Manažment vývoja hier

Bc. Martin Panáček

Diplomová práca
2021/2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Martin Panáček**
Osobní číslo:	**A20200**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Management vývoje her**
Téma práce anglicky:	**Game Development Management**

Zásady pro vypracování

1. Nastudujte a popište relevantní metodiky a doporučení v kontextu vývoje her.
2. Rozeberte problematiku vývoje her v závislosti na platformě.
3. Rozveďte problematiku jednotlivých týmových rolí v rámci vývoje v herním průmyslu.
4. Zpracujte vlastní doporučení pro management vývoje her.
5. Porovnejte vaše doporučení s aktuálními a vhodně reprezentujte výsledky.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. CHANDLER, Heather Maxwell. The Game Production Handbook. 3rd ed. 5 Wall Street, Burlington: Jones and Bartlett Learning, 2013. ISBN 978-1449688097.
2. IRISH, Dan. The Game Producer's Handbook. Cengage Learning PTR, 2005. ISBN 978-1592006175.
3. HIGHT, John a Jeannie NOVAK. Game Development Essentials: Game Project Management. Thomson Delmar Learning, 2008, 284 s. ISBN 9781418015411.
4. BECK, Kent, Mike BEEDLE, Arie van BENNEKUM, et al. Agile Manifesto: The 12 Principles of Agile. Agile Alliance [online]. The Agile Manifesto Authors, 2001, , 1 [cit. 2021-04-17]. Dostupné z: https://www.agilealliance.org/wp-content/uploads/2019/09/agile-manifesto-download-2019.pdf
5. KEITH, Clinton. Agile Game Development with Scrum: Deliver Better Games Faster, On Budget And Make Game Development Fun Again. Pearson Education, May 23, 2010. ISBN 9780321670281.

Vedoucí diplomové práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. prosince 2021**
Termín odevzdání diplomové práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, Martin Panáček v.r.

ABSTRAKT

Diplomová práca sa zaoberá problematikou vývoja a manažmentu herných projektov. Hlavným cieľom tejto práce je poskytnúť súčasným a aj budúcim producentom nástroj, vďaka ktorému získajú základné povedomie o tom ako viesť herné projekty. Informácie získané z teoretickej časti testujeme v praxi pomocou dvoch prípadových štúdií, vďaka ktorým si priblížime ako sú vedené herné projekty v reálnom svete. Následne na základe výstupov z jednotlivých štúdií a teórie, sme spísali doporučenia aplikovateľné na manažment herných projektov.

Kľúčové slová: Herný vývoj, extrémne programovanie, scrum, waterfall, agile

ABSTRACT

The diploma thesis deals with the issues of development and management of game projects. The main goal of this work is to provide current and future producers with a tool that will give them a basic understanding of how to lead game projects. In the practical part, we put into use the information obtained from the theoretical part in practice using two case studies, thanks to which we will get closer to how game projects are conducted in the real world. Subsequently, based on the results of individual studies and theory, we wrote recommendations applicable to the management of game projects.

Keywords: Game development, extreme programming, scrum, waterfall, agile

Rád by som poďakoval pánovi Ing. Petrovi Žáčkovi, Ph.D., vedúcemu diplomovej práce, za jeho ochotu, čas a cenné pripomienky poskytnuté pri spracovávaní diplomovej práce. Ďalej by som chcel poďakovať svojej rodine a priateľke za pochopenie, neúnavnú podporu a trpezlivosť.

OBSAH

ÚVOD

Herný priemysel sa teší neustále väčšej popularite vďaka rýchlosti svojho rastu a množstvu hráčov po celom svete. Hry sa stávajú súčasťou každodenných životov mnohých ľudí rovnako ako iné tvorivé priemysle, či už sa jedná o filmový, hudobný, alebo o televíziu a rozhlas. V porovnaní so zmienenými priemyslami, hry predstavujú interaktívne médium, ktoré využíva kombináciu rôznych iných priemyslov. V herných produktoch v dnešnej dobe bežne nájdeme orchestriálnu hudbu, špičkových hercov, alebo kvalitne spracovaný scenár a naratív.

Témou diplomovej práce sú spôsoby a procesy vývoja herných produktov. Aj keď sa hry radia medzi kreatívne priemysle, stále sa jedná o softvér, ktorý treba vyrobiť a dodať cieľovým zákazníkom. V začiatkoch tohoto priemyslu hry predstavovali oveľa jednoduchšie produkty ako dnes. Na vývoj hier v tej dobe stačilo zopár vývojárov, ktorí dokázali doručiť hru za pár mesiacov. V dnešnej dobe, mnohé produkty vyžadujú desiatky profesionálov rôznych zameraní. Títo vývojári musia byť spolu synchronizovaní za účelom doručiť úspešný produkt a následne ho u cieľových zákazníkov predávať. S nárastom komplexnosti jednotlivých produktov narástla aj konkurencia. Denne sa vydávajú desiatky až stovky hier na rôznych platformách. Kvôli narastajúcemu rozsahu hier a konkurencii môže predstavovať tvorba väčších hier veľké finančné riziko. Z tohoto dôvodu sa začal vyvíjať tlak na procesy, ktoré vedú k dokončeniu hry. Jedným z cieľov hry je poskytnúť zábavu koncovému zákazníkovi, aby sa mohla predať. Spôsoby vývoja týchto produktov a procesy musia brať do úvahy túto kreatívnu stránku rovnako ako aj technickú.

V teoretickej časti práce rozoberáme rôzne štádiá vývoja herných produktov v závislosti na platforme. Preberáme význam jednotlivých fáz, rovnako ako aj ciele tímov. Ďalej opisujeme zodpovednosti oddelení podieľajúcich sa na vývoji. Dôležitou časťou teórie sú aj samotné procesy a metodiky, ktoré sa aplikujú na vývoj. V tejto časti si v rýchlosti preberáme waterfall model, kvôli priblíženiu, ako sa hry vyvíjali. Následne sa zameriavame na modernejšie prístupy, ktorými sú agilné metodiky. Viac do detailu rozoberáme extrémne programovanie a scrum.

V praktickej časti práce testujeme teoreticky získané znalosti v praxi na dvoch herných projektoch v podobe prípadových štúdií. Výstupy z týchto štúdií následne využívame na spísanie odporúčaní pre vedenie a vývoj herných projektov.

I. TEORETICKÁ ČASŤ

1 ZAČIATKY HERNÉHO VÝVOJA

Známky o prvej videohre siahajú do roku 1958 kde fyzik William Higinbotham vytvoril produkt, ktorý sa považuje za prvú hru [1]. Išlo o veľmi jednoduchú hru, kde hráč hral proti počítaču tenis. Tvorba videohier je stále jeden z mladších priemyslov, no neustále sa vyvíja a rastie obrovskou rýchlosťou. Predsa len, prvá hra vznikla pred 6 desaťročiami [1] a keď sa pozrieme na jej rozsah, vizuálny štýl, technológiu na akej bola postavená a porovnáme to s dnešnými titulmi, tak nájdeme obrovský a neporovnateľný rozdiel. Tento rýchly vývoj, ktorým si priemysel prešiel za pár desaťročí sa odrazil aj na cene výroby úspešnej hry. V začiatkoch vývoja sme na tvorbu hry nepotrebovali obrovské tímy ľudí špecializovaných na grafiku, animácie, dizajn, umelú inteligenciu, platformy, technickú stránku hry a testovanie. Na začiatku boli hry len hardvérové skrinky poskladané elektroinžiniermi pre špecifickú hru. Neskôr, s príchodom programovateľných hardvérových platforiem, na ktorých bolo možné spustiť viacero hier, začal vývoj arkádových automatov a domácich konzolí. S možnosťou vytvárať hry ako softvér sa z elektroinžinierov pomali stali programátori videohier [2].

V roku 1965 Gordon Moore predpovedal, že počet tranzistorov ktoré sa zmestia do čipu procesoru sa zdvojnásobí každý rok [9]. Vďaka tomuto technologickému napredovaniu dostali vývojári možnosť do hier vkladať zložitejšie funkcionality a prichádzať s inováciami, ktoré ich užívatelia vyžadovali. Trh s domácimi konzolami a počítačmi bol poháňaný touto predpoveďou [2]. Pre vývojárov to znamená, že technologické možnosti konzolí a počítačov sa zdvojnásobili každým rokom. Zvýšil sa výpočtový výkon procesorov, výkon grafických kariet, a aj pamäť. Každá generácia hardvéru priniesla nové možnosti ako vyššiu kvalitu grafiky, zvuku, 3D-renderovania, a väčší ukladací priestor.

To zároveň znamená, že každá generácia hardvéru priniesla aj väčšie náklady na vývoj hier. Z jedného programátora schopného vytvoriť hru za pár mesiacov, sa stal niekoľko členný tím vývojárov s mnohými špecializáciami, ktorému pri väčších projektoch trvá vytvoriť hru niekoľko rokov. S narastajúcou komplexnosťou sa zvýšil aj rozpočet potrebný na tvorbu hry. So zvýšením nákladov na vývoj sa zvýšil aj risk tvorby veľkého produktu. Na začiatku mohol jeden vývojár vytvoriť za pár mesiacov hru, ktorá ak neuspela, tak sa jednoducho zameral na ďalšiu. No v dnešných dňoch, ak po niekoľkých rokoch investovania peňazí do vývoja hry povedie k tvorbe neúspešného produktu, môže to znamenať aj koniec vývojárskeho štúdia a utopenie veľkej investície v neúspešnom projekte. Tým sa zvyšoval tlak na vývojárov čo veľmi rýchlo začalo viesť k dlhým nadčasom [2], ktoré sa v mnohých štúdiách stali štandardom. To viedlo k tomu, že priemerný vývojár po pár rokoch vývoja hier vyhorel a odišiel do iného priemylu [2]. Následokom bolo, že v tomto priemysle nezostali vedomosti, ktoré by dokázali predávať skúsení ľudia mladším, menej skúseným [2]. Rovnako to malo

dopad na samotnú kvalitu vytvorených hier. Veľkí investori sa začali viac spoliehať na istotu investícií do už overených frančíz a vývojári sa snažili znížiť risk doručením nižšieho počtu hodín zábavy. Obrovský nárast tímov v krátkom čase potrebných na vývoj hry, mal za následok nezvládnutie procesov spojených s manažmentom vývoja hry, čo následne viedlo k už spomenutým nadčasom a odchodom skúsených ľudí.

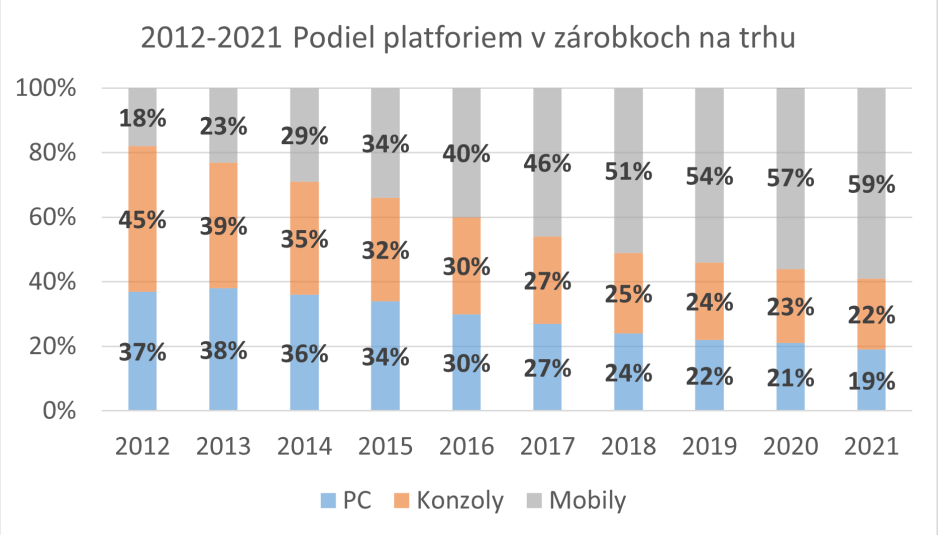
2 HERNÝ VÝVOJ V ZÁVISLOSTI NA PLATFORME

V dnešnej dobe majú vývojári veľa možností pri výbere platformy, na ktorú by chceli vydať svoj produkt. Medzi najrozíšerenejšie platformy patria počítače, konzoly, mobliné telefóny, virtuálna realita alebo streaming. Dnes sa stále môžeme stretnúť aj s možnosťou vydať hru na arkádové automaty, televíziu alebo webové prehliadače. [3]

Pri výbere správnej platformy musia vývojári zohľadniť typ hry, technológiu, rozpočet a cieľovú skupinu zákazníkov, ktorú chcú zaujať [4].

Vo všeobecnosti sú mobilné hry distribuované primárne na dve základné platformy: Android a iOS. Momentálny hardvérový výkon mobilných telefónov ponúka vývojárom široké možnosti pri výbere komplexnosti hry a jej náročnosti na výkon. Vývoj jednoduchej minihry môže stáť medzi 3 000 až 5 000 dolárov. Kďežto zložitejšia stratégia s možnosťou pre viacerých hráčov, môže stáť štúdio aj viac ako 150 000 dolárov. [4]

Komplexné konzolové a počítačové hry sú primárne doménou veľkých vývojárskych štúdií. Výkon dnešných konzolí dovoľuje vývojárom tvoriť prepracovanú grafiku, animácie a efekty, ktoré mnohokrát pripomínajú realitu. Vývoj takejto hry si vyžaduje veľa času, zdrojov a financií, preto sa táto úroveň kvality začala nazývať AAA (Triple A). Náklady na vývoj takejto hry môžu predstavovať viac ako 1 milión dolárov. [4]



Obr. 2.1 Podiel platforiem v zárobkoch na trhu. Dáta prevzaté z [5].

Technológie a enginy slúžiace na vývoj hier sa môžu tiež líšiť, či už na základe platformy alebo typu hry, ktorú vyvíjame. V dnešnej dobe moderné enginy pokrývajú širokú škálu platforiem, ktoré podporujú. Napríklad herný engine Unity podporuje Windows, Android, iOS, Linux, herné konzoly a mnohé ďalšie [6]. Vďaka možnosti vyvíjať na viaceré platformy zároveň, dokážu vývojári ušetriť veľa prostriedkov a času. Samozrejme, nie je to bezproblémové, keďže počas vývoja pre viacero platforiem môžu

vývojári naraziť na mnoho problémov. Jedným môže byť potreba optimalizácie na rôzne druhy hardvéru [7]. Ak vyvíjate hru primárne pre konzoly, no chcete ju vydať aj pre mobilných uživateľov, pravdepodobne budete mať veľa práce s optimalizáciou vizuálu a výkonu. Taktiež je potreba spolupracovať s rôznymi platoformami na dodaní hry, ktoré môžu mať pre publikovanie verzie rôzne požiadavky a pravidlá [7].

Obrázok 2.1 ukazuje, ako sa menil podiel na zárobkoch v hernom priemysle na jednotlivých platformách od roku 2012 do roku 2018. Je dôležité poznamenať, že roky 2018, 2019, 2020 sú odhadnuté.

Na obrázku môžeme taktiež vidieť, ako sa podiel mobilnej platofrmy na trhu zväčšoval počas jednotlivých rokov. Tento nárast má za následok dostupnosť výkonných smartfónov a internetové pokrytie v krajinách po celom svete. [8]

3 FÁZY HERNÉHO VÝVOJA

Vývoj hry môžeme rozdeliť do niekoľkých základných fáz, ktoré pomáhajú tímu definovať termíny miľníkov a sústrediť sa na ciele. Väčšinu fáz majú herné projekty spoločnú, no niektoré sa môžu líšiť na základe veľkosti projektu, jeho potrieb alebo platformy.

3.1 Koncept

Hlavným cieľom konceptovej fázy je odpovedať si na najzákladnejšie otázky ohľadom hry. Počas tejto fázy sa tím snaží vyprodukovať rôzne nápady na hru a prototypmi overiť, či daná hra má potenciál byť zábavná a či má zmysel na nej ďalej pracovať. Súčasťou konceptovej fázy je výber konceptu, ktorý sa bude ďalej rozpracovať v pre-produkčnej fáze. Výber konceptov sa zvykne nazývať aj "proof of concept". [10]

3.2 Pre-produkcia

Pre-produkcia má viacero cieľov, medzi ktoré patrií rozpracovanie návrhu hry z konceptovej fázy. Zistiť, čo je v hre zábava, urobiť prvé odhady na cenu vývoja, vytvoriť postupy a procesy na tvorbu hry [11]. Počas tejto fázy pracuje celý tím na definovaní rozsahu hry pomocou vnášania väčšieho detailu do potrebných mechaník a funkcionalít, ktoré hra bude obsahovať. Napríklad art pracuje na definícii vizuálneho štýlu, čo zahŕňa vytváranie prototypov charakterov, prostredia, alebo užívateľského rozhrania [10]. Dizajnéri sa snažia navhrnúť základné piliere, na ktorých bude hra stáť a na ktorých sa bude stavať počas produkcie. Technické oddelenia počas tejto fázy podporujú ostatné tímy pri tvorbe prototypov, nástrojov, alebo sa samé pripravujú na produkciu rôznymi výskumami ohľadom technológii, ktoré sa použijú.

3.3 Produkcia

Táto fáza je najdrahšia zo všetkých [10]. Počas produkcie celý tím pracuje na výrobe hry a jej potrebných častí na základe dohôd a výstupov z predchádzajúcich fáz. Zmena základných návrhov hry počas tejto fázy môže predstavovať vysoké náklady, preto je dôležité, aby výtupy z pre-produkcie boli čo najkvalitnejšie. Hlavnou výzvou tejto fázy je maximalizácia efektivity jednotlivých tímov a minimalizovanie množstva zahodených častí hry [2], aj keď mnohokrát býva bežné, že sa mnoho vecí zahadzuje [10].

3.4 Alpha

Počas tejto fázy by mala byť hra hrateľná od začiatku do konca. Niektoré časti hry nie sú vo finálnej podobe, ako napríklad animácie, zvuky, assety, rovnako ako aj niektoré

funkcionality. Cieľom tejto fázy je doladiť zostávajúce časti a pomaly pripraviť hru na vydanie. [12]

3.5 Beta

V Bete by mal byť obsah hry dokončený a tím by sa mal sústrediť primárne na jej stabilizáciu a optimalizáciu [13].

