimální časové náklady proti všem výše uvedeným. Byl použit pro vedení tří semestrálních prací zpracovávaných jednotlivci (jedné v zimním semestru a dvou v letním semestru).

Ukázalo se, že v realizačním týmu mohou být zastoupeny všechny stupně zralosti studentů. Během vedení tak musely být vhodně použity všechny uvedené styly vedení.

7 Zhodnocení zkušeností a doporučení pro další vývoj

Tato kapitola popisuje zkušenosti, které během realizace projektu nabyli vedoucí členové jeho realizačního týmu. Jejím účelem je jednak předat zkušenosti nastupujícímu vedení, a dále pak poskytnout doporučení pro další postup při realizaci projektu. Následující text tak obsahuje výčty poznatků a skutečností zjištěných v průběhu realizace projektu z pohledu vedoucích členů. Některá z uvedených fakt se mohou zdát samozřejmá, ale historie ukázala, že se najdou tací, kteří si tyto skutečnosti neuvědomují. Z tohoto důvodu zde mají svůj význam.

7.1 Varování pro nastupující vedení projektu

Historie projektu ukázala (viz diplomová práce Richarda Kocmana [29]), že následky poznatků uvedených v seznamu níže mohou mít drtivý dopad na splnění cílů.

1. Nový vedoucí (manažer) projektu ve většině případů nemá zkušenost s řízením takto rozsáhlého projektu.
2. Pokud se nové vedení na začátku včas důkladně neseznámí s projektem, nebude schopné stanovit optimální plán pro zimní semestr, zadat vhodné semestrální práce a uvést sestavený tým do realizace projektu.
3. Čím je vytvářená hra blíže ke svému dokončení, tím více dokumentací je nutné nastudovat (nebo alespoň přečíst) pro plné pochopení konceptu hry a její stávající implementace.

Vždy existuje riziko, že nový manažer projektu nezvládne svoji roli dostatečně zastat. Pravděpodobnost, že toto riziko nastane je zvýšena tím, že katedra v žádném ze svých studijních programů nenabízí studium tohoto oboru. Hrozbou nezvládnutí vedoucí role je možné alespoň částečně potlačit dvěma opatřeními:

- postavit do čela projektu více vedoucích studentů
- a obsadit vedoucí pozice studenty zapálenými do realizovaného projektu.

Pokud bude ve vedení projektu více studentů z posledního ročníku studia, mohou si všemožně vypomáhat a být si oporou. Zapálení do projektu a do jeho vedení je pak velkým motivačním faktorem, který významně působí na kvalitu odváděné nejen manažerské práce.

Nové vedení se musí po svém vstupu do projektu seznámit se všemi jeho součástmi, aby nenastal scénář uvedený v bodě 2 výše uvedeného seznamu poznatků. Je nutné si na prostudování projektu vyhradit dostatek času, neboť se zvětšujícím se rozsahem implementovaného řešení se zvyšuje jak jeho složitost tak i množství dokumentace.

Jakkoli je však vytváření dokumentace přínosné pro snadné pochopení principů návrhu i programové implementace, není možné zdokumentovat vše a musí se volit kompromis mezi množstvím odvedené programátorské práce a množstvím vytvořené dokumentace. Důvodem je, že hlavním zdrojem v projektu je čas studentů, kterého není mnoho. Student se ve většině případů musí nejprve seznámit s technologií a naučit se s ní pracovat, než začne být produktivní.

Se zbývajícím časem, který student projektu věnuje je dle názorů odstupujících vedoucích lepší naložit tak, aby odvedená programátorská práce byla co nejvíce kvalitní na úkor plného zdokumentování. Tento fakt však nepopírá trend stále se zvyšujícího množství dokumentací v projektu.

7.2 Získávání studentů

Následující seznam obsahuje výčet zkušeností se získáváním studentů do projektu.

1. U studentů nelze předpokládat dlouhodobou spolupráci.
2. U studentů nelze předpokládat, že se budou na vývoji podílet nad rámec svého studijního plánu.
3. Nejlepším způsobem, jak získat studenty pro spolupráci, je zřejmě zadávání speciálních semestrálních prací na obsahově související vyučované předměty.
4. Studenty je nutné k výběru speciálních zadání motivovat.
5. Pokud není v zadání semestrální práce jednoznačně uvedeno, že používání nástroje Redmine během vývoje je nutnou podmínkou pro získání zápočtu, najdou se takové dílčí týmy, které ho i přes četné výzvy vedoucích používat nebudou.

První dva body seznamu není třeba dále rozvádět, jejich význam je jednoznačný. Důležitější je zaměřit se na jejich řešení. V průběhu realizace projektu se ukázalo, že klíčem k získání studentů pro spolupráci jsou zadání speciálních semestrálních prací ve vybraných vyučovaných předmětech. Výsledkem analýzy této možnosti v přípravné fázi projektu bylo zjištěno, že zadání musí pro přijetí ve zvoleném předmětu splňovat dvě základní podmínky:

- výsledný produkt, který bude zpracován, musí splňovat stejná kritéria na použití vyučovaných témat jako standardní zadání,
- speciální zadání musí být formulováno ve stejném rozsahu a detailu jako standardní zadání.

Pokud je problém nastavit zadání tak, aby výsledný produkt splňoval požadavky předmětu a přitom měl ještě užitečný přínos pro projekt Space Traffic, je lepší takové zadání opustit. Případně je možné jej sestavit tak, aby jeho řešení maximalizovalo přínos, a zadat jej jako projekt (předmět KIV/PRJ).

Aby bylo dosaženo maximálního užitku, musí vytvořená zadání budit dojem jednoduchosti. Hlavně však musí být motivující k výběru, neboť již samotný fakt, že se jedná o speciální zadání odrazuje od jeho výběru. Výběr těchto zadání je možné podpořit prezentacemi na přednáškách předmětů, na kterých jsou zadávána.

Na těchto prezentacích je možné více přiblížit, jaký bude úkol studentů, kteří si zadání vyberou, je možné reagovat na názory studentů (viz podkapitola 4.7.1 Lidské zdroje) i když je nikdo nahlas nevysloví. Dále je dobré sdělit výhody spolupráce v týmu projektu Space Traffic a motivovat k výběru zadání, jak je uvedeno v kapitole 5.5 Motivování studentů.

7.3 Plánování

Níže uvedené poznatky mají vliv na splnění všech vytvářených plánů v projektu (plán pro akademický rok nebo jednotlivé semestry) i při vytváření zadání speciálních semestrálních prací.

1. Úvazek studentů k projektu je určený jejich semestrálními pracemi a jeho délka je tak omezena na pouze jeden semestr.
2. Množství času, které studenti věnují spolupráci na projektu není v průběhu semestru rovnoměrně rozložené, ale kolísá.
3. Čas studentů věnovaný vlastní programové implementaci je omezený dalšími jimi vykonávanými činnostmi (viz dále).
4. Stanovený plán se nikdy nestihne, ani když se udělá dostatečně malý.
5. Množství odvedené práce je určeno v první řadě velikostí týmu a kvalifikací jeho členů.
6. Množství i kvalita odvedené práce je ovlivněna počtem osobních schůzek s dílčími týmy.
7. Výsledky vedoucích vývojového týmu jsou závislé na výkonu všech jeho členů.
8. Součástí vývojového týmu mohou být i studenti nekvalifikovaní řešit zadaný úkol i spolupracovat v týmu.
9. Zejména mladší studenti nevykonávají před samotným programováním dostatečné přípravy (sběr požadavků, návrh, atd.).
10. Nechat vypracovat stejné zadání semestrální práce více dílčími týmy je vhodné pouze u opodstatněných případů (viz dále).

Výše uvedené poznatky by měly být brány v úvahu zejména při vytváření plánu pro akademický rok, jednotlivé semestry, i při vytváření zadání speciálních semestrálních prací. Některé z nich budou v následujícím textu blíže rozebrány.

ad 1. Protože úvazek studentů ke spolupráci v projektu je omezen na jeden semestr a nelze předpokládat jejich dlouhodobější zapojení, měla by být snaha minimalizovat náklady z hlediska času vedoucích studentů na zaučení nových členů týmu do používaných technologií a nástrojů.

ad 2. Tento charakter času věnovaného projektu je dán akademickým prostředím. Studenti musí během semestru dodržovat termíny zápočtových testů a termíny odevzdávání průběžných zápočtových úloh jiných předmětů. Důsledkem toho je pak ztížení možnosti plánovat, vytvářet časové odhady a řídit se na základě proběhlých iterací.

ad 3. Studenti – členové vývojového týmu (jak vývojáři, tak vedoucí) jsou zatíženi v průběhu jejich zapojení do projektu mnoha činnostmi. Vývojáři se musí s projektem nejprve seznámit (nebo alespoň s jeho částí, se kterou budou pracovat). Ve většině případů se musí seznámit také s programovacím jazykem C# a vývojovým prostředím. Až teprve poté vykonávají činnosti návrh řešení, programová implementace, testování výsledku a tvorba dokumentace. Vedoucí projektu provádějí kromě vlastní programové implementace také tvorbu a prezentace zadání speciálních semestrálních prací, vedení jim podřízených členů týmu, kontrolování jejich aktivity a vzniklých artefaktů (návrhy řešení, programový kód, testy, dokumentace) a integrují výsledek jejich snažení do hlavní vývojové větve projektu, pokud integrace nebyla jejich úkolem.

ad 4. Během vývoje se ukázalo, že ať byl vytvořený plán iterace jakkoliv pesimistický či optimistický, nikdy nebyly splněny všechny jeho body. Pro lepší představu je možné říci, že z plánu bylo splněno vždy 80 až 90 procent. Z toho plyne, že nemá smysl vytvářet plány malého rozsahu, neboť s malým plánem vzniká dojem, že na jeho splnění je stále dost času a v konečném důsledku stejně není splněn. Naopak je vhodné vytvořit plán o něco větší, ale nepřehánět, aby nevznikl dojem, že plán je nereálný. V takovém případě se ho nikdo ani nesnaží dodržet.

ad 6. Časté osobní schůzky se ukázaly jako velmi dobrý prostředek pro zajištění kvality práce odvedené vývojovým týmem. Více o jejich přínosech je uvedeno v podkapitole 6.3.1 Podpora vývojového týmu.

ad 7. Nebude-li vedení projektu s vývojovým týmem vhodně komunikovat (jednat a vyjednávat), nebudou dobré výsledky ani týmu ani vedoucích.

ad 9. Tento fakt vychází především z nezkušenosti studentů vytvářet nějaký produkt formou zakázky. Studenti jsou zvyklí, že standardní zadání jsou zadána do jisté míry volně (aby nutily studenta přemýšlet a nenastínily hned celé řešení úlohy) a fantazii tedy meze nekladou. Problémem však je, že studenti tuto kreativitu přenášejí i do projektu Space Taffic a co není v zadání přímo sděleno, si domyslí.

ad 10. Duplicitní zadávání semestrálních prací více dílčím týmům má své výhody i nevýhody a je tak nutné předem důkladně rozmyslet, bude-li pro projekt přínosem. Výhodami jsou:

- možnost vybrat si řešení, které bude použito (integrováno do hlavní vývojové větve),
- vyšší pravděpodobnost dokončení práce (pro případ, že by jeden tým selhal),
- možnost poučení všech z lepšího řešení.

Nevýhodami tohoto způsobu zadávání semestrálních prací je, že:

- duplicitní týmy nemohou pracovat v hlavní vývojové větvi (nutná integrace řešení),
- pro získání jednoho efektivního výsledku musí být vedeno více týmů.

Jak je vidět, při rozhodování bude mít určitě rozhodující vliv důležitost vypracování konkrétní úlohy a množství času, které vedení projektu může podpoře řešitelských týmů věnovat.

7.4 Spolupráce v týmu

Tento seznam uvádí zkušenosti se spoluprací členů vývojového týmu.

1. Pro potřeby rychlé komunikace mezi členy týmu se velmi osvědčil nástroj Skype.
2. Domlouvání osobních schůzek je nejvhodnější řešit prostřednictvím e-mailu.
3. Softwarové nástroje pro podporu ekosystému projektu nejsou vším.
4. Budování „kamarádské“ atmosféry mezi všemi členy realizačního týmu (včetně vedení) pomáhá odstranit komunikační bariéry.
5. Vytvořené návody snižují v závislosti na oblasti, kterou pokrývají, zátěž kladenou na vedoucí projektu.
6. Zodpovědnost dílčího vývojového týmu (popř. jednotlivce) za programový kód vytvářený v rámci jemu zadané semestrální práce znemožňuje rychlé opravení chyb nalezených v tomto kódu jiným vývojářem.

V následujícím textu budou opět rozebrány některé z výše uvedených zkušeností s týmovou spoluprací a jiných poznatků týkajících se její podpory.

ad 2. Stalo se, že na schůzku domluvenou prostřednictvím nástroje Skype, vývojový tým nedorazil. Skype po přihlášení uživatele nezobrazuje konverzaci starší než jeden den a snadno se tak na schůzku zapomene.

ad 3. Stejně důležité jako softwarové nástroje pro podporu projektu jsou také komunikační schopnosti vedoucích, motivace, fantazie, nápaditost a také odvaha zkoušet nové věci.

ad 5. Během příprav projektu byly vytvořeny návody, které zajišťovaly snadné zprovoznění pracovního prostředí nových členů týmu. Návody byly velkou výhodou, protože jednak odlehčily zátěž na vedoucí projektu a jednak urychlily vstup nových členů do projektu.

ad 6. Pokud je nalezena nějaká chyba v řešené semestrální práci, musí být oprava zadána řešiteli úlohy. Ten však nemusí mít zrovna čas na její opravu. Ten, kdo chybu nalezl by ji ale opravit neměl, přestože by mu to zabralo pár minut. Vyřešení chyby je totiž v rámci kompetence konkrétního dílčího vývojového týmu nebo jednotlivce, který má za řešení své úlohy zodpovědnost.