3.6 Post-produkcia

Akonáhle je dokončená produkcia a hra je vydaná, vývoj pokračuje s menším počtom vývojárov s primárnym cieľom uržiavať hru alebo tvorby ďalšieho obsahu (DLC). [14]

Taktiež sa môže uskutočniť retrospektíva počas ktorej tím diskutuje, čo vo vývoji fungovalo a čo nie. [14]

3.7 Soft launch

Táto fáza sa používa primárne pri mobilnom vývoji F2P hier. V podstate sa jedná o vydanie testovacej verzie hry do špecifických malých území pred samotným globálnym vydaním. Cieľom tohoto vydania je monitorovanie dát od cieľových zákazníkov a možnosť následných úprav hry pred vydaním do celého sveta. [15]

4 ODDELENIA POTREBNÉ NA VÝVOJ HRY

Hry pozostávajú z mnohých častí na ktoré je potreba veľa špecializácií. V tejto kapitole si zhrnieme tie najzákladnejšie.

4.1 Herný dizajn

Pozícia herného dizajnu pokrýva širokú škálu špecializácií. Vo väčších tímoch sa tieto špecializácie delia na jednotlivé pozície, ako napríklad naratívny dizajnér zaoberajúci sa herným príbehom, gameplay dizajnér alebo systémový dizajnér. Kďežto v menších tímoch, musí jeden dizajnér zvládať aj viacero špecializácií. V nasledujúcich riadkoch si prejdeme, aké ciele by mal sledovať herný dizajnér.

- Vízia. Základnou víziou je, že dizajnér by si mal položiť otázku, čím sa táto hra líši od iných hier, aký je jej účel, jej štýl a aký zážitok má hráčovi poskytnúť.
- Herné mechanizmy. Najjednoduchšie veci, ako strieľanie, sú mechanikou hry. Napríklad v hernej sérii Call of Duty je veľmi dôležité, aby mechanika strieľania bola veľmi prepracovaná, keďže patrí medzi najpoužívanejšie.
- Herný príbeh. Počas písania príbehu pre hru je potrebné brať do úvahy postavy, prostredie, mechaniku, hrateľnosť a hudbu.
- Systémy. Každá hra má sériu rôznych systémov. Napríklad systém odmien, kde hráč získa cenu alebo mince, systém poškodenia, kde by mal prísť o život. Tieto systémy musia byť navrhnuté tak, aby sa podporovali.
- Celkový pocit z hry. Hudba a grafika menia pocit z hry, sú tu však aj iné veci, ktoré si hráč priamo nevšimne. Napríklad, kde je hra ľahká alebo ťažká, kde je hráč frustrovaný, kde sa viac snaží a ako vyzerajú nepriatelia. To všetko ukazuje, že herný dizajnér musí mať znalosti takmer v každej oblasti, od fyziky po ekonómiu, matematiku, literatúru a históriu. [16]

Je veľmi dôležité, aby herný dizajnér dal všetko na papier. Dokument, ktorý herný dizajnér pripraví, sa nazýva herný dizajnový dokument (GDD). Primárnou motiváciou pre tvorbu dokumentu je komunikácia vízie s tímom a memorovanie informácií [17]. Z tohoto dokumentu následne čerpá celý tím informácie o navrhnutej časti hry. Napísať GDD nieje jednoduché, pretože rovnako, ako musí byť tento dokument zrozumiteľný pre ostatných dizajnérov v tíme, musí pokrývať dostatok detailov aj pre programátorov, alebo artistov, ktorí si vytvárajú predstavu o hre.

Rovnako dôležité ako písanie kvalitných GDD, je aj udržiavať ich aktuálne. Tvorba hry je iteratívny proces, preto sa takáto dokumentácia môže veľmi rýchlo stať zastaralou. Následne môžu nastať problémy či už pri testovaní, kde testerský tím bude hodnotiť funkcionalitu na základe zastaralých požiadaviek. Prípadne pri implementácii novej funkcionality, ktorá či už priamo alebo nepriamo pracuje so zastaranou dokumentáciou a teda časom nieje jasné, ako má v skutočnosti táto funkcionalita fungovať.

4.2 Herný programátor

Hlavnou úlohou herného programátora je implementácia požiadaviek dizajnérov. Oddelenie programátorov môže mať veľa špecializácií. Tieto špecializácie závisia od toho, akú hru vyvíjate alebo aké technológie k vývoju používate [18]. Najpoužívanejšími enginami na vývoj hier sú v dnešnej dobe Unity a Unreal Engine [19], avšak veľa tímov si tvorí aj vlastný engine. V tom prípade potrebujú aj programátorov enginu.

Základným typom je gameplay programátor. Tento programátor sa zameriava primárne na implementáciu herných mechaník.

Ďalším typom môže byť programátor užívateľského rozhrania, ktorý zapája interaktívne elementy a pridáva im funkcionality.

Ak vyvíjate hru, ktorá obsahuje multiplayer, určite potrebujete serverového a sieťového programátora.

V prípade, že od NPC herných charakterov očakávate samostatné správanie, potrebujete programátora umelej inteligencie.

Rovnako, ak vaša hra má komplikované animácie, budete potrebovať aj programátora na animačný systém. Programátorských špecializácií je mnoho a každý tím je zložený z rôznych kombinácií, ktoré sú závislé od potrieb hry, ktorú vyvíjate.

Je dôležité, aby programátori dokumentovali svoju implementáciu do takzvaných technických dizajnových dokumentov. Tieto dokumenty špecifikujú, ako by sa mali implementovať požiadavky, rovnako ako aj nástroje, ktoré sa na implementáciu použijú [20].

4.3 Art

Ďalším oddelením v procese vývoja hry je art. Primárnou zodpovednosťou artistu je vytvárať vizuálne assety do hry. Existujú dve hlavné kategórie artu, 2D a 3D art.

Artista zameraný na 2D dokáže nájsť uplatnenie v dvojrozmerných rovnako ako aj v trojrozmerných hrách. V dvojrozmerných hrách jeho primárnou zodpovednusťou môže predstavovať kreslenie spritov a textúr pre charaktery, prostredie, alebo efekty. V trojrozmerných hrách si môže nájsť uplatnenie pri kreslení konceptov, textúr pre 3D

modely, rovnako ako pre 2D elementy užívateľského rozhrania.

Artisti zameraní na 3D môžu zastupovať niekoľko rôznych pozícií podľa špecializácie. Medzi tie môže patriť samotné modelovanie, alebo tvorba efektov. Artisti špecializovaní na modelovanie sa delia na ďalšie špecializácie, ako napríklad modelovanie charakterov, prostredia, propov, alebo nastavenie osvetlenia. [21]

4.4 Animátor

Animátori sú zodpovední za vnesenie života a dynamiky do inak statických postáv. Cieľom animácie je často poskytnúť realizmus a zlepšiť zážitok z interakcie s hrou pridaním života a emócie do akejkoľvek scény alebo postavy. Animátori taktiež môžu pracovať na rôznych druhoch animácií, jednou z nich môže byť napríklad animovanie mimiky tváre. Bežnou prácou animátorov je študovanie, ako sa ľudia a zvieratá pohybujú, aby ich mohli čo najlepšie napodobniť vo svojich animáciách. [21]

4.5 Audio

Úlohou zvukového skladateľa je produkovať soundtracky, hlasové herectvo a rozprávanie, ako aj aplikovať zvukové efekty. Zvukové efekty sú zvuky, ktoré sa vydávajú pri rôznych reakciách, napríklad pri skoku alebo náraze do prekážky. [16]

4.6 Tester

Testeri sú dôležitou súčasťou každého tímu. Ich primárnou úlohou je zaručiť kvalitu hry pomocou hľadania a reportovania problémov, ktoré neodpovedajú akceptačným kritériám požiadaviek. Okrem samotného testovania a hľadania chýb sú testeri výbornou posilou pri dávaní spätnej väzby na jednotlivé funkcionality, alebo vizuál. V počiatočných fázach projektu môžu pomôcť rôznymi prieskumami ohľadom konkurenčných hier, ktoré pomôžu dizajnérom vyriešiť základné problémy počas pre-produkčnej fázy. [16]

4.7 Producent

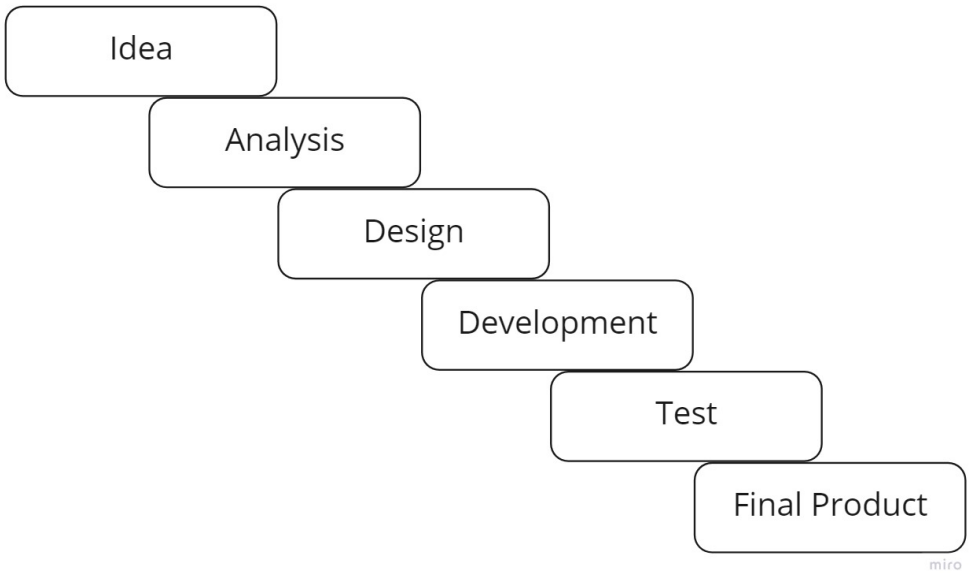
Herný producent zabezpečuje, aby sa proces a postup pri navrhovaní, vývoji a vydaní hry skutočne uskutočnil. Producenti pomáhajú tímom počas vývoja hry s odstraňovaním prekážok, komunikáciou alebo dodržiavaním termínov. [22]

Je to rola, ktorá je v každom štúdiu iná. Producent niekedy veľmi úzko spolupracuje s kreatívnym riaditeľom a hlavným dizajnérom a má veľa kreatívneho vkladu a kontroly. V iných prípadoch môže byť ich úloha čisto organizačná, alebo môžu byť najatí ako nezávislí pracovníci neskoro počas vývoja, aby pomohli dostať projekt pod

kontrolu a nasmerovať ho na cieľovú čiaru. Nie vždy je potreba producenta na projektoch s menšími tímami. Pri nižšom počte vývojárov je bežné, že niektorý z nich na seba čiastočne prevezme produkčnú rolu. Pri projektoch s desiatkami alebo stovkami zamestnancov môže byť niekoľko producentov. [22]

5 WATERFALL

V začiatkoch herného vývoja sa používal waterfall model [2], ktorého autorom je Winston Walker Royce [23]. Jedná sa o sequenčný vývojový proces, pri ktorom vývoj produktu prechádza cez viaceré fázy (obr. 5.1). Tieto fázy sú koncept, analýza, dizajn, implementácia a testovanie. Každá fáza viedla k ďalšej fáze, ktorá bola nákladnejšia ako predchádzajúca fáza [2]. Počiatočné fázy (koncept, analýza, dizajn), spočívali v písaní plánov a dokumentácie o tom, ako by mala finálna hra vyzerať [23]. Samotná práca vývojárov začala až v implementačnej časti. Poslednou fázou bola integrácia všetkých softvérových komponentov a testovanie hry. Cieľom každej fázy bolo znížiť riziko pred prechodom na drahšie fázy [2].



Obr. 5.1 Vodopádový model. Obrázok prevzatý z [24].

Ako môžeme vidieť podľa obrázku (obr. 5.1), tento model predpokladá, že všetky požiadavky a návrhy smerujúce k úspešnej hre je možné spísať ešte pred začatím práce. Čo znamená, že kompletný návrh produktu, jeho funkcionalít a analýza všetkých riskov sa deje ešte pred samotnou implementáciou.

To nám môže napovedať, že koncept tohoto modelu nepočíta so zmenami počas vývoja, keďže sa celý čas riadi požiadavkami a dokumentáciou spísanou na začiatku. Rovnako po celý čas vývoja pracujeme na produkte s otáznou kvalitou, kvôli neskorému testovaniu. [25]

Vo vývoji s použitím vodopádového modelu sa produkt stával funkčný až v posledných fázach vývoja. Pri zohľadnení ceny výroby počítačovej alebo konzolovej hry v dnešnej dobe, môže byť nebezpečné a finančne nepraktické evaluovať takýto drahý produkt v posledných konečných fázach.

Pri zbieraní požiadaviek a písaní základných kreatívnych dokumentov, sa môže javiť koncept hry zábavný. Veľká šanca je, že aj dobre premyslený návrh hry sa nestretne s požiadavkami cieľových zákazníkov. Tento scenár nastáva s nedostatočnou znalosťou uživateľských potrieb na začiatku vývoja, rovnako ako aj s chýbajúcou možnosťou meniť kreatívne návrhy na základe informácií zozbieraných po testovaní jednotlivých implementácií.

Projekty, ktoré používajú vodopádový model obvykle vykazujú minimálny postup v hľadaní zábavy v prvých dvoch tretinách vývoja projektu. Tím vlastne nevidí hru až takmer do konca projektu, kedy sa pomaly doimplementujú všetky časti a hra sa začne stabilizovať. Vtedy väčšinou prichádzajú nápady ako vylepšiť daný produkt. Bohužiaľ, táto fáza je asi najhoršou fázou na implementáciu veľkých zmien, ktoré by mohli posunúť zábavu v hre na vyššiu úroveň a tým zväčšiť aj jej úspech. Na konci vývoja stojí projekt pred konečným termínom, na základe ktorého môžu byť nápady na vylepšenie odmietnuté. [2]

6 AGILNÝ VÝVOJ

Problémy v softvérovom vývoji spôsobené nevhodnými prístupmi, donútili rôznych expertov venovať väčšiu pozornosť spôsobu, akým sa vyvíja softvér. Na základe tohoto podnetu začalo vznikať veľa publikácií a príručiek o tom, ako viesť softvérové projekty. Jedna z nich bola o inkrementálnom vývoji za pomoci iterácií. Kde každá iterácia predstavovala drobné časti jednotlivých fáz developmentu namiesto toho, aby tieto fázy boli časovo rozdelené do celého vývoja produktu, ako tomu bolo pri waterfall metodológii. [2]

Jednotlivé iterácie môžu trvať jeden alebo niekoľko týždňov, no obsahujú analýzu, dizajn, programovanie, testovanie a prípadnú opravu chýb. Mnoho vznikajúcich iteratívnych a inkrementálnych metodík bolo známych ako "lightweight methods"(ľahké metódy), až pokiaľ v roku 2001 skupina expertov nenazvali tieto metodiky agilnými. [2]

Výsledkom tohoto zhromaždenia, na ktorom pomenovali tieto metodiky, bolo vytvorenie agilného manifestu, ktorý sumarizuje hodnoty a princípy týchto metód.

6.1 Agilný manifest

Agilný manifest znie následovne.

Odhaľujeme lepšie spôsoby vývoja softvéru tým, že ich aplikujeme a tým, že pomáhame druhým, aby ich robili. Vďaka tejto práci, sme dospeli k nasledovným hodnotám:

- ***jednotlivci a interakcie*** *nad procesmi a nástrojmi,*
- ***fungujúci softvér*** *nad obsiahlou dokumentáciou,*
- ***spolupráca so zákazníkom*** *nad rokovaním o zmluve,*
- ***reakcia na zmenu*** *nad nasledovaním plánu.* [26]

Zatiaľ čo v položkách vpravo je hodnota, my si vážime položky vľavo viac [26].

6.1.1 Jednotlivci a interakcie nad procesmi a nástrojmi

S narastajúcimi tímami na herných projektoch sa vytvárali komplikovanejšie hierarchie. Začali sa písať veľké a detailné dizajnové dokumenty, ktoré mali predpovedať každú jednu drobnosť. Všetky tieto zmeny boli považované za nevyhnutné pri riadení týchto projektov.

Herný vývoj vyžaduje vývojárov s rôznymi triedami špecializácií [2]. Zoberme si ako príklad hrdinu z populárnej hry Overwatch. Na tvorbu takéhoto hrdinu potrebujeme rôzne oddelenia, medzi ktoré patrí napríklad dizajn, art, ľudí špecializovaných na efekty, animátorov, ľudí zaobstarávajúcich zvuky alebo programátorov.

Je veľmi dôležité, aby tieto rôznorodé disciplíny spolu dokázali spolupracovať a komunikovať bez prekážok, ktoré môžu spôsobiť komplikované projektové hierarchie. Agilné metodiky sa týmto problémom venujú zdola. Jedným zo spôsobov je podpora tímov, ktoré sú schopné samé vyriešiť mnohé z týchto problémov. Spravujú najmenšiu úroveň detailov, ale nie najvyššiu úroveň. Tým umožňujú vedeniu sústrediť sa na celkový obraz a smer produktu. [2]

6.1.2 Fungujúci softvér nad obsiahlou dokumentáciou

Projektová dokumentácia je veľmi dôležitá dokonca až nevyhnutná. Napríklad technická dokumentácia rôznych implementácií pomôže novým členom vývojového tímu sa rýchlejšie zoznámiť s projektom. Rovnako ako aj dizajnový dokument poskytne obraz mnohým programátorom alebo dizajnérom, ako by mali špecifické funkcionality fungovať. Dokumentácia by mala slúžiť na komunikáciu, čo uz vieme o danom projekte. Nie na detailný rozpis funkcionalít, o ktorých tím nič nevie, pretože ešte neboli odskúšané. [2]

6.1.3 Reakcia na zmenu nad nasledovaním plánu

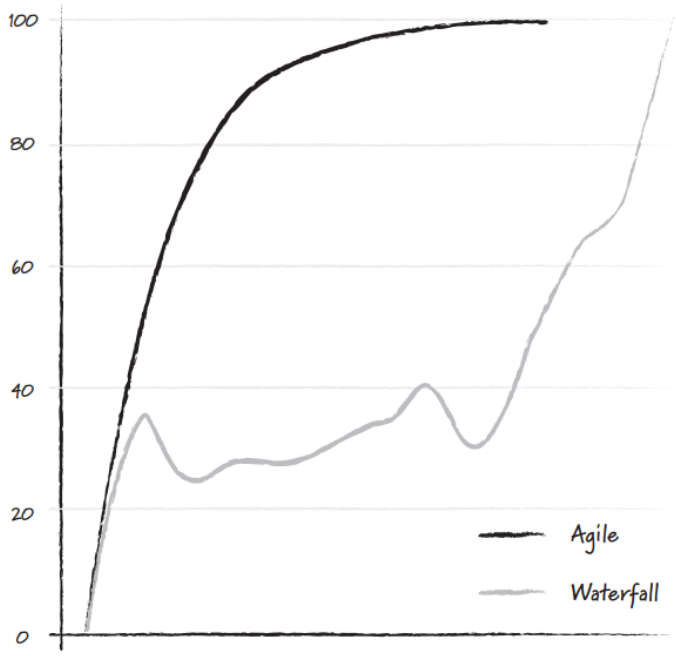
Agilným prístupom je plánovanie toho, čo je známe, a opakovanie toho, čo nie je známe. Rozširujúce sa projektové tímy, neustále sa zvyšujúca zložitosť funkcií a hardvéru, motivovali manažérov obrátiť sa k čoraz podrobnejšiemu plánovaniu. Definované procesy sa najlepšie uplatnia, keď máme istotu o technológii požadovanej projektom a dobre pochopené požiadavky, o ktorých vieme, že sa implementujú do úspešnej hry. [2]

6.2 Hľadanie zábavy

Hľadanie zábavy je dôležitou súčasťou vývoja každého herného projektu. Čím skôr tím nájde zábavu v hre, tým skôr môže robiť rozhodnutia, ktoré zábavu podporujú. K dosiahnutiu spomínaných cieľov napomáha iteratívny a inkrementálny vývoj. Počas jednotlivých iterácií, keďže obsahujú všetky fázy vývoja, by mal byť tím schopný doručiť funkčný a testovateľný build. Čo je pri hernom vývoji obzvlášť dôležité, pretože to, či je hra zábavná, zistíme až s ovládačom v ruke. [2]

Na obrázku (Obr. 6.1) môžeme vidieť porovnanie, kedy je zábava objavená pri tvorbe hry s použitím waterfall modelu a agilného vývoja.

Ako môžeme vidieť, projekty, ktoré sú vyvíjané pomocou waterfall obvykle vykazujú minimálny postup v hľadaní zábavy v prvých dvoch tretinách vývoja. Tím vlastne nevidí hru až takmer do konca projektu, kedy sa pomaly doimplementujú všetky časti a hra sa začne stabilizovať. Testovanie hry je ideálna príležitosť na analýzu nedostatkov a priestoru na vylepšenie. V prípade waterfall modelu je testovacia fáza asi najhoršou



Obr. 6.1 Nájdenie zábavy pri waterfall a agile metodikách.
Obrázok prevzatý z [2].

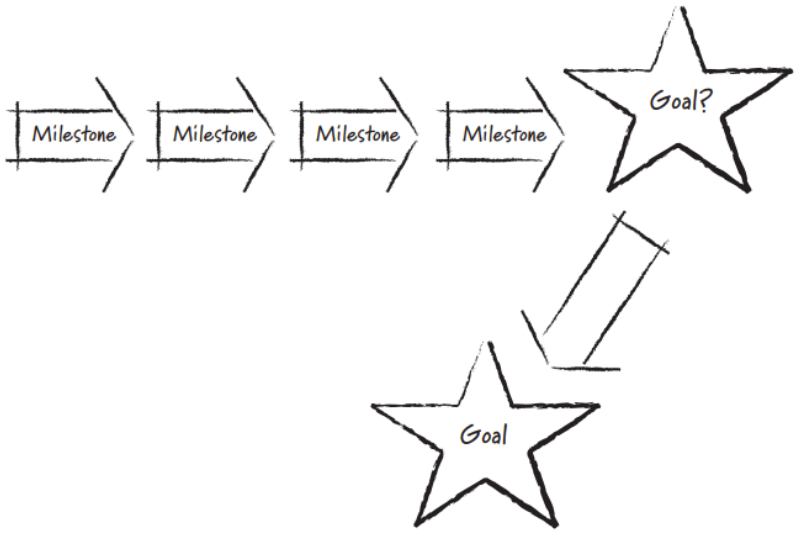
fázou na robenie veľkých zmien, ktoré by mohli posunúť zábavu v hre na vyššiu úroveň a tým zväčšiť aj jej úspech. Na konci vývoja stojí projekt väčšinou pred konečným termínom, na základe ktorého môžu byť nápady na vylepšenie odmietnuté.

Naopak agilný prístup sa sústreďuje primárne na hodnotu. Toto sa deje počas jednotlivých iterácií kedy sa projekt pomaly zväčšuje a pridávajú sa jednotlivé funkcionality. Na konci každej iterácie môžu byť tieto funkcionality otestované vo funkčnej verzii a tím sa môže rozhodnúť, či je daná funkcionalita hotová alebo ešte potrebuje vylepšiť. Taktiež môže nastať rozhodnutie, že sa daná funkcionalita odstráni z hry, ak neprináša potrebnú hodnotu a tím sa sústredí na dôležitejšie implementácie. [2]

Vďaka tomuto prístupu môže tím pri tvorbe projektu predísť plytvaniu práce, času, peňazí a motivácie na projektoch, ktoré nebudú komerčne úspešné. Hra, ktorá nie je zábavná by mala byť spochybňovaná na každom kroku. [2]

Hra prechádza evaluáciou na konci každej iterácie. Na základe výstupu z tejto evaluácie sa upravujú ciele pre nasledujuce iterácie. Každých niekoľko iterácií sa verzia hry dostane do takzvaného release miľníku, čo znamená, že ciele pre tento miľník sú dosiahnuté a hra sa privedie takmer do stavu pripraveného na vydanie. [2]

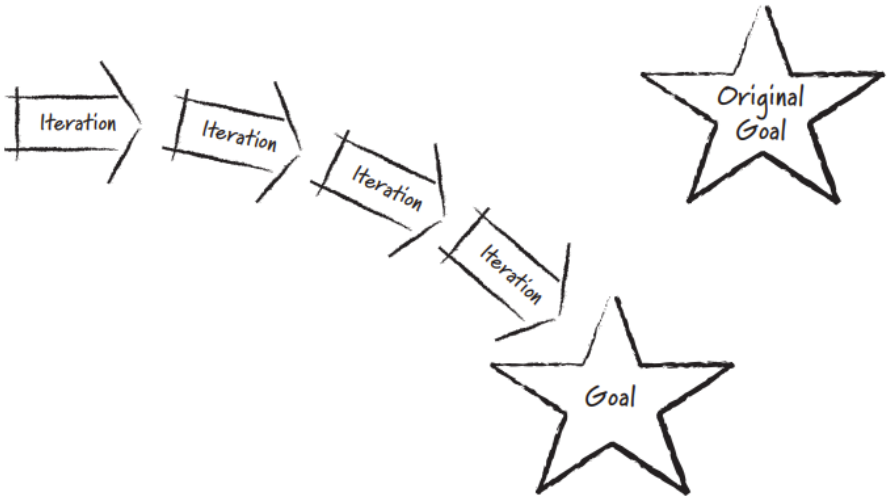
Agilné projekty sa nevyhýbajú plánovaniu. Prijímajú plánovacie postupy, ktoré vývoju projektu umožňujú zmeny. Vo väčšine waterfall projektov, míľniky vedú projekt k cieľu definovanému ešte pred začiatkom vývoja. Akonáhle projekt dosiahne tieto ciele, tak si všetci začnú uvedomovať, že vlastne chceli byť niekde úplne inde a môže to vy-



Obr. 6.2 Pri waterfall projektoch si tím na konci uvedomí, že chceli byť s hrou niekde inde. Obrázok prevzatý z [2].

zerať ako na obrázku (Obr. 6.2). Ako sme si už povedali, projekt v tejto fázi väčšinou nemá čas ani peniaze. [2]

Na obrázku (Obr. 6.3) môžeme vidieť ako sa dokáže projekt využívajúci agilné procesy prispôsobovať počas jednotlivých iterácií. Na konci každej iterácie tím hodnotí implementované funkcionality, vďaka čomu dostane lepšiu predstavu o vyvíjanej funkcionalite na pravidelnej bázi.



Obr. 6.3 Agilný projekt sa dokáže prispôsobovať a meniť ciele každú iteráciu. Obrázok prevzatý z [2].

6.3 MoSCoW prioritizácia

MoSCoW prioritizácia patrí medzi populárne metódy slúžiace na riadenie požiadaviek. Skratka MoSCoW predstavuje štyri kategórie priorít. [27]

- Must-have. Ako už názov napovedá, iniciatívy alebo funkcionality označené touto prioritou, sú povinné na doručenie. Predstavujú základné potreby pre daný projekt alebo vydanie.
- Should-have. Tieto iniciatívy alebo funkcionlaity sú nevyhnutné pre produkt alebo vydanie, no nie sú životne dôležité. Ak sa vynechá niektorá z iniciatív, projekt stále funguje. Od must-have sa odlišujú primárne tým, že môžu byť naplánované na budúce vydanie bez toho, aby to ovplyvnilo to súčasné.
- Could-have. Iniciatívy could-have nie sú nevyhnutné pre základnú funkciu produktu. Avšak v porovnaní s iniciatívami should-have majú oveľa menší vplyv na výsledok, ak sa vynechajú.
- Won't-have. Kategória môže spravovať očakávania týkajúce sa toho, čo tím nezahrnie do konkrétneho vydania (alebo iného časového rámca). Zaradenie iniciatív do kategórie will-not-have je jedným zo spôsobov, ako pomôcť zabrániť zväčšovaniu rozsahu. Ak sú iniciatívy v tejto kategórii, tím vie, že nie sú prioritou pre tento konkrétny časový rámec. [27]

7 EXTRÉMNE PROGRAMOVANIE

Medzi agilné metodológie používané v hernom vývoji patrí aj aj Extrémne Programovanie [28].

Extrémne programovanie bolo vyvinuté, aby dokázalo pomôcť vývojárom reagovať na neustále zmeny požiadaviek a dizajnov jednotlivých funkcionalít. Rovnako ako ostatné agilné metodiky, extrémne programovanie nám nedáva priame postupy ako vyvíjať hry, no ponúka nám hodnoty a princípy na základe ktorých môžu jednotlivé tímy reagovať na výzvy spojené s vývojom [29].

Jedným z kľučových predpokladov extrémneho programovania je, že náklady na zmenu môžu byť v priebehu času konštantné [29]. Tento predpoklad sa dá docieliť napríklad krátkymi iteráciami, frekventovaným testovaním, elimináciou defektov hneď po ich vzniku, alebo pracovaním so spätnou väzbou od uživateľa.

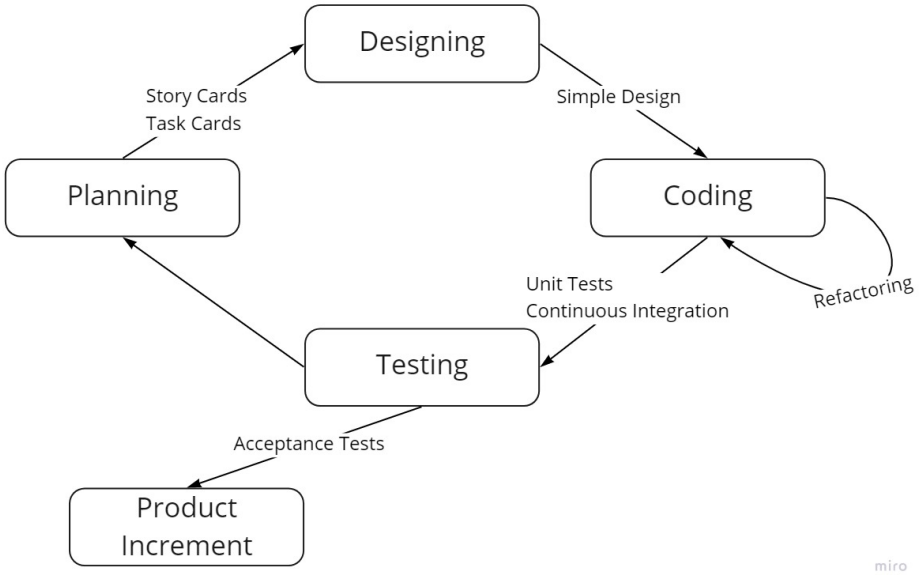
Extrémne programovanie zahŕňa viacero praktík, medzi ktoré patrí aj

- vývoj riadený testami (TDD),
- kontinuálny dizajn,
- párové programovanie,
- kontinuálna integrácia (CI),
- rýchle vydania,
- práca so zákazníkom počas celého vývoja,
- plánovacia hra [29].

7.1 Vývoj riadený testami (TDD)

Programovanie riadené testami je vývojový proces pri ktorom programátor, ešte predtým ako začne pracovať na konkrétnej funkcionalite, napíše unit test. Tento unit test má pokrývať akceptačné kritéria na implementáciu špecifickej funkcionality a pri prvom behu by mal zlyhať. Následne ako tento test zlyhá, programátor môže začať implementovať danú funkcionalitu. Implementácia z pohľadu programátora je dokončená, keď unit test, ktorý napísal na začiatku prejde úspešne. Funkcionalita sa teda môže presunúť do testovania. [30]

Tento proces má mnoho výhod, no automatizované testovanie je asi najväčšou z nich. Automatizované unit testy ukazujú, že každá časť hry funguje podľa návrhu. Vývojári môžu tieto testy jednoducho pustiť a veľmi rýchlo poznať stav jednotlivých



Obr. 7.1 Produktová iterácia. Obrázok prevzatý z [29].

funkcionalít. V začiatkoch vývoja sa tento proces môže zdať zbytočný, ak hra obsahuje len zopár funkcionalít. No s narastajúcou komplexnosťou produktu sa zvyšuje potreba regresného testovania. Bez automatizácie to znamená, že testeri musia po zmenách vo verzii ručne otestovať jednotlivé funkcionality, či fungujú na základe požiadaviek a ich funkcionalita nebola ovplyvnená zmenami. [30]

7.2 Kontinuálny dizajn

Kontinuálny dizajn sa snaží uchovávať koplexitu dizajnovanej funkcionality na základe aktuálnych potrieb hry. To znamená, že sa neimplementuje komplikovaný systém, pokiaľ to hra nepotrebuje. Samozrejme, dizajnéri musia zaobstarať kvalitný a detailný dizajn zjednodušenej verzie funkcionality. Rovnako k tomu pristupujú aj programátori, ktorí jednoducho implementujú funkcionalitu tak ako bola nadizajnovaná, nič viac. Ak sa ukáže, že daná funkcionalita prináša hodnotu a oplatí sa do nej investovať viac času, môžeme ju rozšíriť ďalšou iteráciou. Tento prístup urýchľuje celý proces dizajnovania aj implementácie. Taktiež eliminuje riziko implementácie komplexnej funkcionality alebo systému, pri ktorom sa po otestovaní zistí, že nieje zábavný a v hre sa nepoužije. Vďaka tomu sa môžu vývojári sústrediť na hľadanie zábavy v hre a predísť podobným prípadom. Súčasťou tohoto procesu je aj refaktoring, ktorý udržuje kvalitný a zdravý kód. Keď tím lepšie pochopí dizajn alebo keď jednoduchšia implementácia nevyhovuje rastúcim potrebám funkcionality, často zistí, že program by bol zdravší alebo jednoduchší, keby kód vyzeral inak. Refaktoring je spôsob ako programátori môžu udržiavať kód flexibilný. [31]

7.3 Párové programovanie

Párové programovanie je proces, pri ktorom pracujú dvaja programátori na jednom počítači a snažia sa vyriešiť nejaký problém. Spoluprácu týchto programátorov si môžeme prirovnať k jazde v aute, jeden je vodič a druhý navigátor. [32]

Navigátor sa nezameriava na chyby, ktoré môže vodič urobiť. Namiesto toho premýšľajú o kontexte, v ktorom pracujú. Kladú si otázky ohľadom vhodnosti kódu, ktorý píšu, jeho kvalite a jednoduchosti. [32]

Kvalita a jednoduchosť kódu, ktorý vytvárajú dvaja programátori, je zvyčajne vyššia v porovnaní s tým, čo by vytvorili samostatne. Tento kód vyššej kvality sa mení jednoduchšie a bezpečnejšie ako kód napísaný iba jedným členom z dvojice. [32]

Herní programátori zvyčajne robia drobné dizajnové rozhodnutia pri implementácii. Tieto rozhodnutia môžu viesť kvalitu herného zážitku na vyššiu úroveň. Pár minút premýšľania dvoch kreatívnych ľudí môže viesť ku kvalitným výsledkom. Párovanie podporuje tieto rýchle diskusie bez toho, aby sme museli zhromažďovať ľudí na veľké stretnutia alebo niekoho vyrušovať. [32]

Párovanie môže pomôcť budovať pevné a šťastné tímy vytváraním dôvery prostredníctvom riešenia spoločných problémov [32].

7.4 Kontinuálna integrácia (CI)

Kontinuálna integrácia predstavuje testovanie verzie a jej integráciu po určitom množstve zmien, niekoľko krát denne. Jednou z najlepších praktík je, mať na tento účel dedikovaný počítač alebo použiť služby tretej strany. Táto postupná integrácia po zmenách na jednotlivých verziách, pomáha tímu rýchlejšie odhaliť príčiny defektov. Testy by sa mali púšťať pri každej integrácii z dôvodu jednoduchšej analýzy problému. Ak pri predchádzajúcom zostavení verzie prešli všetky testy na 100%, no pri poslednom zostavení niektoré testy zlyhali, vieme, že problém bude v zmenách medzi jednotlivými verziami. Tento prístup šetrí tímu veľké množstvo času, pretože beh automatizovaných testov by mal zabrať menej času ako keby to mali testeri ručne testovať. [29]

7.5 Rýchle vydania

Ďalšou z praktík extrémneho programovania sú rýchle vydania. Podľa tejto praktiky by mali vývojári vydať na začiatku jednoduchú hru, ktorú na základe spätnej väzby budú rozširovať. Každá aktualizácia by mala byť čo najmenšia, aby jej tvorba nezabrala veľa času a aby sa spetná väzba z dát alebo od uživateľov lepšie vyhodnocovala. Taktiež to pomáha vývojovému tímu udržiavať vysokú kvalitu verzie. Nie vždy môže ísť projekt do produkcie po pár mesiacoch. Rovnako nie vždy sa dajú vydávať aktua-

lizácie v denných až mesačných cykloch. Obvykle nejaký čas trvá, pokiaľ sa spracujú požiadavky, vytvoria návrhy na implementáciu a vykoná sa implementácia samotná. [29]

Na prvý pohľad môže sedieť táto praktika primárne na monetizačný model F2P, kde cieľom vývojárov je vytvoriť hru ako službu. No môže byť efektívne využitá u každého projektu. Jednotlivé vydania nemusia byť priamo ku zákazníkom no môžu slúžiť ako projektové miľníky pri ktorých sa tím zamyslí nad kvalitou a smerom produktu, ktorý vytvárajú.