7.5 Výsledky vedoucích projektu

V předchozích podkapitolách bylo uvedeno, jakým způsobem bylo postupováno při řízení projektu. Tato kapitola uvádí kvantitativní vyhodnocení úspěšnosti použitých postupů. Na tomto místě tedy budou v číslech shrnuty výsledky práce celého týmu po uplynulém akademickém roce.

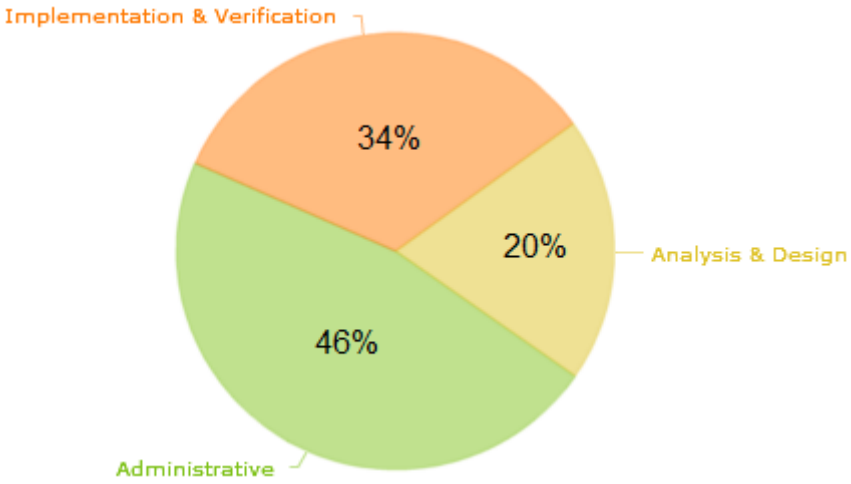
Z nástroje Redmine a z odhadů míry aktivity studentů v zimním semestru bylo zjištěno, že na projektu bylo v uplynulém akademickém roce stráveno 1100 hodin. Toto množství času bylo rozloženo mezi deset iterací (viz podkapitola 5.2.2 Fáze vývojového cyklu). Protože je však čas strávený prací na projektu značně nerovnoměrný (viz 4.7 Charakteristika prostředí), nelze vyvodit závěr, že na jednu iteraci připadalo zhruba 110 odpracovaných hodin.

Na projektu se v uplynulém roce podílelo celkem 15 studentů. Tři z nich (včetně autora této práce) byly jeho součástí prostřednictvím svých diplomových prací. Další čtyři studenti magisterského studia se projektu účastnili v rámci předmětu Pokročilé softwarové inženýrství (KIV/ASWI). Zbylých sedm studentů bylo z bakalářského studia a podíleli se na vývoji v rámci předmětů: Programovací techniky (KIV/PT), Znalostní a informační management (KIV/ZIM), Základy softwarového inženýrství (KIV/ZSWI), Programování v prostředí .NET (KIV/NET), Projekt 2 (KIV/PRJ2) a Projekt 4 (KIV/PRJ4). Zadání semestrálních prací, která byla formulována písemně, jsou umístěna v příloze A.

Nepočítají-li se diplomové práce, bylo v průběhu uplynulého akademického roku zadáno osm speciálních semestrálních prací. To znamená, že bylo vedeno osm dílčích týmů a uplatněno 16 úvazků k projektu. V zimním semestru byly zadány tři práce, které do projektu přinesly pět úvazků, a v letním semestru bylo zadáno pět prací s přínosem jedenácti úvazků.

Čím více studentů se na projektu podílí, tím více času je nutné věnovat jejich vedení a administraci spojené s touto činností. Graf na obrázku 7.1, získaný z nástroje Redmine, ukazuje poměr doby strávené administrativními úkony k ostatním aktivitám. Je tedy vhodné sestavit pouze tak velký tým, jaký je možné z pozice studenta řídit.

Při vytváření odhadů náročnosti jednotlivých úkolů i plánů celých iterací nebyla dosud k dispozici žádná pomůcka. Nyní s odstupem jednoho akademického roku, změřenými



Obrázek 7.1: Poměr dob strávených různými aktivitami v projektu.

základními veličinami a získanými zkušenostmi je možné pro budoucí vedení projektu takovou pomůcku vytvořit. Onou pomůckou je tzv. studento-měsíc.

Alternativního označení měrné jednotky studento-měsíc místo běžného označení člověko-měsíc bylo použito především z důvodu specifických charakteristických vlastností prostředí realizace projektu (viz kapitola 4.7 Charakteristika prostředí). Dalším důvodem bylo, aby bylo na první pohled zřejmé rozdílné chápání pracnosti. Jednotka studento-měsíc udává přibližný produktivní čas jednoho studenta za měsíc. Jedná se však o čas, který student věnuje jednomu z více projektů (popř. semestrálních prací), v nichž je současně zapojen.

Počet produktivních hodin byl vypočítán na základě celkové doby strávené všemi studenty na realizaci projektu (1100 hodin) a průměrného počtu studentů podílejících na realizaci projektu v celém akademickém roce. Průměrný počet studentů za akademický rok byl spočítán jako aritmetický průměr počtu studentů podílejících se na projektu v obou semestrech (zimní semestr: 7 studentů, letní semestr: 13 studentů[1], průměr za rok: 10 studentů).

Z 1100 produktivních hodin za akademický rok a počtu 10 studentů získáme hodnotu 110 produktivních hodin jednoho studenta za akademický rok. Po odečtení dvou měsíců (prosinec a leden) od délky akademického roku dostaneme 8 produktivních měsíců. Vydělením získaných hodnot zjistíme přibližnou hodnotu jednoho studento-měsíce, kterou je 13,75 produktivních hodin jednoho studenta za měsíc.

Zaměřme se zde nyní na tok lidí v projektu. Na počátku zimního semestru se do projektu zapojilo pět bakalářských studentů. Při přechodu ze zimního semestru do letního odstoupili z projektového týmu dva členové. Tito studenti studovali jiný obor než informatiku a jejich studijní plány na letní semestr nezahrnovaly předměty, na které byly zadávány speciální semestrální práce. U zbylých tří studentů byla situace pro projekt pří-

[1] Jeden ze studentů řešil pro projekt Space Taffic v letním semestru dvě semestrální práce zároveň, proto je uvedeno 13 studentů místo 12.

znivější. Všichni si opět vybrali některou z nabízených speciálních prací a jeden z nich se projektu účastnil prostřednictvím dokonce dvou předmětů.

Na začátku letního semestru se do projektu zapojilo ještě dalších sedm studentů – tři z bakalářského a čtyři z magisterského studia. Z tohoto počtu projevili vážný zájem na pokračování v projektu čtyři studenti bakalářského studia.

Na konci uplynulého akademického roku se díky vynaloženému úsilí při tvorbě propagačních plakátů podařilo získat dva studenty pro vedení projektu v rámci jejich diplomových prací. Byla tak zajištěna realizace projektu i v příštím roce.

7.6 Doporučení

Závěrem této kapitoly si autor této práce společně s Martinem Štěpánkem jakožto vedoucí projektu Space Traffic v letošním roce dovolují vzhledem k úspěšnosti jejich vedení sedmero doporučení:

1. Nepodcenit důležitost příprav během letních prázdnin.
2. Rozšířit soubor vytvořených návodů o testování a tvorbu dokumentace a využít k tomu vytvořeného videotutoriálu o tom, jak vytvořit videotutorál.
3. Postupně v průběhu další realizace projektu vybudovat na základě řešených problémů v nástroji Tiki wiki sekci pro často kladené dotazy (FAQ).
4. Pokusit se o motivaci studentů, jak je popsáno v podkapitole 5.5 Motivování studentů.
5. Snažit se do dalšího vývoje zapojit především ty studenty, kteří se na projektu už v minulosti podíleli.
6. V případě duplicitního zadávání semestrálních prací pořádat schůzky společné pro všechny týmy řešící stejné zadání.
7. Ověřovat již na schůzce, že studenti jednak pochopili co mají dělat, ale také, že vědí jak.

Uvedená doporučení není potřeba více komentovat.

8 Závěr

Tato diplomová práce se zabývá podporou a řízením vývoje webové hry Space Traffic pro více hráčů. Cílem této práce bylo zajištění dostatečné podpory pro ekosystém projektu vývoje této hry, který je realizován na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni. Dosažení tohoto cíle spočívalo v zajištění zdrojů a technického zázemí projektu, vytvoření dostatečné podpory pro komunitu projektu, navržení postupu vývoje hry a napomoci dosažení kvalitního výsledku.

Na základě analýzy stavu projektu při jeho převzetí, charakteristických vlastností prostředí jeho realizace, existujících rizik a metodik pro vývoj softwarových projektů byly navrženy a použity postupy pro jeho řízení a podporu. Postup řízení vývoje zohledňující znalosti, schopnosti a zkušenosti členů realizačního týmu (studentů) je popsáno v kapitole 5 Řízení projektu Space Traffic. Zajištění podpory z výše uvedených hledisek bylo podřízeno aktuálním potřebám projektu. To znamená, že největší důraz byl kladen na plynulost probíhajícího vývoje, zajištění kvality výsledků, podporu komunikace a sdílení informací mezi členy realizačního týmu, jak uvádí kapitola 6 Podpora ekosystému projektu.

Na reálném projektu vývoje webové hry Space Traffic vyhodnocuje tato práce vhodnost použitých postupů a podpůrných nástrojů v akademickém prostředí. Postupy, které byly použity při řešení této práce pro řízení projektu Space Traffic a zjištění jeho podpory, byly popsány v takovém detailu, aby je bylo možné na základě této práce opakovat, poučit se z nich, případně je zlepšit. Práce tedy popisuje, jakým způsobem projekt v tomto roce fungoval, a na svém konci uvádí autorovy zkušenosti s řízením projektu, vedením jeho realizačního týmu a zvolenými podpůrnými nástroji.

Výsledkem této práce je, že byla zajištěna podpora projektu postupy popsanými v praktické části a v jejím průběhu se nevyskytly žádné závažné problémy. Efektem dobrého vedení jednotlivých členů vývojového týmu je, že většina jeho členů z bakalářského studia má zájem se v následujícím akademickém roce opět podílet na vývoji hry. Cíle této práce tedy byly naplněny a díky vynaloženému úsilí se ke svým cílům přiblížil i projekt vývoje hry Space Traffic.

Symboly a zkratky

Přehled v práci použitých či obecněji používaných symbolů a zkratek

ANSI	–	American National Standards Institute
CMS	–	Content management system
CRAMM	–	Analýza rizik a řízení bezpečnosti informací
CRI	–	Continuous Risk Improvement
ERP	–	Enterprise resource planning
EWRM	–	Enterprise-Wide Risk Management
FAQ	–	Frequently Asked Questions
FDD	–	Feature Driven Development
FPS	–	First-person shooter
IDE	–	Integrated development environment
IIS	–	Internet Information Services
JSON	–	JavaScript Object Notation
KIV	–	Katedra informatiky a výpočetní techniky
LS	–	Letní semestr
MMC	–	Master Control Computer
MMOG	–	Massively-Multiplayer Online Game
M_o_R	–	Management of Risk
MSDN	–	Microsoft Developer Network
PMI	–	Project Management Institute
RMF	–	Risk Management Framework
RTC	–	IBM Rational Team Concert
RTS	–	Real-time strategy
RUP	–	Rational Unified Process
SVG	–	Scalable Vector Graphics
TBS	–	Turn-based strategy
TOPSIS	–	Technique for Order Preference by Similarity to Ideal Solution
TPS	–	Third-person shooter
UP	–	Unified Process
XP	–	Extreme Programming
ZS	–	Zimní semestr

Literatura

[1] PROJECT MANAGEMENT INSTITUTE. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide) 4th edition*. Pennsylvania: Project Management Institute, Inc., 2008. ISBN: 978-1-933890-51-7.

[2] OFFICE OF GOVERNMENT COMMERCE. *Managing Successful Project with PRINCE2™, 6th edition*. United Kingdom: The Stationery Office, 2009. ISBN 978-0-11-331059-3.

[3] SALEN, Katie a Eric ZIMMERMAN. *Rules of Play: Game Design Fundamentals*. Cambridge, MA, USA: MIT Press., 2004, s. 414-415. ISBN 0-262-24045-9

[4] COSTIKYAN, Greg. *I Have No Words & I Must Design*. Costik.com [online]. 1994 [cit. 2012-06-28]. Dostupné z: http://www.costik.com/nowords.html

[5] O'REILLY, Tim. *Web 2.0 Compact Definition: Trying Again*. radar.oreilly.com [online]. 2006 [cit. 2012-06-28]. Dostupné z: http://radar.oreilly.com/archives/2006/12/web-20-compact.html

[6] APPERLEY, Thomas H. Genre and game studies: toward a critical approach to video game genres. In: *Simulation and Gaming - Symposium: Video games: Issues in research and learning* [online]. Thousand Oaks, CA, USA: Sage Publications, Inc., 2006, volume 37 issue 1, s. 6-23. ISSN: 1046-8781, doi: 10.1177/1046878105282278. Dostupné z: http://sag.sagepub.com/content/37/1/6.full.pdf+html

[7] KLEMENT, Milan. Možnosti evaluace výukových programů. In: *Trendy technického vzdělávání*. Olomouc: Votobia, 2005. s. 17–29. ISBN 80-7220-227-8.