Keď sa zamyslíme nad týmto prístupom v kontexte extrémneho programovania a spojíme ho s ďalšími praktikami, zistíme, že nám dokáže pomôcť s lepšou prioritizáciou user stories, takže aj prvotné verzie hry so sebou nesú hodnotu. Testovanie zredukuje množstvo defektov, takže niesú potrebné dlhé testovania pred jednotlivými miľníkmi. [29]

7.6 Práca so zákazníkom počas celého vývoja

Jednoducho povedané, cieľom tohoto prístupu je, aby sa hra dostala do rúk cieľovým hráčom už v počiatočných fázach vývoja. Vďaka tomu prístupu môže tím robiť kvalitnejšiu prioritizáciu funkcionalít a lepšie plánovať do budúcnosti. [33]

Ďalšou víziou tohoto prístupu je veľmi blízka spolupráca dizajnéra s programátorom počas implementácie funkcionality. Vďaka tomu sa úsilie programátora sústreďuje tam, kde prinesie najväčší úžitok, dosiahne najlepšie kompromisy a umožní dizajnérovi povedať, že implementácia je zatiaľ dostačujúca. [33]

Kooperatívny prístup k implementácii funkcií vytvára silné putá dôvery medzi zákazníkom a vývojárom, čo je vždy dobré pre tím [33].

7.7 Plánovacia hra

Hlavný plánovací proces extrémneho programovania sa nazýva plánovacia hra (planning game). Plánovacia hra je stretnutie, ktoré sa koná raz za iteráciu, zvyčajne raz týždenne. Umožňuje rýchlo určiť rozsah ďalšieho vydania produktu, kombináciou priorít na základe hodnoty a technických odhadov. Jedným z cieľov stretnutia je udržiavať technické a obchodné rozhodnutia v súlade. Každá osoba na stretnutí má určité zodpovednosti. Primárne sa tieto zodpovednosti delia na zodpovednosti zákazníka a tímu. [29]

7.7.1 Zodpovednosti zákazníka

Členovia tímu zastupujúci zákazníka počas tohoto stretnutia rozhodujú o nasledujúcich veciach.

- Rozsah. Koľko problémov sa musí vyriešiť, aby systém prinášal hodnotu koncovým zákazníkom? Títo ľudia musia vedieť rozhodnúť, čo je málo a čo je príliš.
- Priorita. Ak máte možnosť výberu funkcionalít, ktorá to bude?
- Zloženie vydaní. Koľko alebo ako málo je potrebné urobiť, aby sa firme darilo so softvérom lepšie ako bez neho? Intuícia vývojára v tejto otázke môže byť úplne mylná.
- Dátum vydania. Aké sú dôležité dátumy, počas ktorých by prítomnosť softvéru znamenala veľký rozdiel? [29]

7.7.2 Zodpovednosti tímu

Vývojári sú zodpovední za nasledujúce veci.

- Poskytovanie časových odhadov. Koľko času potrebujeme na implementáciu tejto funkcionality?
- Dôsledky. Existujú strategické obchodné rozhodnutia, ktoré by sa mali robiť len vtedy, keď sú členovia tímu informovaní o technických dôsledkoch. Vývojári musia vysvetliť tieto dôsledky.
- Procesy. Ako bude organizovaná práca a tím? Tím sa musí prispôsobiť kultúre, v ktorej bude pôsobiť. Softvér musí byť napísaný s určitou kvalitou.
- Podrobnom plánovaní. Ktoré stories by sa v rámci vydania mali urobiť ako prvé? Vývojári potrebujú slobodu najskôr naplánovať najrizikovejšie segmenty vývoja, aby sa znížilo celkové riziko projektu. V rámci tohoto obmedzenia by sa mali presúvať najdôležitejšie funkcionality na začiatok vývoja. Týmto prístupom sa znižuje šanca, že dôležité funkcionality budú musieť byť vynechané ku koncu vývoja z dôvodu časových obmedzení. [29]

7.8 Cyklus extrémneho programovania

Extrémne programovanie je iteratívne a je poháňané časovo ohraničenými cyklami. Preto je rytmus procesu extrémneho programovania kľúčový.

Extrémne programovanie má nasledujúce úrovne aktivít:

- životné cykly produktu,
- vydania produktu (release),
- iterácie,

- úlohy (stories),
- vývoj,
- spätná väzba. [29]

7.8.1 Životné cykly produktu

Nazýva sa to aj fáza prieskumu. Zahŕňa definíciu súboru funkcionalít a plánovanie. Zákazník definuje vysokoúrovňové požiadavky, ktoré sú následne uvedené ako user stories.

Stories sú primárnym výstupom aktivity tejto úrovne. [29]

7.8.2 Vydania produktu

Môže sa nazývať aj fáza záväzku. Primárny cieľ tejto fázy je plán vydania produktovej verzie alebo release plán. Následne sa tento cieľ delí na niekoľko podcieľov, ktoré môžu byť dosiahnuté za pomoci rôznych aktivít, ktoré sa v tejto fáze vykonávajú.

- Na začiatku sa diskutuje postup. V tejto časti zákazník urobí evaluáciu implementovaných funkcionalít a kvality produktu, čo môže vyústiť k definícii nových požiadaviek, alebo k prípadnej zmene existujúcich.
- V ďalšej časti zákazník prezentuje požiadavky rozdelené do malých častí (stories) a diskutuje ich s tímom. Úlohou tímu je prediskutovať spôsob, akým sa budú tieto funkcionality implementovať a s tým aj definovať riziká týchto implementácií. Následne poskytnú prvé časové odhady.
- Zákazník odprioritizuje stories na základe hodnoty, ktorú prinesú do produktu a definuje čas vydania. Pri tejto aktivite berie do úvahy riziká a estimácie komunikované s tímom.
- Vývojári sa záväzne dohodnú so zákazníkom na rozsahu funkcionalít, ktoré sa budú implementovať a na dátume ďalšieho vydania (releasu).
- V poslednom kroku tím rozdelí jednotlivé stories do iterácií aj s defektami nájdenými počas akceptačného testovania zákazníkom a začnú pracovať. [29]

7.8.3 Iterácie

Nazývané aj fázou riadenia. Na začiatku sa zíde celý tím, aby sa mohol skontrolovať pokrok a v prípade potreby upraviť plán. Zákazník prezentuje stories pre aktuálnu iteráciu a tím detailne diskutuje tieto stories, aby mali všetky informácie potrebné na implementáciu.

Hlavným cieľom tejto fázy je plán pre aktuálnu iteráciu.

Počas diskusie nad jednotlivými stories, tím detailne preberie technický prístup na implementáciu jednotlivých funkcionalít obsiahnutých v stories a vytvorí k nim potrebné úlohy.

Po prediskutovaní všetkých stories do iterácie, tím môže začat s vývojom. Cieľom tímu počas iterácie je naimplementovať všetky stories v stanovenej kvalite. [29]

7.8.4 Vývoj

Vývojári vytvoria dvojice s kolegami. Následne každá dvojica:

- verifikuje, že rozumie jednotlivým stories,
- určuje podrobný implementačný prístup zabezpečujúci jednoduchý dizajn,
- začne s TDD (testami riadeným vývojom),
- integruje kód do verzovacieho systému vo vhodných intervaloch,
- pravidelne kontroluje pokrok. [29]

7.8.5 Spätná väzba

Dvojice neustále komunikujú medzi sebou a navonok k tímu. Do komunikácie je priebežne zapojený aj zákazník. Niektoré tímy sa uchyľujú ku každodenným stretnutiam, aby rýchlo prediskutovali celkový stav tímu a v prípade potreby možnú opätovnú synchronizáciu a mikroplánovanie.

Iterácie poskytujú prehľad o celkovom stave a príležitosti na úpravu a zlepšenie procesu.

- Výsledky z iterácií môžu spôsobiť prehodnotenie priorít.
- Vývoj jednotlivých úloh môže spôsobiť prehodnotenie stories.
- Opätovný časový odhad story môže spôsobiť zmeny iterácie alebo obnovenie iterácie.
- Výsledky iterácie môžu spôsobiť zmeny plánu vydania. [29]

7.9 Role

V extrémnom programovaní je kladený dôraz na spoluprácu celého tímu, je v nepretržitej komunikácii.

Na to, aby projekt extrémneho programovania fungoval, sú však potrebné určité úlohy a osoby, ktoré tieto úlohy zastávajú. Preberajú zodpovedajúcu zodpovednosť a zodpovedajú za svoju prácu. Odporúča sa, aby sme do rolí pridelili správnych ľudí.

Role, ktoré sa ukázali ako účinné v extrémnom programovaní, sú:

- vývojár,
- zákazník,
- manažér,
- kouč. [29]

7.9.1 Vývojár

Táto rola patrí medzi najdôležitejšie v extrémnom programovaní. Predsa len, vývojár je ten človek, ktorý implementuje potrebné funkcionality. Aj keď sa menom táto rola podobá klasickému vývojárovi z hociakej inej metodológie, XP definuje pre vývojára určité práva a zodpovednosti, za účelom dobrého fungovania tejto metodológie.

Zodpovednosti

Zodpovednosti vývojára definujú prácu, ktorá sa od neho očakáva za účelom zmysluplného fungovania tímu a vývoja. Medzi tieto zodpovednosti môžu patriť časové odhady stories, definovanie úloh zo stories, časové odhady úloh, písanie unit testov, spúšťanie unit testov, alebo refaktorovanie kódu. [29]

Zručnosti

Rovnako ako zodpovednosti sa od vývojára očakávajú určité zručnosti. Medzi tieto zručnosti patrí párové programovanie, komunikácia, udržovanie jednoduchosti, programovanie len toho, čo je potrebné. [29]

7.9.2 Zákazník

V extrémnom programovaní je rola zákazníka rovnako dôležitá ako rola vývojára, pretože je to zákazník, kto by mal vedieť, čo programovať, zatiaľ čo vývojár by mal vedieť programovať.

Zručnosti

Medzi zručnosti zákazníka patrí písanie stories do potrebného detailu, rozvíjanie postoja tímu k úspechu projektu, rozhodnutia o rozsahu požadovaných funkcií, prioritizovanie týchto funkcií, alebo písanie funkcionálnych testov. Síce zákazník prioritizuje backlog a píše požiadavky pre nové funkcionality, je veľmi dôležité si uvedomiť, že nekontroluje projekt. [29]

Zákazník nemusí byť len jeden človek. Môže to byť aj tím ľudí s rôznymi rolami ako sú napríklad produktový manažeri, marketéri, biznis analytici a podobne. [29]

7.9.3 Manažér

Primárne zodpovednosti manažéra v extrémnom programovaní sú nasledujúce.

- Definovanie pravidiel plánovacej hry, oboznámenie tímu a zákazníka s pravidlami plánovacej hry, monitorovanie plánovacej hry.
- Plánovanie a exekutíva stretnutí.
- Zúčastniť sa s tímom časového odhadovania, aby mohol poskytnúť spätnú väzbu o tom, ako sa realita zhodovala s ich predchádzajúcimi odhadmi. To na úrovni tímu aj jednotlivcov, čo by im do budúcna malo pomôcť s kvalitnejšími odhadmi.
- Uistiť sa, že tím pracuje podľa aktuálnych priorít, keď postupuje cez iterácie s potvrdeným plánom a potvrdenou funkcionalitou.
- Sledovanie chýb funkčného testovania.
- Sledovanie skutočného času stráveného každým členom tímu.
- Schopnosť získať všetky požadované informácie bez rušenia tímov. [29]

7.9.4 Kouč

Rola kouča je rolou káždého jedného člena v tíme. No v prípade, že tím je v extrémnom programovaní nový, odporúča sa mať človeka špecializovaného na túto rolu. [29]

U kouča je veľmi dôležitá znalosť praktík a metodík extrémneho programovania a následná schopnosť predať tieto znalosti tímu. Jeho primárnym cieľom je vytrénovať tím, aby sa stal sebestačný v používaní týchto praktík. Jeho zodpovednosti môžeme rozdeliť do bodov.

- Hlboká znalosť extrémneho programovania a schopnosť aplikácie do projektu.

- Identifikácia praktík extrémneho programovania, ktoré pomôžu tímu v prípade akéhokoľvek problému.
- Pozorovanie tímu v tichosti a zasiahnutie len vtedy, keď sa predvída závažný problém.
- Trénovanie tímu k sebestačnosti.
- Byť pripravený kedykoľvek pomôcť. [29]

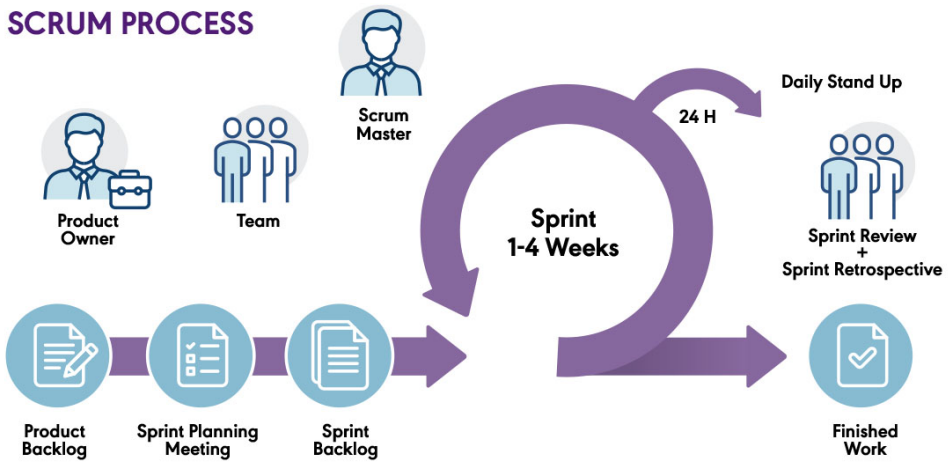
8 SCRUM

Scrum je spôsob riadenia vývoja softvéru, v rámci ktorého môžu ľudia riešiť komplexné problémy a zároveň efektívne dodávať produkty najvyššej možnej kvality [34].

Scrum sa používa od začiatku 90. rokov. Táto metóda predstavuje súbor rôznych prístupov, procesov a nástrojov, ktoré sa môžu použiť na vývoj herných produktov a zlepšenie tímovej efektivity. [35]

Scrum pozostáva zo scrum tímov a ich pridružených rolí, ceremonií, artefaktov, a pravidiel. Každý zo spomenutých komponentov slúži na špecifický účel [35].

V Scrume sa na vytvorenie pravidelnosti používajú predpísané ceremónie. Všetky ceremónie sú časovo ohraničené, takže každá má maximálne trvanie [35]. Tieto ceremónie sú podrobnejšie opísané v nasledujúcich kapitolách.



Obr. 8.1 Scrum proces. Obrázok prevzatý z [34].

8.1 Role

Scrum má tri primárne role: product owner, scrum master a členov vývojového tímu [35].

8.1.1 Vývojový tím

Vývojový tím sú ľudia, ktorí robia prácu spojenú s tvorbou hry. Na prvý pohľad to môže vyzerať, že je tím poskladaný z inžinierov, no nie vždy tomu tak je. Tím môže byť zložený zo všetkých druhov ľudí vrátane dizajnérov, rozprávačov, programátorov, artistov a podobne. [35]

Vývojový tím musí byť schopný samoorganizácie, aby mohol robiť rozhodnutia potrebné na dokončenie práce. Samoorganizácia sa v tomto ohľade neberie ako nerešpektovanie organizácie, ale skôr ako zosilnenie postavenia ľudí, ktorí majú potrebné vedomosti na riešenie problémov. [36]

Zodpovednosti tímu môžu pozostávať aj z nasledujúcich.

- Dokončenie naplánovanej práce pre šprint.
- Zabezpečenie transparentnosti počas šprintu. Toto býva dosiahnuté pomocou denných stretnutí, nazývaných taktiež denný scrum. Denný scrum poskytuje transparentnosť práce a dáva členom tímu vyhradené miesto, kde môžu hľadať pomoc, hovoriť o pokroku, alebo upozorniť na problémy. Scrum master môže pomáhať s vedením tohoto stretnutia, ale v konečnom výsledku je to zodpovednosťou vývojového tímu. [36]

8.1.2 Product owner

Agilné tímy sú svojou povahou flexibilné a pohotové a je zodpovednosťou product ownera zabezpečiť, aby poskytovali čo najväčšiu hodnotu. Produkt owner zastupuje biznisovú stranu, čiže prezentuje vývojovému tímu najdôležitejšie funkcionality. Dôvera s vývojovým tímom je veľmi dôležitá. [36]

Produktový vlastník by nemal len rozumieť zákazníkovi, ale mal by mať aj predstavu o hodnote, ktorú scrum tím zákazníkovi prináša. Produktový vlastník tiež reprezentuje potreby ostatných zainteresovaných strán v organizácii. [36]

Hlavné zodpovednosti product ownera sú nasledovné.

- Spravovanie scrum backlogu. Product owner je zodpovedný za obsah backlogu a jeho prioritizáciu. To však nemusí znamenať, že product owner je jediný, kto vkladá nové stories do backlogu. Mal by vedieť o všetkom, čo je v backlogu a ostatní ľudia, ktorí pridávajú stories do produktového backlogu, by mali zabezpečiť komunikáciu s product ownerom.
- Správa vydania. Pre product ownera je dôležité vedieť, kedy veci môžu a mali by byť uvoľnené.
- Manažment zainteresovaných strán. Do každého produktu bude zapojených mnoho zainteresovaných strán, od používateľov, zákazníkov, správy a vedenia organizácie. Produktový vlastník bude musieť spolupracovať so všetkými týmito ľuďmi, aby efektívne zabezpečil, že vývojový tím prináša hodnotu. To môže znamenať veľké množstvo riadenia a komunikácie. [36]

8.1.3 Scrum master

Scrum master je rola zodpovedná za zabezpečenie hladkého chodu scrumu v tíme. V praxi to znamená, že pomáha product ownerovi definovať hodnotu, vývojovému tímu dodať hodnotu a celému scrum tímu pomáhať v neustálom zlepšovaní. Popri tom neustále mentoruje vývojový tím a zvätšuje povedomie o scrume. Pomáha tímu odstrániť všetky prekážky a zabezpečiť hladký priebeh vývoja. [36]

Medzi zodpovednosti scrum mastera patria nasledujúce.