[8] DOSTÁL, Jiří. Výukový software a počítačové hry - nástroje moderního vzdělávání. *Časopis pro technickou a informační výchovu* [online]. ISSN 1803-537X. Dostupné z: http://infotech.upol.cz/web_sbornik/sbornik_INFOTECH07_dil_1.pdf

[9] DOSTÁL, Jiří. Informační a počítačová gramotnost – klíčové pojmy informační výchovy In: *Infotech 2007 - moderní informační a komunikační technologie ve vzdělávání* [online]. Olomouc: Votobia, 2007. s. 60–65. ISBN 978-80-7220-301-7. Dostupné z: http://infotech.upol.cz/web_sbornik/sbornik_INFOTECH07_dil_1.pdf

[10] JIAN, Jie and Yueguang XIE, Wenhe TANG a Chunhui WANG. Webgame Based Collaborative Learning Design: A Case Study. In: *Simulation and Gaming - Symposium: Video games: Issues in research and learning* [online]. Springer Berlin / Heidelberg, ©2010, volume 6249, s. 223-234. ISBN: 978-3-642-14532-2, Doi: 10.1007/978-3-642-14533-9_23. Distupné z: http://dx.doi.org/10.1007/978-3-642-14533-9_23

[11] SCHELL, Jesse. *The Art of Game Design: A Book of Lenses*. USA: Elsevier Inc., 2008. ISBN: 978-0-12-369496-6.

[12] GERŠL, Vladimír. Co je game-design a jak vlastně vzniká In: *games.zcu.cz* [online]. ZČU v Plzni, FAV, KIV, 27.04.2012 [cit. 2012-06-28].

[13] TAYLOR, Frederick Winslow. *The Principles of Scientific Management* [online]. UK: Harper & Brothers, 1911. Dostupné z: http://www.gutenberg.org/ebooks/6435

[14] WITZEL, Morgen. *Fifty key figures in management*. Routledge, 2003. ISBN 0-415-36977-0.

[15] STANÍČEK, Zdenko, RNDr. Řízení projektů – podstata řízení projektů. In: *IT Systems* [online]. CCB, spol. s r.o., 2002, č.12 [cit. 2012-06-28]. Dostupné z: http://www.ipma.cz/dokumenty_ clanky/RP1.pdf

[16] SCHWALBE, Kathy. *Řízení projektů v IT*.e Brno: Computer Press a.s., 2007. ISBN: 978-80-251-1526-8.

[17] HELDMAN, Kim. *PMP®: Project Management Professional Study Guide*. Alameda, CA 94501: SYBEX Inc., 2002. ISBN: 0-7821-4106-4.

[18] BĚLOHLÁVEK, František, *Jak řídit a vést lidi: jak s nimi jednat, jak je vést a motivovat*. Praha: Computer Press, 2000. ISBN: 80-7226-308-0.

[19] DESAULNIERS, Douglas H. a Robert J. ANDERSON. Matching Software Development Life Cycles to the Project Environment. In: *Proceedings of the Project Management Institute Annual Seminars & Symposium* Nashville, TN. Newtown Square, PA: Project Management Institute, 2001.

[20] ROYCE, Winston W. Managing the Development of Large Software Systems. In: *Proceedings of IEEE WESCON*. August 1970, s. 1-9.

[21] U.S. DEPARTMENT OF TRANSPORTATION. *Systems Engineering for Intelligent Transportation Systems*. Washington DC, 2007.

[22] BOUHM, Barry W. A Spiral Model of Software Development and Enhancement. In: *Computer*. May 1988, volume 21, issue. 5, s. 61-72. doi: 10.1109/2.59.

[23] PALMER, Stephen R. a John M. FELSNG. *A Practical Guide to Feature-Driven Development*. Prentice Hall, 2002. ISBN-10: 0130676152.

[24] SHUJA Ahmad K. a Jochen KERBS. *IBM® Rational Unified Process® Reference and Certification Guide*. USA: IBM Press, 2007. ISBN: 978-0-13-156292-9.

[25] POPELKA, Vladimír. *Srovnávací analýza metodik vývoje software.* Praha, 2009. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky.

[26] McCONNELL, Steve. *Dokonalý kód: Umění programování a techniky tvorby sofrware* Brno: Computer Press a.s., 2006. ISBN: 80-251-0849-X.

[27] NEUDERT, Zbyněk. *Analýza, návrh a vedení týmu v projektu webové hry.* Plzeň, 2010. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky.

[28] ŠTĚPÁNEK, Martin. *Architektura a implementace webové hry pro více hráčů.* Plzeň, 2012. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky.

[29] KOCMAN, Richard. *Řízení projektu webové hry.* Plzeň, 2012. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky.

[30] McCARTHY, Jim. *Softwarové projekty.* Brno: Computer press, 1999. ISBN: 8072261940.

[31] COSMOCODE. WikiMatrix: compare them all. *wikimatrix.org* [online]. Dostupné z: http://www.wikimatrix.org

Pozn.: Pokud u jednotlivých citací chybí nějaký údaj, nepodařilo se mi jej zjistit.

Příloha A – Zadání speciálních semestrálních prací

Dvě zadání pro KIV/NET

O Space Traffic – Co, proč, jak

Na katedře se vytváří webová hra Space Traffic typu MMORTS zaměřená na obchodování ve vesmíru. Hra bude obsahovat programovatelné prvky, jejichž funkci budou hráči moci pomocí svých uložených programů automatizovat. Herním plánem jsou dynamické mapy slunečních soustav, které jsou propojeny červími dírami. Úkolem hráče je nakupovat zboží „levně“ a prodávat „draze“. K tomuto účelu hráč vlastní (programovatelné) vesmírné lodě, jejichž prostřednictvím zboží distribuuje zákazníkům. Podrobnější informace o game designu získáte z wiki http://spacetraffic.kiv.zcu.cz/dokuwiki/doku.php?id=gamedesign

Hra má za cíl prezentovat katedru informatiky a výpočetní techniky a zároveň má dostat katedru do povědomí více středoškolských studentů a tím také podnítit jejich zájem o studium na KIV. Více informací k hlavní ideje projektu je možné získat na wiki stránkách katedry na adrese http://wiki.kiv.zcu.cz/SpaceTraffic/HomePage

Space Traffic je ASP.NET MVC 3 webovou aplikací. Serverová část je psána v jazyce C# a je nasazena na webovém serveru IIS 7. Aplikační vrstva na klientské straně je vytvářena v JavaScriptu a běží tedy ve webovém prohlížeči. Dalšími technologiemi použitými ve větší míře jsou jQuery, JSON a SVG.

Bonus

V případě dalšího zájmu je možné v projektu pokračovat v rámci PRJ5, bakalářské práce i diplomové práce.

Kontakt

Petr Vogl, voglpetr@students.zcu.cz

Grafický editor pro vytváření a úpravu hvězdných systémů pro hru Space Traffic

Cílem práce je vytvořit grafický editor pro soubory s popisy hvězdných systémů (soubor je ve formátu XML). Editor má být schopen grafické vizualizace celého hvězdného systému a editace trajektorií pomocí drag & drop myší (tzn.: uchopí se trajektorie a tahem myši se modifikuje její tvar).

Editor musí být schopen jak načíst a upravit existující XML popis hvězdného systému, tak vytvořit úplně nový (buď na základě existujícího, nebo s náhodnými daty či úplně čistý). Měl by obsahovat různá vstupní pole pro zadávání textového popisu jednotlivých planet a červích děr. Pro usnadnění editace propojení červích děr s ostatními hvězdnými systémy musí být schopen načíst XML popis galaktické mapy.

Také bychom chtěli, aby bylo možné spustit simulace pohybu hvězdného systému (velká část je již připravena).

Editor a podpora pro achievementy hry Space Traffic

Popis

Achievementy představují cíle, které hráč při hraní hry může dobrovolně splnit. Představují doplňující systém postupu hráče hrou.

Achievementy lze dělit na tři základní skupiny:

- Nevyhnutelné (získané normálním hraním hry)
- Dobrovolné (představují úkoly a cíle, jejichž splnění si může hráč sám zvolit)
- Inspirující (předkládají hráči výzvy, které mění herní zážitek a umožňují hráči vyzkoušet alternativní způsoby hraní)

Typy achievementů

- **Jednorázový** - získán při dosažení kritérií.
- **Víceúrovňový** - získáván na základě kritérií s odstupňovanou obtížností (např.: good, great, perfect).
- **Seznam** - získává se postupně "sbíráním" jednotlivých položek seznamu.
- **Nekompletní seznam** - získán za "sebrání" určitého počtu položek seznamu.

- **Počítadlo** - získává se sbíráním bodů, měl by obsahovat několik úrovní splnění (např.: bronzová, stříbrná, zlatá).
- **Meta-achievement** - Funguje jako seznam, ale položkami jsou achievementy.

Zadání

Vytvořte editor (včetně GUI) pro vytváření achievementů. Definice achievementů budou ukládány do souborů ve formátu XML a budou mít podporu pro lokalizaci. Dalším požadavkem je navržení způsobu ukládání stavu plnění achievementů jednotlivými hráči do databáze. Prezentace seznamu achievementů bude v HTML.

Dále implementujte jednoduchý program pro demonstraci fungování všech typů achievementů. Demo bude simulovat aktivitu hráče, vyhodnocovat stav plnění ukázkových achievementů a signalizovat jejich dosažení. Implementace vyhodnocování achievementů musí mít jednotné rozhraní nezávislé na typu konkrétních achievementů.

Inspirace WoW, Steam.

Zadání pro KIV/ZIM

Podpůrné systémy pro projekt SpaceTraffic

Krátce o SpaceTrafficu

Jde o týmový projekt webové online hry zaštítěný p. Bradou. Vytvářená hra bude vesmírnou obchodní strategií, která má za cíl přilákat středoškoláky ke studiu na FAV - KIV. Vývojový tým se pro tento semestr skládá ze sedmi studentů.

Co potřebujeme, co by nám pomohlo

Potřebovali bychom nalézt vhodný nástroj pro výměnu informací a znalostí (tj. dokumentací a návodů v jakékoliv podobě - text, video, atd.) v týmu studentů, jehož členové (i vedení) se často mění. Tento nástroj by měl umožňovat snadné vytváření a design dokumentů a dokumentací (podobně jako na wiki) s možností exportu definova- ných částí do strukturovaného textového dokumentu (pdf, rtf, atp.).

Vaším úkolem by tedy, kromě jiného (viz Portál), bylo sepsat analýzu možných řešení, vybrat nejvodnější (po konzultaci s týmem SpaceTrafficu) a poté řešení realizovat.

Kontakt

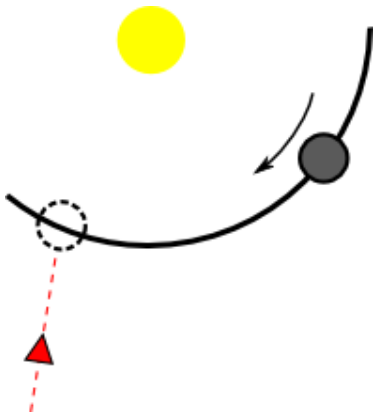
Petr Vogl, voglpetr@students.zcu.cz

1. zadání pro KIV/ZSWI

Nalezení místa střetu dvou těles

Zadání

Na katedře se vytváří webová hra Space Traffic zaměřená na obchodování ve vesmíru. Vaším úkolem bude vytvořit mechanismus pro nalezení místa střetu dvou vesmírných těles tj. vesmírné lodi a planety (pouze v rovině, nikoliv v prostoru).



Požadavky

- Dodržovat pravidelné schůzky s vedoucími projektu
- Seznámit se se základní prací se subversion
- Implementovat mechanismus hledání střetu dvou těles
- Implementace v jazyce C# (v tomto případě není žádný rozdíl od Javy)

Bonus

V případě dalšího zájmu lze v projektu pokračovat v rámci PRJ5 a bakalářské práce.

Zadavatel

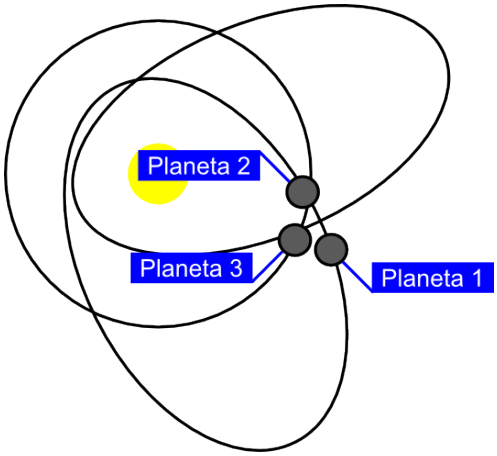
Petr Vogl, e-mail: voglpetr@students.zcu.cz

2. zadání pro KIV/ZSWI

Popisky vesmírných těles

Zadání

Na katedře se vytváří webová hra Space Traffic zaměřená na obchodování ve vesmíru. Vaším úkolem bude navrhnout a implementovat vykreslení popisku planet a lodí. Popisky dvou (a více) těles, která se nacházejí blízko sebe, by se neměly překrývat.