- Transparentnosť. Scrum master má za úlohu zabezpečiť, aby scrum tím fungoval transparentným spôsobom. Toto zahŕňa napríklad vytváranie story máp a aktualizáciu stránok dokumentácie s nápadmi z retrospektív.
- Empirizmus. Základom pre scrum a agilné prístupy je myšlienka, že najlepším spôsobom plánovania je robiť prácu a učiť sa z nej. Empirický proces nie je jednoduchý a vyžaduje si od scrum mastera, aby koučoval scrum tím pri rozdeľovaní práce, popisoval jasné výsledky a kontroloval tieto výsledky.
- Samoorganizácia. Vývojový tím sa pri prechode na scrum musí naučiť samoorganizácii. Keďže to nieje jednoduchý prechod, je úlohou scrum mastera, aby danému tímu pomáhal v tomto smere.
- Hodnoty. Scrum definuje hodnoty odvahy, zamerania, odhodlania, rešpektu a otvorenosti pretože vytvárajú prostredie bezpečia a dôvery. Toto prostredie je dôležité pre prosperitu agility. Dodržiavanie hodnôt je zodpovednosťou každého v scrum tíme, ale scrum master preberá aktívnu úlohu pri povzbudzovaní a pripomínaní dôležitosti týchto hodnôt každému. [36]

8.2 Release

Release alebo vydanie je súborom šprintov, ktorých cieľom je priniesť hru s novými funkciami do stavu, v ktorom by mala byť pripravená na vydanie. Typické vydanie trvá dva až štyri mesiace. Tempo vydávania je podobné ako tempo míľnikov na typickom hernom projekte. [2]

Dĺžka releasu tiež môže závisieť od potrieb produktu. Napríklad ak potrebujete posielať publisherovi funkčný build na konci každého mesiaca, releasy môžu predstavovať aj mesačné miľníky.

Verzia hry v stave takmer pripravená na vydanie môže znamenať mnoho vecí. Definícia podľa poučiek je, že by mala byť hrateľná pre potencionálnych zákazníkov, no nie nevyhnutne pripravená na vydanie s plným pokrytím všetkých požiadaviek [2]. V

rámci viacročného projektu by vydania vedúce k dokončenej hre, mali mať pravidlá o demo kvalite jednotlivých verzií. Samozrejme, stav verzie hry pri jednotlivých vydaniach môže byť taktiež definovaný na základe dohody medzi vedením projektu. [2]

Vydania stanovujú dlhodobé ciele pre tím a zainteresované strany. Vyžadujú zvýšenú úroveň kvality a ladenia, čo znižuje veľkú mieru neistoty ohľadom práce, ktorú treba urobiť pri odoslaní hry. Vydania začínajú plánovacím stretnutím, ktoré stanovuje hlavné ciele hry. Plán vydania riadi ciele pre každý šprint. [2]

8.3 Šprint

Srdcom scrumu je šprint. Šprint si môžeme predstaviť ako časový rámec pozostávajúci väčšinou od dvoch týždňov do jedného mesiaca. Dĺžka šprintu sa väčšinou určuje na základe potrieb a schopností tímu. Počas tohoto časového rámca sa pracuje na funkcionalitách, ktoré boli naplánované do tohoto šprintu. [2]

Nový šprint začína ihneď po ukončení predchádzajúceho šprintu. Tím a zainteresované strany stanovia cieľ na stretnutí plánovania, ktoré sa uskutočňuje na začiatku šprintu. Postup práce tímu je zdieľaný na denných scrum (standup) stretnutiach. Na konci šprintu tím demonštruje svoj pokrok na šprint review stretnutí. Po review tím uskutoční retrospektívu, kde diskutuje o tom, ako tím spolupracoval počas šprintu a hľadá vylepšenia pre nadchádzajúce šprinty. [2]

8.4 Plánovanie šprintu

Na začiatku šprintu sa tímy stretnú s vedením projektu, aby naplánovali ďalší šprint. Plánovanie šprintu si vyžaduje dve stretnutia: stretnutie o stanovení priorít a stretnutie plánovania šprintu. Stretnutie o stanovení priorít pripravuje alebo upravuje produktový backlog a identifikuje potenciálny cieľ šprintu. Stretnutie plánovania šprintu vytvorí backlog šprintu definujúci prácu, ktorú sa tím zaväzuje dokončiť do ďalšej kontroly šprintu. [2]

8.4.1 Prioritizácia backlogu

Cieľom tohoto stretnutia je definícia funkcionalít s najvyššou prioritou v produktovom backlogu a ich následný výber do šprintu.

Na začiatku stretnutia product owner odprezentuje funkcionality s najvyššou prioritou. Tím počas tejto prezentácie musí porozumieť každej jednej story. Je to výborná príležitosť pýtať sa na otázky ohľadom dizajnu (herného dizajnu, technického dizajnu, artu a podobne). Ak napríklad funkcia vyžaduje, aby hlavná postava skočila, môžu byť otázky o tom, ako sa súčasná animácia a fyzika používa v hre. Táto diskusia identifikuje

detaily návrhu a implementácie, napríklad či ide len o fyziku, animáciu alebo spojenie oboch. [2]

Môže sa stať, že niektoré stories s vysokou prioritou sú príliš komplexné, aby boli implementované počas jedného šprintu. V tomto prípade by mal tím rozdeliť stories na menšie celky. [2]

Následne tím prediskutuje približný cieľ pre šprint vybraním stories z produktového backlogu, ktoré následne vložia do šprint backlogu.

8.4.2 Plánovanie šprintu

Po identifikácii cieľov, tím prediskutuje jednotlivé stories do detailu a vytvorí potrebné tasky. Dokončenie taskov povedie k splneniu akceptačných kritérií v daných stories. Na tomto stretnutí by sa mali podieľať okrem samotného tímu aj ďalší doménoví experti, ktorí môžu v prípade potreby pomôcť tímu či už s prediskutovaním implementácie alebo estimáciou práce (napríklad multiplayer programátor alebo zvukár). [2]

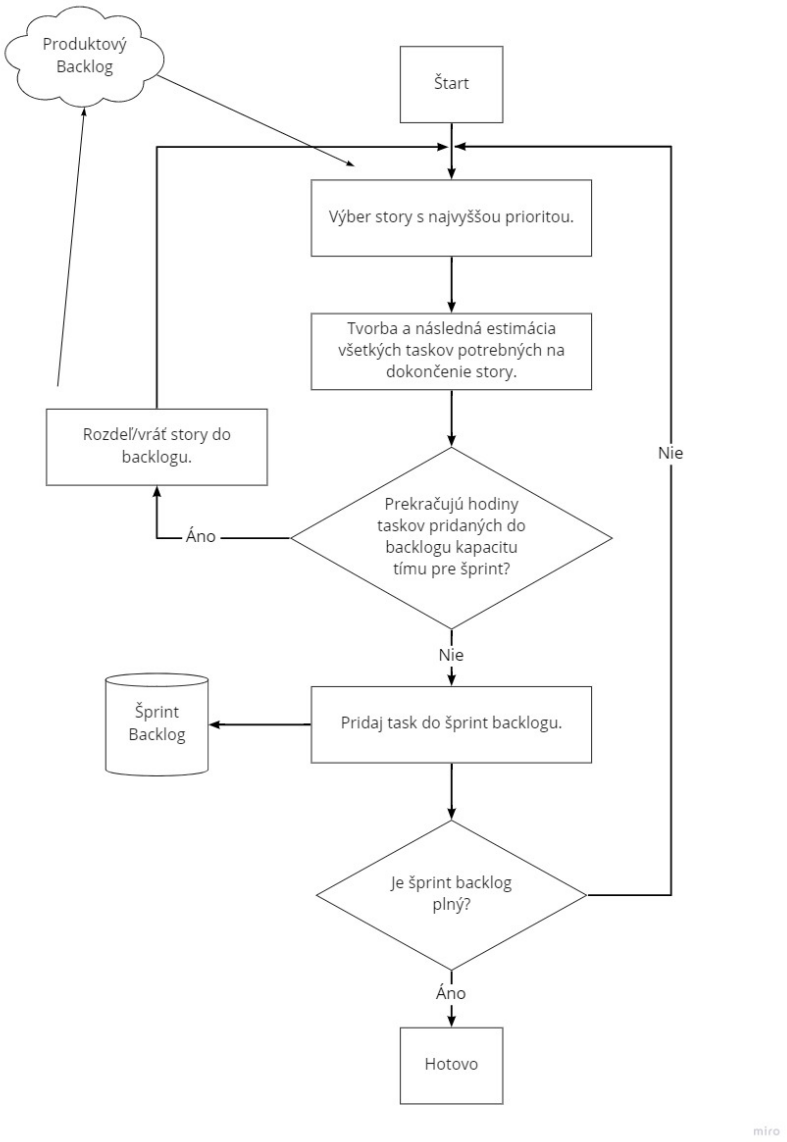
Na začiatku stretnutia pomáha scrum master definovať obmedzenia, ktoré by mohli ovplyvniť schopnosť tímu zaviazať sa k cieľu šprintu. Medzi tieto obmedzenia môžu patriť napríklad dovolenky, zdieľanie ľudí s inými tímami a podobne. Schopnosť tímu zaviazať sa k práci je založená predovšetkým na ich minulom výkone. Najlepšie sa to dá zistiť preskúmaním toho, čo tím dokázal doručiť v minulých šprintoch.

Následne tím začne diskutovať implementačné detaily pre jednotlivé stories. Účasť product ownera na tomto stretnutí je veľmi dôležitá, pretože môže poskytnúť tímu dôležité informácie v prípade potreby. Akonáhle sú odkomunikované akceptačné kritéria a požiadavky na dokončenie funkcionalít, tím začne vytvárať tasky na základe potreby a zároveň ich časovo odhadovať. Tieto tasky predstavujú čiastočné operácie alebo kroky potrebné na doručenie stories. Časové odhady taskov prebiehajú v rámci jednotlivých disciplín. Ak má tím napríklad troch programátorov, odhadujú programátorské úlohy spoločne. Naopak, ak je v tíme iba jeden, odhadne sám všetky programovacie úlohy. [2]

Na spoločné estimovanie sa používajú rôzne techniky, jednou z najznámejších je planning poker.

Na odhadovanie taskov sa môžu použiť rôzne prístupy, ako napríklad odhad v hodinách, story pointov, man days, a podobne. Tento odhad sa uvádza v ideálnom čase, čo je čas, ktorý by mala úloha zabrať bez prerušenia alebo akýhkoľvek problémov. To znamená, že osemhodinová úloha nie je to isté ako jednodňová. Splnenie osemhodinovej úlohy zvyčajne trvá viac ako jeden kalendárny deň. Dôvodom je, že dni vývojárov sú plné prerušení, problémov a rozhovorov. [2]

Odhady pre veľké tasky sú menej presné ako odhady pre malé tasky. Limit veľkosti



Obr. 8.2 Tok vytvárania šprint backlogu. Obrázok prevzatý z [2].

tasku je ľubovoľný, ale 16 hodín je rozumná hranica pre veľkosť, kedy je task potreba rozdeliť na menšie. Môže sa stať, že tím nemá dostatok informácií na rozdelenie takéhoto tasku. Namiesto toho môžu vytvoriť zástupný task s väčším odhadom, kým nebudú pripravení na úlohe pracovať a nebudú vedieť viac informácií. [2]

Ako môžeme vidieť na obrázku (obr. 8.2), akonáhle je story rozdelená na tasky, počet naestimovaných hodín potrebných na dokončenie šprintu sa navýši o hodiny potrebné na dokončenie danej story. Celkový počet sa potom porovná so zostávajúcimi hodinami, ktoré sú k dispozícii v backlogu šprintu. Ak má šprint backlog priestor na hodiny, ktoré nová story pridá, tím sa zaviaže dokončiť túto story. Samozrejme, každá špecializácia na projekte musí byť naplánovaná v rámci svojich možností.

Ak by sa náhodou nová story nevošla do backlogu šprintu, je k dispozícii jedna z troch možností. Najprv sa story vráti do produktového backlogu a nahradí ho ďalšia menšia story. Druhou možnosťou je rozdeliť pôvodnú story na menšie časti. Vďaka rozdeleniu môže tím identifikovať časť pôvodnej story, ktorá sa hodí do šprintu. Treťou možnosťou je odobrať nejakú story z backlogu šprintu, aby sa nová položka zmestila. Product owner pomáha tímu s rozhodovaním, ktoré riešenie je najlepšie. [2]

Neodporúča sa úplne zaplniť alokáciu tímu pre šprint do poslednej hodiny. Je bežné, že sa počas šprintu objavia problémy alebo úlohy s ktorými tím jednoducho nepočítal pri plánovaní. V prípade, že tímu zostane nejaký voľný čas na konci šprintu, môžu ho využiť na dolaďovanie funkcionalít, alebo môžu začať pracovať na úlohách v ďalšom šprinte, ak je šprint backlog pripravený. [2]

8.5 Denný scrum

Primárnym cieľom tohoto stretnutia, ktoré sa opakuje na dennej báze je zabezpečiť, aby scrum tím zdieľal prehľad o tom ako šprint napreduje. Tím sa taktiež informuje o nečakaných prekážkach. Počas stretnutia členovia tím odpovedajú na otázky ako: Čo som urobil včera? Čo budem robiť dnes? Mám nejaké prekážky, ktoré ohrozujú naplánovanú prácu?

Odporúča sa, aby toto stretnutie bolo krátke a všetky diskusie boli ponechané buď na iné ceremónie alebo po stretnutí, aby nezainteresovaný členovia diskusie nestrácali čas. Nieje špecifikovaný presný čas, no odporúča sa, že by meeting nemal presiahnuť 15 minút.

Na dennom scrume sa účastní vývojový tím, scrum master a v prípade potreby aj product owner. [37]

8.6 Šprint review

Hlavným cieľom stretnutia je demonštrácia dokončenej práce počas šprintu, oslavovanie, vyzdvihovanie úspechov a dohoda nad ďalšími krokmi.

Počas stretnutia vývojový tím prezentuje svoju dokončenú prácu a zbiera spätnú väzbu od product ownera. Následne, product owner na základe tímového postupu môže meniť priority v produktovom backlogu.

Toto stretnutie sa odohráva na konci šprintu a mal by sa na ňom zúčastniť celý scrum tím, product owner a obchodne zainteresované strany. [37]

8.7 Retrospektíva šprintu

Cieľom retrospektívy šprintu je pochopiť, čo fungovalo a čo nefungovalo počas posledného šprintu a následne tieto lekcie pretaviť do riešení. Tieto zlepšenia by mali posunúť tímovú spoluprácu vpred a pomôcť efektívnejšie doručovať výsledky.

Toto stretnutie sa koná vždy na konci šprintu a mal by sa ho účastniť scrum tím. [37]

II. PRAKTICKÁ ČASŤ

9 PRÍPADOVÉ ŠTÚDIE

Praktická časť sa zaoberá dvomi prípadovými štúdiami a spísaním doporučení na základe teoretických znalostí a výstupov z týchto štúdií. Projekty a spoločnosti, na základe ktorých boli napísané tieto študie boli vybrané kvôli viacerým faktorom. Prvý z nich je vývoj hier na rozdielne platformy. Prvá štúdia sa zaoberá vývojom na mobilnú platformu a druhá na počítače. Ďalším je história a vek jednotlivých spoločností. Mobilná spoločnosť už má za sebou určitú históriu a má svoju kultúru, ktorú sa počas opisovaného časového obdobia snaží vylepšiť. Spoločnosť zameraná na tvorbu počítačových hier je na trhu len jeden rok v čase písania tejto práce a pracuje na svojej prvej vlastnej hre pre investora.

Práve rozdiely jednotlivých spoločností či už v projektoch, skúsenostiach tímu, alebo histórie a kultúry dokážu poskytnúť kvalitný základ na skúmanie procesov a metodík použitých na vývoj herných projektov.

10 MOBILNÁ HRA

Prvý projekt na ktorý sme robili prípadovú štúdiu bola mobilná hra s monetizačným modelom F2P (free-to-play). Analýza projektu pokrýva len určité časové obdobie počas vývoja projektu, nie celý vývoj danej hry. Toto časové obdobie bolo vybrané preto, lebo sa v ňom odohralo veľa zmien či už v spoločnosti, v ktorej sa hra vyvíjala, ale aj pri samotnom vývoji hry.

Táto prípadová štúdia sa zameriava primárne na implementáciu scrumu a agilných metodík do herného vývoja.

10.1 Hra

Samotná hra bola vo vývoji už viac ako rok. Keďže sa jedná o F2P model, hra bola na trhu už 7 mesiacov, no neustále sa na nej pracovalo. Každá aktualizácia v pravidelnosti približne dva mesiace predstavovala pre uživateľov nové funkcionality a stabilnejšiu verziu. Medzi tieto dvojmesačné väčšie aktualizácie tím občas vydal menšiu aktualizáciu, ktorá predstavovala buď záplatu nejakých problémov, alebo balancovanie funkcionalít. Vydávanie aktualizácií pre mobilné zariadenia dávali tímu pomerne voľné ruky, keďže kontrolné doby buildu na Android platforme boli 1-2 dni [38] pre už vydanú hru, na iOS 2-4 dni [39]. Vďaka tomu tím mohol pomerne rýchlo reagovať na produkčné problémy.

Keďže sa jedná o multiplayerovú hru, samotné servery tvorili veľmi dôležitú časť hry. Rovnako ako technické riešenie implementácie niektorých základných sieťových funkcionalít tak ako aj samotné serverové riešenie nebolo najstabilnejšie. Preto sa tím musel potýkať s mnohými nečakanými problémami počas behu hry na produkcii. Medzi tieto problémy patrili nečakané pády serverov, strata pripojenia so servermi, odpojenia hráčov z hry alebo vysoký ping.

Okrem sieťových problémov hra narážala aj na mnohé klientské problémy spôsobené nedostatočne kvalitným technickým riešením.

Všetky tieto poruchy mali veľký dopad na úspešnosť hry a na uživateľské hodnotenie. Celá hra bola od prvého vydania plná podobných problémov, takže väčšina porúch bola zakorenená v základných implementáciách a robili hru od prvého vydania stratovú na finančných prostriedkoch. Okrem samotných finančných strát, hra taktiež nedisponovala ani základnými KPIs, podľa ktorých sa určuje úspešnosť free-to-play mobilnej hry. Tými sú napríklad retencia (návratnosť uživateľov) alebo ltv (life time value).

Celých 7 mesiacov sa teda tím svojou prácou snažil napraviť tieto zlé technické riešenia prerábkou rôznych základných častí hry a zároveň s hlavnými aktualizáciami dodávať hráčom nové funkcionality, aby si ich udržali v hre a zároveň, aby ich konkurencia nepredbehla.

Počas týchto vyčerpávajúcich siedmich mesiacov, sa tímu podarilo vylepšiť celkový technický stav hry, rovnako ako naimplementovať nové funkcionality. Keďže tím počas tohoto obdobia pracoval pod tlakom, mnohé nové funkcionality so sebou priniesli ďalšie problémy.

Tímu sa za toto obdobie podarilo zostabilizovať vývojovú vetvu verzovacieho systému do stavu, kedy bola viac menej pripravená s pár úpravami na vydanie. Tomu dopomohlo zavedenie nových procesov, a takzvaného technického výboru. Výbor kontroloval technické dokumenty pre implementáciu rôznych funkcionalít ešte pred tým, ako boli tieto funkcionality naimplementované.

Posledný mesiac tím pracoval na stabilizačnej aktualizácii, keď sa vedenie spoločnosti rozhodlo, že celá práca sa zahodí a všetci sa presunú na vývoj nových funkcionalít pre ďalši veľkú aktualizáciu. Teraz tím nemal dva mesiace ako zvyčajne, ale len 2-3 týždne.

Tímu sa podarilo aj po ťažkých predchádzajúcich mesiacoch doručiť sľúbenú hlavnú aktualizáciu s drobným oneskorením. Bohužiaľ, predstavil v nej aj nové problémy, ktoré sa potom s hrou ťahali ďalej.

V tomto štádiu mal tím zaužívané základné procesy, ako napríklad denná synchronizácia, ktorá bola nazývaná stand-up. Toto stretnutie bolo veľmi podobné dennému scrum stretnutiu, akurát, že sa pri ňom stretal celý tím a mnohokrát stretnutie zabralo 30 minút až jednu hodinu. Robili sa aj plánovacie stretnutia, ktoré ale nerozoberali jednotlivé funkcionality do detailu, kedy by členovia tímu odhadovali čas implementácie. Taktiež sa plánovalo pre najbližšiu aktualizáciu, takže nevyužívali výhody kratších iterácií. Tím bol zvyknutý aj na review a retrospektívne stretnutia. Tieto stretnutia sa odohrávali každý týždeň v pondelok. V prvej časti stretnutia producenti odprezentovali postup a najvyšie priority a v druhej časti mal každý člen tímu možnosť povedať niečo k retrospektíve z predchádzajúceho týždňa. Zo spätného pohľadu producentov to boli kontraproduktívne stretnutia, pretože sa na nich nachádzal celý tím, takže automaticky zabrali veľa času. No čo bolo dôležitejšie, v takom počte sa mnoho členom tímu ťažko hovorilo svoj názor na vývoj a tím.

10.2 Aktualizácia X

Po náročných mesiacoch plných pridávaní nových funkcionalít a stabilizácie hry prišlo vedenie s požiadavkou na vydanie ďalšej aktualizácie približne s dvojnásobným množstvom funkcionalít do dvoch mesiacov (rovnaký časový horizont ako pri menšom počte funkcionalít). Táto aktualizácia mala takzvane zachrániť hru, prípadne rozhodnúť o jej budúcnosti.

Okrem toho vedenie chcelo, aby počas tejto aktualizácie manažment vývoja zaviedol

agilné metodiky, presnejšie scrum a pomocou nich doručil túto verziu hry na čas a vo výbornej kvalite.

Keďže sa jednalo o aktualizáciu pred Vianocami, bolo nesmierne dôležité, aby hra bola poslaná do App Store a Google Play vo výbornej kvalite, pretože počas sviatkov sa obchody zatvárajú na dva týždne a neprímajú nové hry a ani aktualizácie existujúcich hier.

Hra podporovala široké množstvo zariadení, preto sa nie vždy dalo predpovedať, koľko problémov takáto veľká aktualizácia môže na produkcii spôsobiť. Preto vedenie prišlo s ďalšou požiadavkou, vydať Beta verziu hry ešte pred samotným vydaním finálnej hry. To pre tím predstavovalo vytvoriť úplne novú verziu hry so všetkými novými funkcionalitami minimálne dva týždne pred samotným vydaním hlavnej aktualizácie (aby sa hra stihla na produkcii poriadne otestovať a tím mal dostatok času na opravy).

10.3 Riziká

- Množstvo funkcionalít, ktoré treba naimplementovať. Napríklad, v minulosti trval jeden nový charakter do hry tímu naimplementovať približne dva mesiace, teraz bola požiadavka doručiť desať takýchto charakterov za dva mesiace (mimo iné funkcionality).

- Množstvo dizajnu pripraveného na implementáciu. Požiadavky na túto veľkú aktualizáciu vznikli až po vydaní poslednej veľkej aktualizácie, čiže dizajnéri nemali čas pracovať na nových dizajnových dokumentoch.

- Vianoce. Keďže sa na Vianoce zatvárajú obchody, nebolo možné posúvať dátum vydania. To pre tím znamenalo, že sa budú musieť primárne sústrediť na rozsah jednotlivých funkcionalít. Či už redukovanie rozsahu, alebo viesť detailné diskusie s dizajnérmi a hľadať jednoduchšie implementácie za účelom doručenia rovnakej hodnoty pre hráča.

- Doručenie beta verzie. Na začiatku vydania, to bola pre producentský tím veľká neznáma, keďže hra v bete nikdy nebola.

- Neustále menenie rozsahu aktualizácie. Vedenie sa nevedelo dohodnúť na finálnom backlogu, takže sa stále menil jeho rozsah.

- Technický stav. Tesné termíny a Vianoce nedovoľujú tímu vydať záplatu v prípade potreby. Takže ak by hra bola vydaná v zlej kvalite, mohlo by to mať fatálny dopad na jej dlhodobý úspech.

10.4 Ciele

Na základe požiadaviek vedenia, producentský tím si definoval ciele, pre najbližšie dva mesiace. Rovnako, každý cieľ mal adresovať niekoľko rizík definovaných v predchádzajúcej kapitole.

- Zvýšiť efektivitu tímu pomocou častého a krátkodobého plánovania na interácie.
- Zlepšiť komunikáciu medzi členmi jednotlivých oddelení pomocou malých tímov zodpovedných za jednotlivé funkcionality.
- Vytvoriť ideálne podmienky vývojárom na implementáciu pomocou retrospektívy.
- Implementácia všetkých funkcionalít pomocou kontinuálneho dizajnu. K tomu mala dopomôcť zvýšená komunikácia medzi tímom a vedením projektu počas plánovacích a review stretnutí.
- Tvorba dokumentácie a firemných postupov, ktoré sa týkajú manažmentu projektov.
- Úprava doteraz používaných nástrojov za účelom zlepšenia prehľadnosti a priorít.
- Budovanie agility medzi členmi tímu, dávanie im väčšej zodpovednosti a voľnosti pri riešení problémov pomocou kontinuálneho mentoringu.
- Za pomoci predchádzajúcich cieľov doručiť aktualizáciu v stanovenom dátume a kvalite.

10.5 Implementácia Scrumu

Pri implementácií scrumu bol jedným z prvých krokov pripraviť multidisciplinárne tímy na základe potrieb projektu a možností vývojárskeho tímu.

10.5.1 Procesovanie tvorby tímu

Pred samotnou tvorbou tímov si producenti sadli a spísali spoločné body, ktoré musia vyriešiť a pripraviť pred tým, ako sa tím môže pustiť do práce.

- Backlog kvalitne definovaných stories, ktoré tím bude následne implementovať.
- Tím musí mať spoločné miesto, kde môže komunikovať. V ideálnom prípade by všetci členovia tímu mali spolu sedieť v jednej miestnosti, no keďže to nebolo možné kvôli práci na diaľku, vytvoril sa tímu spoločný čet, kde mohli riešiť všetky problémy spojené s funkcionalitami, ktoré implementovali.

- Tím potreboval vlastný kanban tabuľu, kde videl prácu spojenú s ich šprintom. Vďaka tejto tabuli sa mali lepšie zamerať na doručenie cieľu šprintu. Bolo ale veľmi dôležité, aby mal tím prístup k informáciam ohľadom celého projektu a postupu iných tímov.

10.5.2 Tvorba tímov

Obecne sa dá povedať, že všetky tímy, ktoré vznikli sa dali rozdeliť do troch kategórií.

- Tímy na základe oddelení,
- tímy implementujúce meta funkcionality,
- tímy implementujúce gameplay funkcionality,
- cross tímy.

K tomu, čo znamenajú jednotlivé názvy tímov sa dostaneme neskôr. Delenie tímov bolo postavené na potrebách produktu. Je dôležité podotknúť, že životnosť týchto tímov vydržala približne 1 až 3 šprinty. Pod pojmom životnosť sa myslí, že po tomto časovom úseku sa menili členovia podľa potreby.

Pred tým, ako tím začal pracovať, všetci členovia dostali školenie o tom, čo je to scrum, aké praktiky používa, čo je cieľom scrumu, aké sú ich nové zodpovednosti a aké role scrum obsahuje. Toto školenie bolo rozdelené na dve stretnutia, kde na prvom dostal tím krátku prezentáciu a dokumentáciu a na druhom mohol klásť otázky.

V tejto časti bohužiaľ nebolo viac času na školenie jednotlivých členov. Bolo veľmi dôležité, aby sa začalo pracovať čo najskôr, preto veľmi dôležitou súčasťou každého tímu bol scrum master, ktorého rolu prebral producent. Keďže všetky tímy boli v týchto procesoch nové, scrum master mal neustále veľa práce s mentorovaním jednotlivých členov o agilných metodikách a ich prípravou na samostatnosť.

Každý z týchto tímov používal trošku odlišné agilné metodiky a procesy. Zo začiatku existovalo pravidlo, že všetci začnú podľa "knižky"a následne si prispôsobia vývojový proces podľa vlastných špecifických potrieb. Podľa producentov to malo dopomôcť k tvorbe vlastníctva tímu nad procesmi. Prispôsobovaním jednotlivých procesov si mal tím vybudovať väčšiu zodpovednosť a pochopenie, prečo tieto procesy používa. Nakoniec sa toto riešenie ukázalo ako kontraproduktívne, pretože krátke školenie členov tímu na začiatku práce na aktualizácii bolo nedostačujúce.

10.5.3 Tímy na základe oddelení

Tímy poskladané z vývojárov na základe oddelení plnili niekoľko účelov, napríklad, udržanie komunikácie medzi členmi rovnakého oddelenia, keďže veľká časť oddelenia

pracovala v jednotlivých scrum tímoch, ale aj práca na veciach, ktoré hráč automaticky nemusí vidieť. Dobrým príkladom sú programátori pracujúci na optimalizácií výkonu, alebo opravách rôznych problémov.

Procesy ktoré tímy používali boli odľahčenou verziou procesov používaných v scrum tíme. Rovnako ako ostatné scrum tímy, oddelenia tiež pracovali v šprintoch a mali svoje retrospektívy a plánovania. Na začiatku mali tieto tímy denné standupy na vyžiadanie od lídrov, no časom sa to ukázalo ako veľmi kontraproduktívne. Na tieto stretnutia muselo chodiť celé oddelenie, čiže aj ľudia zo scrum tímov, ktorí mali mimo iné aj standupy so svojimi tímami. Táto zmena zabrala nejaký čas a ukázala veľký posun v mentalite vývojárov, keďže si to sami vyžiadali počas retrospektívy.

Ďalšie procesy a praktiky záležali na oddelení. Napríklad programátori zvykli praktikovať párové programovanie, ktoré im pomohlo pri prerábke rôznych systémov.

Rolu product ownera väčšinou zaujal samotný tím líder, ktorý úzko spolupracoval s producentami alebo iným vyšším manažmentom projektu. Táto spolupráca pomohla lídrovi kvalitnejšie prioritizovať prácu. Rola scrum mastera alebo kouča bola ako v ostatných tímoch zabraná producentom.

10.5.4 Tímy implementujúce meta funkcionality

Meta sa v jednoduchosti dá opísať ako časť hry, kedy sa hráč nachádzal v menu. Čiže nebol priamo v gameplay. V tomto prípade sa už jedná o plnohodnotný scrum tím pozostávajúci z viacerých oddelení spolupracujúcich na doručení špecifickej funkcionality.

Tento tím pozostával z herného dizajnéra, UI dizajnéra, 2D artistu, UI programátor a backend programátora.

V takomto tíme existujú určité závislosti. UI dizajnér musí ideálne najskôr vytvoriť nejaké návrhy alebo mockupy pred tým ako sa UI programátor môže pustiť do práce. Rovnako ako aj backend programátor musí najskôr naprogramovať systém, ktorého dáta UI programátor zobrazí v uživateľskom rozhraní.

Vďaka tomu, že sa tento tím stretával na dennej báze a detailne spolu komunikoval pri plánovaní, mohlo sa veľa vecí robiť paralelne. UI programátor začal vytvárať uživateľské rozhranie takmer paralelne s tvorbou wireframov. Toto uživateľské rozhranie bolo automaticky pripravené na zapojenie dát z backendu, keďže odpovedalo technickému návrhu backendového programátora.

10.5.5 Tímy implementujúce gameplay funkcionality

Analogicky k meta časti hry, gameplay tímy implementovali samotné herné mechaniky používajúce sa počas hrania. Týchto tímov bolo viacero, no v tejto prípadovej štúdií stojí primárne za zmienku tím implementujúci nové charaktery.

Tento tím bol ako jediný, ktorý sa nezmenil od začiatku práce na aktualizácii až do jej konca. Práve tento fakt poskytol producentom zaujímavé informácie. Vďaka tomu, že sa členovia nemenili, tento tím sa časom stal najefektívnejším. Tu sa ukázalo, aká dôležitá je chémia tímu a procesy prispôsobené k potrebám jednotlivcov.

Pri následnom meraní efektivity jednotlivých tímov vo forme počet odhadovaných hodín dokončených počas šprintu, sa tímu vytvárajúcemu charaktery táto efektivita zvyšovala na rozdiel od ostatných tímov, ktorým tá efektivita zostala rovnaká. Okrem samotnej efektivity sa ukázalo, že počas retrospektív začali pribúdať bodovy, ktoré reflektovali spokojnosť tímu.

10.5.6 Cross tím

Jediný svojho druhu, poskladaný primárne z ľudí spolupracujúcich so všetkými tímami zároveň. Tento tím bol plánovaný jeden šprint "pozadu"v porovnaní s ostatnými tímami. Bolo to z jednoduchého dôvodu, v tomto tíme sa primárne písali a lokalizovali texty a vytvárala hudba. Takmer každá funkcionalita potrebuje lokalizované texty a ozvučenie. Vo väčšine prípadov, aby sa tieto veci mohli začať tvoriť, funkcionalita musí byť z určitej časti hmatateľná. Pre texty je napríklad dôležitý dizajnový dokument a mockup, kdežto zvuky okrem toho potrebujú ideálne aj animácie.

Členovia tohoto tímu boli pozvaní na plánovacie, review a retrospektívne stretnutia všetkých tímov. Ich účasť nebola povinná. Keďže boli dopredu informovaní o tom, na čom budú jednotlivé tímy pracovať, bolo pre nich jednoduché vybrať, na ktorých stretnutiach sa budú zúčastňovať.

Tento tím sa stretával na standupoch 2-krát za týždeň, kde komunikovali svoj postup, prípadne prekážky alebo riziká ktorým čelia.

10.6 Postup implementácie

Postup opísaný v tejto kapitole odzrkadľuje procesy a postupy pri implementácií jednotlivých funkcionalít.

10.6.1 Príprava backlogu

Úplne prvým a najzákladnejším krokom bola príprava backlogu. Bez backlogu je náročné prioritizovať, plánovať alebo odhadovať množstvo potrebnej práce na dokončenie či už šprintu alebo danej aktualizácie.

V tomto prípade to ale nebolo také jednoduché, pretože dizajnéri nemali pripravené žiadne dokumentácie k práve vytvoreným požiadavkám takže producenti museli nájsť iný spôsob, akým merať progres tímu a ako plánovať. Nemohli vytvoriť backlog, scrum tímy a začať plánovať, pretože požiadavky neboli dostatočne detailné.

Prvým krokom bolo založiť nový backlog pre aktualizáciu a tvorba stories primárne pre tvorbu samostatných požiadaviek. V tomto štádiu teda stories pokrývali len potrebné dizajnové dokumenty, ktoré museli byť doručené ešte pred tým ako začala samotná práca.

Na začiatok to stačilo. Po dokončení tvorby týchto stories ich producenti za pomoci product ownera (kreatívneho riaditeľa) odprioritizovali.

10.6.2 Tvorba prvého tímu a plánovanie

Prvý tím používajúci niektoré z agilných procesov a praktík bol poskladaný len z dizajnérov. Neobsahoval teda kombináciu rôznych oddelení.