Požadavky

- Dodržovat pravidelné schůzky s vedoucími projektu
- Seznámit se se základní prací s verzovacím systémem subversion
- Seznámit se s aktuální implementací vykreslení popisků
- Implementovat vlastní mechanismus vykreslení popisků
- Implementace v jazyce JavaScript

Bonus

V případě dalšího zájmu lze v projektu pokračovat v rámci PRJ5 a bakalářské práce.

Zadavatel

Petr Vogl, e-mail: voglpetr@students.zcu.cz

Zadání pro KIV/PT

Space Traffic - Implementace mapy galaxie

Úvod do problematiky

Jsou dány XML soubory definující hvězdné soustavy. Každá soustava obsahuje jednu hvězdu, několik planet a koncové body červích děr (vstupní/výstupní). Planety a červí díry obíhají kolem hvězdy po určité trajektorii (kruhová/eliptická), která je definována v XML souboru dané hvězdné soustavy. Modely hvězdných soustav jsou pouze rovinné (2D).

Koncové body červích děr slouží k definování cest mezi hvězdnými soustavami. V XML popisech jednotlivých soustav není definováno, kam která díra vede. Definice propojení systémů se bude nacházet v samostatném souboru, jehož návrh je jedním z cílů této práce (viz dále).

Mezi hvězdnými soustavami cestují kosmické lodi, které využívají červí díry k přechodu z jedné soustavy do druhé. Doba cesty červí dírou je zanedbatelná, neuvažuje se při žádném výpočtu. Cestování v rámci jedné soustavy probíhá konstantní rychlostí. Dráha lodi je dána startovní pozicí, cílovým tělesem (planetou) a sekvencí červích děr, kterými musí loď proletět k dosažení cílové soustavy.

Zadání

- Implementovat objektovou reprezentaci hvězdných systémů (hvězdy, planety a červí díry) na základě XML schematu.
- Navrhnout a implementovat vhodnou objektovou reprezentaci mapy galaxie, která určí propojení koncových bodů červích děr v jednotlivých systémech. (Každý koncový bod může být propojen maximálně s jedním dalším koncovým bodem jiné soustavy).
- Navrhnout a implementovat algoritmus pro výpočet času příletu do cíle pro zadanou dráhu lodi a jednotlivé časové body průletů červími dírami. Algoritmus počítá s pohybem vesmírných těles. Loď tedy nesměřuje k určitému tělesu (planetě nebo koncovému bodu červí díry), ale k bodu, ve kterém se bude toto těleso nacházet v budoucnu, s cílem dosáhnutí nejkratší možné dráhy. Tato dráha musí zohlednit průlet kolem hvězdy v zadané bezpečné vzdálennosti. Součástí je i funkce pro výpočet aktuální polohy lodi při daném počátečním a cílovém bodu dráhy.

Požadavky

- Implementace v jazyku C#.
- Dokumentace v kódu v angličtině.

- Dodržování štábní kultury projektu SpaceTraffic.
- Podrobná dokumentace algoritmu výpočtu dráhy.
- Lze implementovat jako jednoduchou konzolovou aplikaci.
- Průběžné konzultace a výsledky práce.

Možná rozšíření

Cestovní rychlost v rámci jednoho systému není konstantní, ale loď polovinu cesty zrychluje a polovinu cesty zpomaluje.

Kontakty

Petr Vogl, voglpetr@students.zcu.cz

Společné zadání pro KIV/ASWI a KIV/DB2

Databázová vrstva hry Space Traffic

O Space Traffic – Co, proč, jak

Na katedře se vytváří webová hra Space Traffic typu MMORTS zaměřená na obchodování ve vesmíru. Hra bude obsahovat programovatelné prvky, jejichž funkci budou hráči moci pomocí svých uložených programů automatizovat. Herním plánem jsou dynamické mapy slunečních soustav, které jsou propojeny červími dírami. Úkolem hráče je nakupovat zboží „levně“ a prodávat „draze“. K tomuto účelu hráč vlastní (programovatelné) vesmírné lodě, jejichž prostřednictvím zboží distribuuje zákazníkům. Podrobnější informace o game designu získáte z wiki http://spacetraffic.kiv.zcu.cz/dokuwiki/doku.php?id=gamedesign

Hra má za cíl prezentovat katedru informatiky a výpočetní techniky a zároveň má dostat katedru do povědomí více středoškolských studentů a tím také podnítit jejich zájem o studium na KIV. Více informací k hlavní ideje projektu je možné získat na wiki stránkách katedry na adrese http://wiki.kiv.zcu.cz/SpaceTraffic/HomePage

Space Traffic je ASP.NET MVC 3 webovou aplikací. Serverová část je psána v jazyce C# a je nasazena na webovém serveru IIS 7. Aplikační vrstva na klientské straně je vytvářena v JavaScriptu a běží tedy ve webovém prohlížeči. Dalšími technologiemi použitými ve větší míře jsou jQuery, JSON a SVG.

Zadání

Cílem práce je vytvoření perzistentní vrstvy webové hry Space Traffic. Jedná se o návrh databáze, její implementace a implementace služeb pro zajištění perzistence objektů (CRUD++). Požadavkem je, aby byla tato vrstva vhodně oddělena od zbytku herního systému. Požadavkem je mimo jiné i vytvoření automatických testů pokrývajících implementované řešení.

Práce je určena pro tým max. 4 studentů jako společné zadání na předměty ASWI a DB2. Pro řešitele úlohy to znamená:

- docházet na pravidelné schůzky s vedoucími projektu,
- provést sběr požadavků na perzistentní vrstvu hry,
- na základě získaných informací provést analýzu možného řešení,
- aktivně se zapojit do návrhu architektury a výběru technologie,
- své řešení implementovat a otestovat (vytvořit unit testy).

Všechny tyto činnosti řešitelé provedou v úzké spolupráci s vedoucími projektu.

Jako bonus se lze zapojit do návrhu formátů pro ukládání statických dat mimo databázi, nebo navrhnou úložiště pro data mini-her.

Popis oblasti související se zadáním

Space Traffic je hra o obchodování ve vesmíru. Každý hráč má tedy svou firmu a pomocí svých kosmických lodí převáží zboží po galaxii. Lodí je více modelů s volitelnými parametry konfigurace. Galaxie je tvořena hvězdnými systémy, které jsou mezi sebou propojeny červími dírami. Hvězdný systém se skládá ze slunce, planet a červích děr. Na některých planetách se nacházejí základny, na kterých lze přistát a nakupovat/prodávat zboží. Cílem hráče je maximalizovat svůj zisk a minimalizovat náklady.