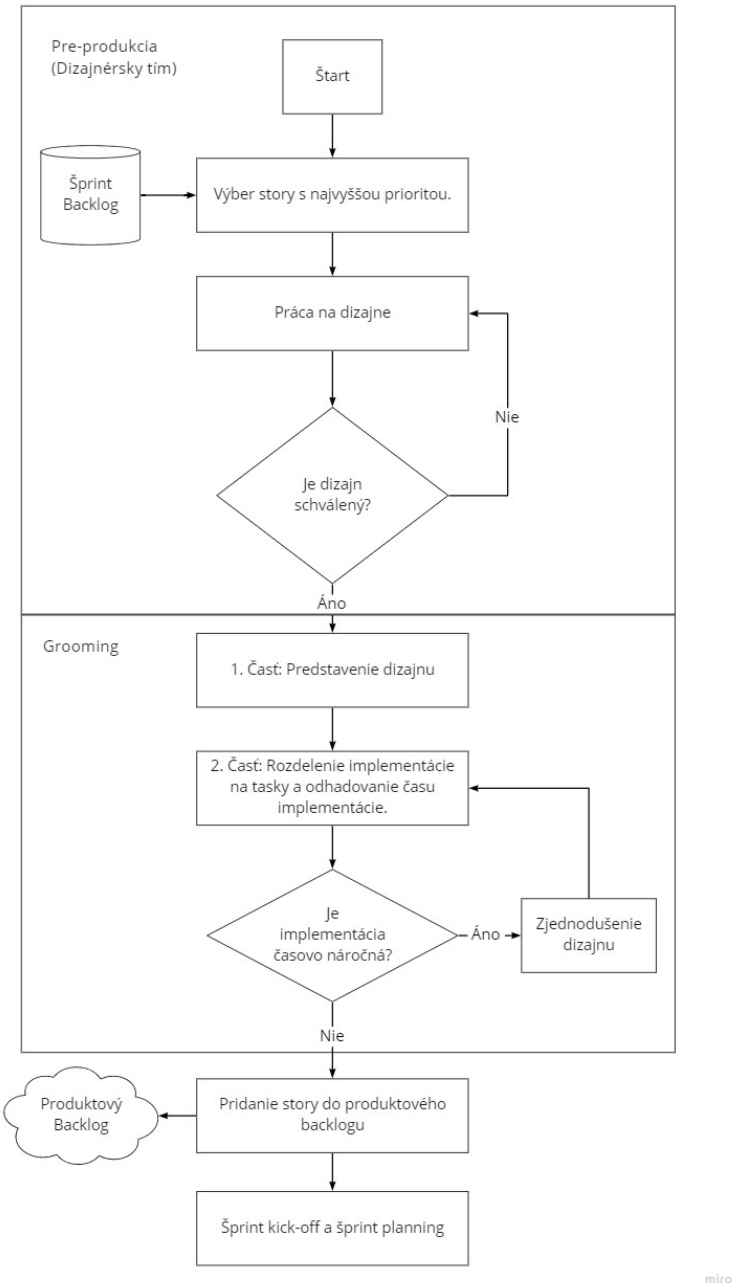
Títo dizajnéri fungovali na týždených iteráciách (šprintoch). Šprint začínal plánovaním na daný týždeň. Beh tohoto stretnutia bol veľmi podobný, ako plánovanie šprintu opísané v kapitole 8.4.2.

Kreatívny riaditeľ, ktorý predstavoval product ownera, opisoval jednotlivé stories tímu a za ich pomoci ich vkladali do týždeného šprintu.

Tím sa stretal každé ráno na ranných synchronizáciach a bavil sa o progrese.

Na konci týždňa sa celý tím stretol na review a product owner hodnotil dizajnové dokumenty a schvaľoval tie, ktoré mohli ísť do implementácie.

Product owner dostával tieto dokumenty vždy skôr, aby nepredlžoval stretnutie ich čítaním, ale len dávaním spätnej väzby/schvalovaním. Prišli aj časy, kedy bolo nových dokumentov toľko, že ich product owner nestíhal schvaľovať. Vtedy ho zastúpil líder dizajnérov. Keďže vývoj bol blokovaný dokumentáciou, väčšinou sa nečakalo do konca šprintu na dokumenty. Akonáhle bol dokument pripravený na schválenie, scrum masteri ho predali product ownerovi. Reviews boli využívané aj na dávanie spätnej väzby dizajnérom ohľadom celkovej vízie projektu.



Obr. 10.1 Tok tvorby produktového backlogu. Zdroj (vlastný).

Akonáhle bol niektorý z dokumentov dokončený, vytvoril sa tím a začalo sa s tvorbou taskov a časovými odhadmi. Počas tohoto stretnutia dizajnér predstavil kreatívny dokument, tím sa ho počas prezentácie mohol pýtať a keď už žiadne otázky neboli, prešlo sa na diskusiu ohľadom implementácie. Po tejto diskusii tím vytvoril tasky pre jednotlivé story a dopísal k nim časové odhady.

V prvých iteráciách sa táto diskusia odohrala na jednom stretnutí. No neskôr sa zistilo, že pri komplikovanejších funkcionalitách tím začal robiť väčšie chyby v odhade

implementácie, pretože nemal dostatočný čas na premýšľanie. Producenti spolu s technickými lídrami neskôr prišli s riešením, ktoré toto stretnutie rozdelilo na dve časti. V prvej časti dizajnéri odprezentujú svoj nápad a tím môže klásť otázky. V druhej časti sa urobil opäť priestor na otázky a pokračovalo sa s celým plánovaním. Medzi týmito stretnutiami mal tím čas detailne preštudovať dizajnový dokument, preštudovať si implementácie jednotlivých systémov, s ktorými daná funkcionalita bude spolupracovať a pripraviť si otázky. Dôležitou osobou na tomto stretnutí bol kreatívny riaditeľ v roli product ownera, ktorý v prípade časovo náročnej technickej implementácie pomohol dizajnérom hľadať zjednodušené verzie danej funkcionality tak, aby sa zredukoval čas implementácie bez straty hodnoty.

Dobrým príkladom je tím vyrábajúci charaktery. Ako je na začiatku kapitoly spomenuté, tento tím musel vytvoriť desať charakterov do necelých dvoch mesiacov, aj keď v minulosti trvalo vytvoriť jeden charakter približne dva mesiace. Vďaka iterovaniu a zjednodušovaniu dizajnového dokumentu priamo počas týchto stretnutí, sa podarili implementácie zjednodušiť na toľko, že bolo možné doručiť jeden charakter v priebehu šprintu. V niektorých prípadoch to boli aj dva charaktery.

Po vytvorení backlogu sa robil scrum kick-off, ktorý pozostával zo školenia členov tímu a plánovania šprintu. Samotné plánovanie šprintu prebiehalo veľmi podobne ako môžete vidieť na obrázku (obr. 8.2) z teoretickej časti. V neskorších fázach sa toto stretnutie používalo aj na diskusiu s tímom o tom, aké procesy budú používať od začiatku. V prípade, že členovia tímu si už prešli nejakými scrum tímami, mali možnosť sa vyjadriť a odprezentovať svoje skúsenosti.

10.7 Vyhodnotenie výsledkov a diskusia

Vývoj pokračoval až do polovice decembra, kedy sa tímu podarilo odoslať funkčnú a stabilnú verziu hry do obchodov. Aby sa mohla kvalitná verzia hry dostať včas k užívateľom, musel sa neustále upravovať plán na základe tímového postupu. Úpravou plánu v tomto prípade rozumieme ako neustále úpravy rozsahu backlogu a reprioritizovanie jednotlivých stories na základe náročnosti implementácie a hodnoty, ktorú prinášajú tieto funkcionality koncovým hráčom. V konečnom výsledku sa tímu nepodarilo doručiť "všetko na 100%", ale to je v tomto prípade úplne v poriadku. Cieľom bolo doručiť hodnotu hráčom, čo sa vďaka neustálym úpravám rozsahu a agilnému plánovaniu podarilo v rámci možností projektu.

Neustála komunikácia medzi oddeleniami v malých tímoch pomohla vývojárom rýchlejšie rozpoznať riziká a následne na ne reagovať dostatočne dopredu. K samotnej kvalite implementácie dopomohla aj beta verzia hry, ktorá so sebou priniesla určité komplikácie, keďže na to hra nebola pripravená. No pomohla tímu opraviť kritické chyby

pred samotným vydaním hry.

Aktualizácia a nové funkcionality, ktoré obsahovala, zlepšila aj niektoré kľúčové identifikátory na úspešnosť hry. Pre firmu predstavovali tieto dva mesiace začiatok ich novej kultúry postavenej na agilných metodikách a agilných samostatných tímoch. Slovo kultúra je veľmi dôležité, protože agilita nieje len o procesoch, ktorými sa doručujú projekty. Je to o nastavení mentality celej spoločnosti. Vedenie sa musí naučiť dôverovať svojim ľuďom, aby im boli schopní odovzdať zodpovednosti, ktoré sú v mnohých spoločnostiach nad rámec samostatného tímu. Rovnako tak tím musí mať dôveru vo vedenie, aby vedel, že ho v prípade potreby podporí.

Aj keď to v predchádzajúcich riadkoch môže vyzerať, že táto spoločnosť dosiahla agilnej kultúry, v skutočnosti bola len na začiatku. Tréning ľudí a zmena kultúry vyžadujú dlhší čas a úsilie.

10.7.1 Poučenia z vývoja

Manažment si uvedomil množstvo chýb, ktoré urobili počas zavádzania agilných metodík. Najhlavnejšie z týchto chýb si v krátkosti rozoberieme v tejto kapitole.

Kvalitnejšie školenie

Ako je písané na začiatku štúdie, školenie ľudí sa ukázalo ako nedostačujúce. Aj keď každý jeden člen tímu prešiel cez dve stretnutia zamerané na scrum, zodpovednosti tímu a jeho procesy, veľakrát sa stalo, že ľudia nerozumeli, prečo to robia. Rovnako sa tiež mnohokrát nesprávali na základe agilných metodík. Samozrejme, je to spôsobené aj tým, že takáto zmena u ľudí vyžaduje čas, no aj samotnou témou školenia. Manažment na základe tejto skúsenosti prerobil školenie o agilných metodikách, aby sa zameriavalo primárne na mentalitu a kultúru, ktorú sa snažia vytvoriť. Samotné procesy začali prezentovať ako cestu k tejto kultúre. Manažment taktiež rozšíril školenia na viacero stretnutí. Tieto stretnutia už ale neboli pasívnou prednáškou, ale primárne prezentovanie názorov jednotlivých ľudí, čo sú podľa nich agilné metodiky a ako pomáhajú tímu.

Nemeniť tímovú štruktúru

Počas opísanej aktualizácie sa členovia tímov menili na základe funkcionalít, ktoré boli implementované. No bol tam jeden tím, na výrobu nových charakterov, ktorý sa počas celej produkcie nezmenil. Ako je v kapitole uvedené, tento tím začal vykazovať kvalitnejšie výsledky a jeho členovia boli aj spokojnejší, pretože si lepšie zvykli na prácu medzi sebou. Tento princíp stálych tímov sa manažment snažil uplatniť na všetky tímy. Samozrejme, nie vždy je to možné a veľakrát sa stalo, že jeden človek bol súčasťou hneď

niekoľkých tímov. Dôležité ale je, aby sa tímy nemenili na bázi šprintov.

Nezamieňať role

Jednou z najzávažnejších chýb manažmentu bolo zamieňať rolu scrum mastera za product ownera na reviews v prípade, že product owner sa ceremónie nemohol zúčastniť. Kvôli tejto chybe bolo oveľa náročnejšie pre tím akceptovať scrum mastera ako svojho člena. Rovnako to spôsobilo veľa nedorozumenia o scrum masterovej roli pre tím, hlavne medzi vývojármi, ktorí začínajú s agilnými metodikami.

Budovanie tímov na základe špecializácie a personality

V prvých iteráciach počas vývoja sa jednotlivé scrum tímy tvorili z členov primárne na základe špecializácie a seniority. S odstupom času manažment zistil, že síce tento spôsob funguje, no nie je dostatočne efektívny. Motiváciou pre zamyslenie sa nad touto témou bolo, že niektoré tímy, aj keď pozostávali zo skúsených vývojárov neboli také efektívne v práci a v adaptácii agilných metodík v porovnaní s inými tímami. Po analýze zistili, že efektívnejšie tímy obsahujú jedného alebo viac "ťahačov", ktorí svojou aktivitou dokážu potiahnuť a motivovať celý tím k rešpektovaniu agilných metodík. Na základe tohoto zistenia, manažment prekombinoval niektoré tímy, aby obsahovali aspoň jedného "ťahača".

11 PRÉMIOVÁ PC HRA

V tejto kapitole rozoberáme vývoj produktu, ktorý je väčší ako hra v predchádzajúcej štúdii. Rovnako aj počet vývojárov v tíme pracujúci na tejto hre je o trošku väčší. Tím pozostáva primárne zo seniorov s mnohoročnými skúsenosťami a obsahuje aj viacej ľudí v manažmente. Väčšina členov z tímu sa poznala, pretože pracovali spolu v predchádzajúcom štúdiu, z ktorého odišli. Skúsenosť z tohoto štúdia podnietila vývojárov, aby sa pri budovaní novej spoločnosti vyvarovali chýb vedenia a manažmentu z minulosti. Aj tento fakt podnietil kultúru štúdia, ktorá bola od začiatku postavená na dôvere, aby sa pri vývoji uprednostňovali jednotlivci nad procesmi.

V čase písania tejto štúdie projekt ešte nebol ohlásený širokej verejnosti. Taktiež ako v predošlej prípadovej štúdii, vyberieme si časové obdobie počas ktorého budeme rozoberať implementované agilné metodiky a reakcie tímu na tieto zmeny.

11.1 Hra

Spoločnosť, vyvíjajúca hru bola stará iba jeden rok v čase písania tejto prípadovej štúdie. Nie je to ale jeho prvá hra. Tesne po založení pracovali členovia tímu ešte na jednej hre, ktorá bola štúdiu priradená investorom na dokončenie. Aj keď sa členovia tímu poznali pred založením štúdia, bola práca v novom štúdiu a na novej hre plná nových skúseností a výziev. Tieto skúsenosti pramenili primárne z rozdielnych požiadaviek na vývojové procesy od manažmentu. Medzi tieto požiadavky patrila častejšia komunikácia medzi oddeleniami rovnako ako detailnejšia a prepracovanejšia dokumentácia. Keďže sa jednalo o novú spoločnosť, bol oveľa väčší tlak aj na dôležité projektové miľníky. Pre mladé štúdio je veľmi dôležité ukázať investorovi, že dokáže doručiť hotový produkt. Medzi výzvy patrilo aj to, že toto bola prvá hra svojho typu a žánru na ktorej tím pracoval.

Hra, ktorú rozoberáme bola priamo navrhnutá v štúdiu. Nakoľko išlo o prvú hru navrhnutú v štúdiu, bolo o to dôležitejšie, aby bola doručená na čas a v stanovenej kvalite.

Vzhľadom na to, že plán projektu vznikol ešte pred tým ako bola objasnená vízia, od začiatku práce na hre tím pracoval s veľmi tesnými termínami. V skorej pre-produkcii aktívne pracovali primárne dizajnéri, pretože sa vývoj prekrýval s prácou na predošlej hre. Aj keď väčšina tímu dokončovala starý projekt, termíny nového sa neposúvali a neprispôsobovali.

Prvým dôležitým miľníkom pre hru bol koncept build, ktorý mal slúžiť ako vstupenka do ďalšieho vývoja. Na konci miľníka sa odprezentovala pripravená verzia hry investorovi, ktorý ju následne schválil. Hoci koncept build obsahoval mnoho dôleži-

tých funkcionalít, bola to stále len konceptová verzia, ktorej primárnym účelom bolo dokázať, že má zmysel pracovať na hre tohoto rázu. Väčšina týchto funkcionalít predstavovala technický dlh, s ktorým sa tím musel okrem práce na nových funkcionalitách počas vývoja vysporiadať.

Okrem samotného overenia nápadu hry, koncept build poslúžil tímu aj ako miľník, na ktorom si vývojári začali uvedomovať, že dokončiť hru v očakávanom rozsahu a v stanovenom čase bude veľmi náročné. Čím viac kreatívnej dokumentácie bolo hotovej, tým viac si tím uvedomoval, že bude veľmi dôležité pracovať s rozsahom jednotlivých funkcionalít, aby sa podarilo doručiť hru, ktorá sa bude na trhu predávať. Ďalší dôležitý miľník predstavoval vertical slice, kde mala hra obsahovať minimálne hrateľný základ všetkých najdôležitejších funkcionalít, taktiež aj finálnu kvalitu vizuálu v hre. Počas práce na tomto miľníku na základe postupu sa podarilo vývojárom definovať približné oneskorenie spôsobené rozsahom hry. Oneskorenie predstavovalo približne dva mesiace pre miľník vertical slice. Toto oneskorenie sa ale, bohužiaľ, nepodarilo tímu presadiť u investora, ktorý dovolil meškanie v rámci jedného mesiaca za podmienky, že sa termín ďalších miľníkov neposunie. To znamená, že hra musí byť vydaná v rovnakom dátume. O to väčší tlak sa vyvíjal na iteratívnu implementáciu jednotlivých funkcionalít, neustále plánovanie a redukciu rozsahu celej hry.

11.2 Vertical Slice

Prvým krokom po dokončení konceptového buildu bolo definovať backlog pre ďalší miľník a teda aj pre celú hru. Do tohoto procesu sa okrem vedenia projektu zapojil aj celý tím, ktorý vkladal svoje nápady, čo bude pravdepodobne treba vytvoriť pre úspešné dokončenie hry. Backlog, okrem samotných herných funkcionalít, obsahoval aj nástroje dôležité pre jednotlivé oddelenia, potrebné na tvorbu hry. Po nahromadení dostatočného množstva informácií a funcionalít prišiel čas na prioritizáciu. Počas vývoja sa samozrejme objavili ďalšie funkcionality, s ktorými sa na začiatku nepočítalo a ukázali sa ako dôležité na implementáciu. Tím s tým počítal a akonáhle nastala podobná situácia, zvolalo sa stretnutie s vedením projektu a prebrala sa redukcia rozsahu hry do miery kedy bolo množstvo funkcionalít znovu stihnuteľné.

Prioritizácia sa taktiež odohrávala primárne na úrovni vedenia projektu, no v prípade prioritizácie funkcionalít pre celú hru, priložil ruku aj samotný tím. Ak si niekto žiadal novú funkcionalitu alebo nástroj, sám mu priradil prioritu na základe vlastného vyhodnotenia. Niektoré požiadavky mohli byť na úkor iných preprioritizované na nižšiu prioritu.

Funkcionality sa prioritizovali primárne pre vertical slice. Prioritizovalo sa pomocou metódy MoSCoW z kapitoly 6.3 na základe štyroch úrovní priority.

- Must-have: Funkcionality označené touto prioritou museli byť doručené. Ak sa náhdou nejaká z týchto funkcionalít nedoručí, predpokladá sa, že hra bude vzbudzovať pocit, že nie je dokončená a odrazí sa to na jej úspešnosti.
- Should-have: Odporúča sa, aby funkcionality s touto prioritou boli doručené. Ich absencia ovplyvní úspešnosť projektu vo veľkej miere.
- Could-have: Tieto funkcionality dokážu zvýšiť úspešnosť hry na trhu, no ich absencia ju nezhorší.
- Won't-have: Tieto funkcionality nebudú obsiahnuté v danom miľníku.

Po dokončení prioritizácie sa ukázalo, že približne 80% všetkých funkcionalít v backlogu pre vertical slice, je označených prioritou must-have alebo should-have. To znamenalo, že väčšina rozsahu hry musí byť doručená práve v tomto miľníku.