Hráčům se počítá skóre a každý hráč plní během hry achievementy na základě různých kritérií. Hráči také získávají levely na základě dosažených zkušenostních bodů.

Zboží ve hře je generováno továrnami na základnách a je ho v rámci každé planety omezené množství. Hráč může tyto továrny také vlastnit a jejich prostřednictvím také generovat zisk. Továrny hráče jsou stavěny na pronajatých pozemcích na jednotlivých základnách. Na základně může být postaveno více továren. Speciální formou "továrny"je hráčův sklad, shromažďující zboží. Umožňuje tak hráči automaticky nakupovat a prodávat zboží na základě hráčem definovaných prahových cen.

Protože hráči budou své lodě ovládat prostřednictvím svých vytvořených programů. Úkolem studentů (řešitelů této úlohy) bude také navrhnout nejvhodnější způsob ukládání těchto programů do databáze.

Každý hráč vlastní poštovní schránku, díky ní může komunikovat s ostatními hráči.

Požadavky

Přesné požadavky na perzistentní vrstvu hry budou získány samotnými řešiteli této úlohy v rámci předmětu ASWI. Zde jsou uvedeny jen rámcově požadavky na odevzdaný produkt a průběh vypracování:

- výběr vhodné metody přístupu do DB a její implementace,
- nezbytná logika na straně DB (uložené procedury + triggery), popř. ORM,
- unit testy (Mock),
- spolupráce s vedoucími při návrhu architektury,
- pravidelné schůzky s vedoucími projektu.

Použitelné technologie

PostgreSQL nebo MsSQL, Entity Framework, NHibernate, ... (cokoliv použitelného na Windows Server 2008).

Kontakty

Petr Vogl, voglpetr@students.zcu.cz

Příloha B – Plány pro akademické roky 2011-2013

Space Traffic 2011/2012

Popis projektu

Vytvoření on-line webové hry jako nástroje pro propagaci Katedry informatiky a výpočetní techniky.

Cílová skupina

Hráči hry budou převážně studenti středních škol. Hra má za cíl poskytnou hráčům krátkodobou zábavu při prezentaci na dni otevřených dveří a několik následujících týdnů po jeho skončení.

Popis hry

- Hráč začíná s jednou nákladní kosmickou lodí.
- Hráč se pohybuje z planety na planetu, mezi hvězdnými systémy pak prostřednictvím umělých červích děr.
- Loďě je třeba zásobovat palivem.
- Na každé planetě je k dispozici NPC obchodník, který nabízí a poptává zboží různého druhu. (ceny se pohybují)
- Náplní hry je meziplanetární obchod - hráč je v roli dopravce.
- Hráč je odměňován herní měnou.
- Hra obsahuje achievementy.

Cíle pro rok 2011/2012

1. Dokončit registraci a přihlašování
2. Navrhnout a zdokumentovat architekturu (server + client), vychází se z Nette frameworku.
3. Navrhnout a implementovat engine pro vykreslování objektů na straně klienta na základě prototypů (js, svg, html5).
4. Implementovat mapu hvězdného systému a galaxie.
5. Vytvořit nástroje pro vytváření map hvězdných systémů a galaxie (nemusí být implementovány jako webová aplikace).
6. Navrhnout a implementovat pohyb lodi po mapě, ovládání, provedení a zobrazení.
7. Navrhnout a implementovat podporu achievementů.
8. Navrhnout a implementovat podporu pro herní poštu, případně další formy komunikace (chat).
9. Implementace herního gui.
10. Vytvoření tutorialu.
11. Vytvoření textových popisů herních objektů.
12. Vytvoření grafiky gui (nakreslení).
13. Vytvoření grafiky pro herní objekty (nakreslení).
14. Vytvoření testů.
15. Navrhnout a implementovat zakončení hry (např. po splnění všech achievementů).
16. Vytvoření portálu pro hru a blog, komentující vývoj.

Tyto body představují kompletní první verzi hry.

Další rozšíření

- Návrh programovacího jazyku, umožňujícího vytvořit automatické ovládání hráčovy obchodní flotily (automatické vyhledávání a realizace zakázek).
- Možnost budovat základny s vlastními výrobními prostředky a konkurovat NPC a ostatním hráčům na jednotlivých planetách.

- Zakládání korporací, sdružujících hráče a jejich výrobní prostředky (je třeba dále navrhnout důsledky takového sdružování). Podpora pro vytváření miniher (služby pro komunikaci s minihrou, jejich organizace a model odměňování).
- Události v herním světě, například živelné katastrofy.
- Nápověda ve stylu "Civilopedie"- herní encyklopedie s popisem herních prvků a objektů a odkazy na reálný svět, implementováno jako wiki.
- Herní oznamování o přednáškách pořádaných KIV a dalších informacích zajímavých pro hráče.
- Vylepšování lodí pomocí komponent.

Použité technologie

JavaScript, frameworky Mono a *JQuery*.

PHP5, frameworky *Nette 2*, *Dibi* a

SimpleTest

HTML5 a *CSS3*, *SVG*.

.NET a *Java* pro vytváření vývojových a editačních nástrojů.

Space Traffic 2012/2013

Co funguje

- Game design na několik let dopředu.
- Architektura na nejvyšší úrovni - GameServer, Webové Ui, JS engine.
- Definice XML formátů pro mapu a hvězdné systémy (včetně načítání), modely lodí, komponenty.
- Podpora skriptů v C#, dynamický překlad.
- Databáze, přístup přes DAO, snadné rozšíření.
- JS engine pro vykreslování grafiky, lze rozšiřovat, je tam pár bugů, které by se mělo povést odstranit.
- Přihlašování, Registrace na velmi základní úrovni.
- Nákup a pohyb lodí v základní verzi (rozpracováno, zbývá pospojovat - léto).
- Testy na některé části jádra.
- Testovací data (assety).
- Editor hvězdných systémů.
- Nakonfigurována wiki projektu.
- Návody a dokumentace, DP 2x.

Použité technologie

JavaScript, frameworky Mono a *JQuery*.

.NET, *ASP.NET MVC 3*, *Entity Framework*, *NLog*

HTML5 a *CSS3*, *SVG*, *Less* (dotless).

Cíle pro rok 2012/2013

1. Vytvoření komunity kolem projektu (portál, blog, sociální sítě)
2. Implementace obchodu
3. Implementace misí

4. NPC
5. Implementace programování lodí
6. Implementace podpory achievementů
7. Tvorba herního obsahu
8. Balancování, testování
9. Integrace herní wiki
10. Dokončení gui pro ovládání lodí.