Na základe skúseností z predchádzajúceho miľníka, tímu, znalostí o novej hre, si môžeme spísať riziká a cieľe späté s doručením vertical slice.

11.3 Riziká

- Tesné termíny bez možnosti posunutia. Ako sme si už povedali, tímu sa podarilo dohodnúť posunutie termínov o jeden mesiac pre vertical slice, no termín pre nasledujúci miľník alpha zostáva stále rovnaký.
- Veľký rozsah hry, ktorý stále nemusí byť finálny. Počas vývoja sa predpokladá, že sa ukážu funkcionality, s ktorými sa na začiatku nepočítalo.
- Technický dlh spôsobený rýchlym prototypovaním s ktorým sa bude musieť tím vysporiadať počas práce na vertical slice.
- Nové štúdio bez zabehnutých procesov. Aj keď sa väčšina tímu poznala z predchádzajúcej spoločnosti, bolo veľmi dôležité stanoviť nové procesy, ktoré odpovedali spôsobu vedeniu štúdia a jeho behu.
- Málo dokumentácie, či už k samotnej hre tak aj k štúdiu.

11.4 Ciele

- Implementácia nových procesov, ktoré budú vhodné pre chod štúdia a situácie v akej sa tím nachádza. Zároveň je ale dôležité, aby sa tieto procesy nepovyšovali nad potreby jednotlivcov.

- Minimum nadčasov. Rozsah hry je na prvý pohľad priveľký, no s použitím správnej prioritizácie a kontinuálneho plánovania chce manažment redukovať potrebu členov tímu pracovať nadčasy.
- Vytrénovať tím k agilite a samostatnosti.
- Zlepšít komunikáciu medzi oddeleniami a zvýšiť efektivitu tímu.
- Zaviesť iteratívnu implementáciu jednotlivých funkcionalít, ktorá dopomôže k skorej evaluácii a prípadnému reprioritizovaniu.
- Vytvoriť takzvané feature tímy, ktoré budú spolupracovať na dokončení špecifických funkcionalít.
- Zlepsiť kvalitu kreatívnej dokumentácie rovnako ako technickej dokumentácie.

11.5 Zmena procesov a návrh nových pipelines

Základným krokom bolo zaviesť nové procesy a postupy, ktoré majú podporiť tím pri úspešnom doručení hry.

11.5.1 Upravená verzia agilných metodík

Jedným z cieľov bolo zaviesť procesy a zároveň rešpektovať potreby jednotlivcov. To v prípade procesov tohoto štúdia znamená nezavádzať automaticky procesy podľa nejakej špecifickej agilnej metodiky (napríklad scrum), ale vybrať/vytvoriť vhodné procesy, ktoré poslúžia na doručenie špecifickej funkcionality.

Samozrejme, existovali spoločné procesy, ktoré sa používali pri každej funkcionalite, ako napríklad, procesy pre schvaľovanie dokumentov alebo plánovanie implementácie. No samotné procesy jednotlivých tímov záležali na ich potrebách.

Počas prvých mesiacov vertical slicu tímy pracovali v mesačných šprintoch. Tieto dlhé šprinty boli zvolené kvôli mesačným stretnutiam s investorom. Celý vývojársky tím pracoval v týchto mesačných iteráciách. To znamená, že aj samotné feature tímy zdieľali šprint s ostatnými členmi na projekte. Neskôr, keď sa tím blížil k miľníku, šprint sa skrátil na dva týždne. To znamenalo, že každé dva týždne sa musel doručiť stabilný build, ktorý reprezenzoval postup tímu za toto časové obdobie.

11.5.2 Tvorba feature tímov

Feature tím je v tomto kontexte veľmi podobný scrum tímu. V konečnom výsledku funguje na veľmi podobných pravidlách. Už názov napovedá, že feature tímy boli poskladané pre jednotlivé funkcionality. Teda ak bol nejaký vývojár členom feature tímu,

znamenalo to, že má zodpovednosť implementácie svojej časti na základe, ktorej bol feature tím zostavený.

Role

Každý feature tím pozostával z rôznych oddelení potrebných na doručenie špecifickej funkcionality. Podľa potrieb funkcionality sa teda vyberali ľudia so skúsenosťami a schopnosťami na doručenie. Jednotlivé role v takomto tíme dokážeme rozdeliť následovne.

- Feature owner: Producent zodpovedný za doručenie funkcionality. Medzi jeho primárne zodpovednosti patrí komunikácia s tímom, pomáhanie tímu v prípade potreby, analýza rizík, organizovanie nevyhnutných stretnutí, komunikácia s nadriadenými.
- Product owner: Táto rola je prebraná zo scrumu. Ako sme si v teoretickej časti povedali, product owner zastupuje biznisovú stranu. V prípade feature tímov to predstavovalo prioritizáciu jednotlivých funkcionalít a robenie všetkých dôležitých rozhodnutí ohľadom produktu. Celý projekt mal jedného product ownera, ktorý bol spoločný pre všetky tímy.
- Člen tímu: Zástupca oddelenia zodpovedný za implementáciu svojej časti.
- Buddy: Táto rola sa používala primárne v kontexte technických oddelení. Buddy bol vývojár so znalosťou určitého systému alebo škály systémov, ktoré boli v hre naimplementované a práve implementovaná funkcionalita s tými systémami pracuje. Primárnou zodpovednosťou tejto role bolo podporovať členov tímu na plánovacích stretnutiach, byť oboznámený s kreatívnou a aj technickou dokumentáciou pre tieto funkcionality.

Procesy

Okrem procesov, ktoré opisujeme v nasledujúcich častiach tejto štúdie, tímy nemali žiadne špecifické stretnutia prebrané z nejakej agilnej metodiky. Samozrejme, ak si ľudia v tíme tento štýl stretnutia vyžiadali, tak sa organizoval. Tým pádom každý tím fungoval úplne iným spôsobom.

Toto rozhodnutie vzniklo primárne z toho, že špecializácie ľudí v tíme neboli veľmi distribuované, teda bolo úplne bežné, že jeden vývojár bol súčasťou hneď niekoľkých tímov naraz. Preto by bolo pre neho časovo vyťažujúce navštevovať všetky potrebné ceremónie. Napríklad, ak tím potreboval niečo zlepšiť, nečakal do retrospektívy, ale priamo prediskutoval riešenie s ostatnými členmi.

Na druhú stranu tieto rozhodnutia spravili všetky procesy veľmi agilné, odzrkadlujúce potreby jednotlivých ľudí.

11.5.3 Plánovanie šprintu

Plánovanie šprintu sa skladalo z niekoľkých čiastočných krokov.

V prvom kroku exekutívny producent vytvoril ciele pre všetky šprinty na základe výtupu a pokroku z predchádzajúcich šprintov. Tieto ciele neobsahovali drobné detaily, no boli dostatočné, aby sa následne jednotlivý producenti o nich dokázali rozprávať s členmi tímu.

V ďalšom kroku sa producenti stretali s relevantnými tímami alebo ľuďmi a diskutovali jednotlivé ciele a priority pre najbližší šprint. Nie vždy sa plánovanie odohrávalo s celými feature tímami. Keďže všetci ľudia boli distribuovaní medzi niekoľko funkcionalít, nie vždy bolo pre nich relevantné plánovať implementácie s každým tímom. Napríklad, ak mala funkcionalita nejakú systémovú časť, ktorá spracovávala dáta a uživateľskú časť, ktorá dané dáta zobrazovala uživateľom, tak sa tieto časti mohli implementovať nezávistle na sebe v závislosti od dostupnosti vývojárov. Ak náhodou systémový programátor bol priradený na inú funkcionalitu, no programátor uživateľského rozhrania bol voľný, tak sa implementovalo "mŕtve"UI, ktoré nezobrazovalo žiadne dáta a malo len zopár základných funkcionalít. Vďaka tomu už neboli blokovaní UI artisti, ktorí mohli začať vkladať svoj hotový 2D art. Akonáhle sa uvoľnil systémový programátor, tak dorobil svoju časť na funkcionalite a mohla sa poslať na testovanie.

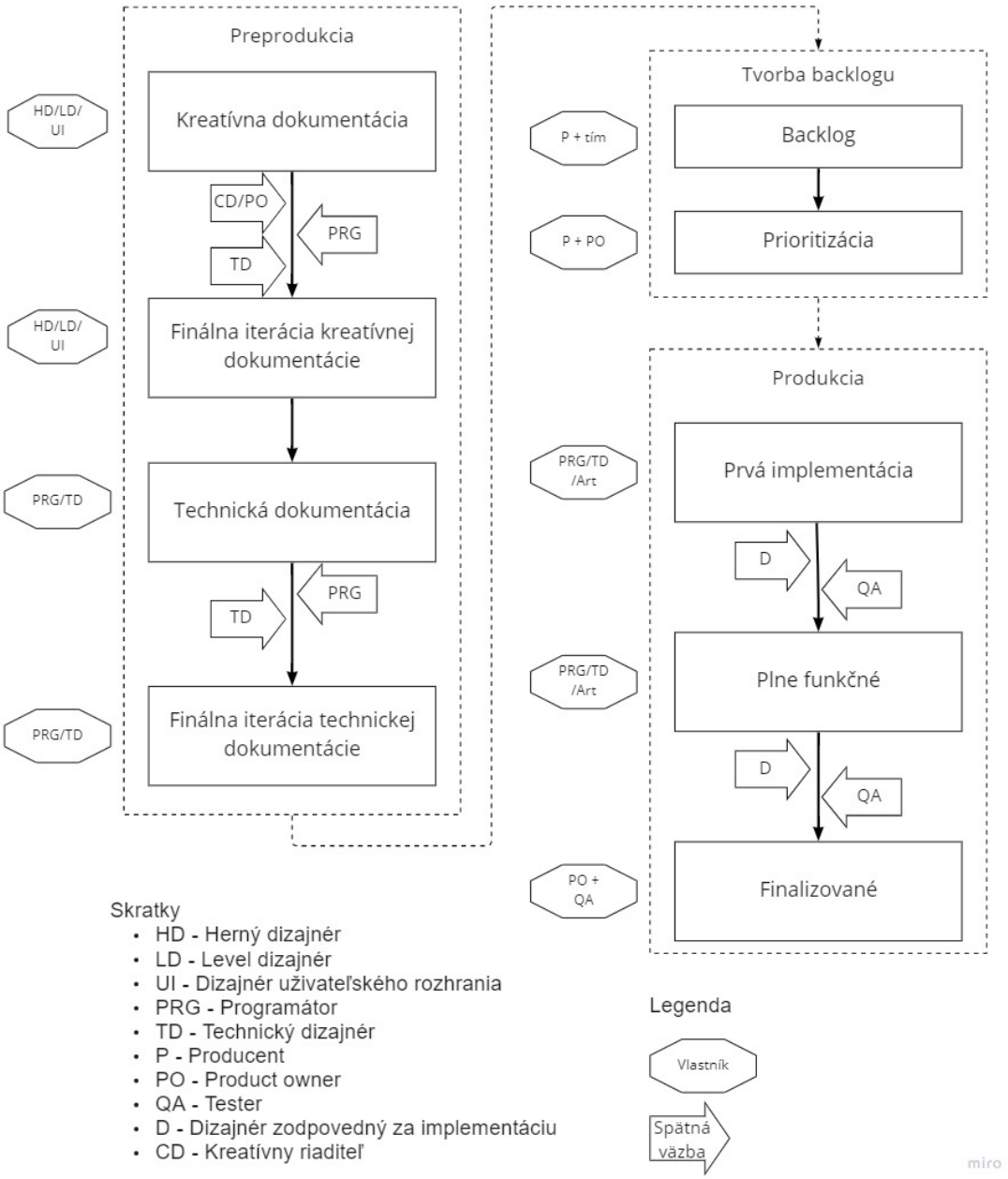
Informácie ohľadom dostupnosti jednotlivých vývojárov boli poskytnuté exekutívnym producentom a zohľadnené v pláne. Takže jeden z cieľom mohol vyzerať napríklad: implementujte nefunkčné uživateľské rozhranie pre manažment postavy.

Následné plánovanie s jednotlivcami alebo tímami sa podobalo plánovaniu z obrázku (obr. 8.2) z teoretickej časti. Predpokladom takéhoto plánovania je samozrejme pripravený backlog. Ako sa k nemu tím dostal si rozoberieme v ďalšej časti.

11.5.4 Šprint

Ako sme si už povedali, samotné procesy pre jednotlivé tímy boli veľmi špecifické. Čiže šprinty neobsahovali žiadne začiatočné ani ukončovacie ceremónie pre tím. Koniec šprintu predstavovalo stretnutie s investorom a krátka prezentácia postupu od exekutívneho producenta.

11.6 Postup implementácie



Obr. 11.1 Postup implementácie funkcionality. Zdroj (vlastný).

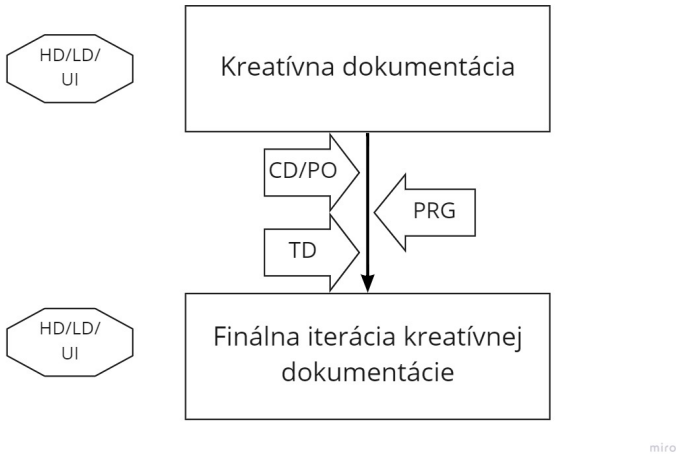
Tento postup na tvorbu a preproduckiu jednotlivých funkcionalít sa používa vo všetkých feature tímoch. Sú to najzákladnejšie postupy pre tvorbu dokumentácie a následne implementáciu, ktoré zaručujú, že všetci členovia tímu vedia, čo majú robiť v jednotlivých krokoch. V nasledujúcich riadkoch si tieto postupy opisujeme detailnejšie.

11.6.1 Preprodukcia

Cieľom preprodukcie je vytvoriť základnú dokumentáciu pre danú funkcionalitu, ktorá je dostatočná na prvú implementáciu. Po naimplementovaní prvej verzie sa táto dokumentácia môže značne zmeniť.

Tvorba kreatívnej dokumentácie

Kreatívny dokument môže byť vytvorený viacerými oddeleniami. Napríklad herným dizajnérom, ktorý opisuje novú funkcionalitu alebo systém, ktorý sa bude implementovať. Alebo level dizajnérom, ktorý potrebuje novú funkcionalitu od programátorov, aby mohol urobiť rôzne levely v hre zábavnejšie. Taktiež dizajnér uživateľského rozhrania potrebuje dokumentovať svoje návrhy, aby mohli byť následne jednoducho realizované programátormi.



Obr. 11.2 Tok tvorby a schvaľovania kreatívnej dokumentácie. Zdroj (vlastný).

Akonáhle je dokumentácia dokončená zo strany autora, príde čas na schvaľovanie. Pred tým ako sa tento dokument posunie implementačnému feature tímu, musí ho najskôr schváliť kreatívny riaditeľ a product owner. Kreatívny riaditeľ schvaľuje primárne víziu danej funkcionality a môže mať poznámky k samotnému návrhu. Product owner naceňuje tento návrh prioritami na základe jeho znalostí o potrebách zákazníka. Tento krok na začiatku tvorby dokumentácie zaručuje, že tím nebude strácať čas na dokumentácii funkcionality, ktorá sa bude celá prerábať, alebo má nízku prioritu.

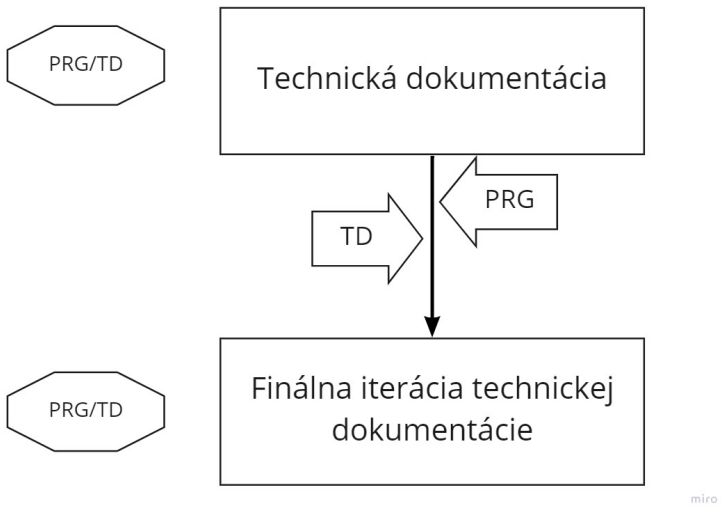
Po kontrole od kreatívneho riaditeľa a product ownera sa dokument presúva k relevantným technickým ľuďom. V prípade štúdia to bol programátor a technický dizajnér. Aj keby celú funkcionalitu implementoval len jeden z nich, stále sa dokumentácia posiela viacerým ľuďom, aby vedeli o nových funkcionalítách, ktoré sa budú implementovať v

blízkej dobe.

Technickí ľudia kontrolujú zrozumiteľnosť dokumentácie pre implementáciu. V dokumente môžu zanechávať otázky, na základe ktorých autor urobí ďalšiu iteráciu.

Tvorba technickej dokumentácie

Technický dokument je tvorený za účelom výskumu implementácie, jej počiatočným návrhom pred samotnou implementáciou, alebo dokumentáciou naimplementovanej funkcionality. Ideálne by sa mal technický dokument vytvárať pred implementáciou do určitej miery, ktorá je programátorovi dostatočne známa, aby mohol začať implementovať funkcionalitu. Tento dokument môže byť samozrejme doplňovaný o nové zistenia počas práce na funkcionalite. Dôležité ale je, že takmer pri každej zmene sa posiela tento dokument na prečítanie relevatným technickým ľuďom. V kontexte feature tímu to sú buď ďalší programátori, ktorí implementujú funkcionalitu v danom tíme, alebo buddy, ktorý slúži ako podpora.



Obr. 11.3 Postup tvorby a schvaľovania technickej dokumentácie. Zdroj (vlastný).

11.6.2 Backlog

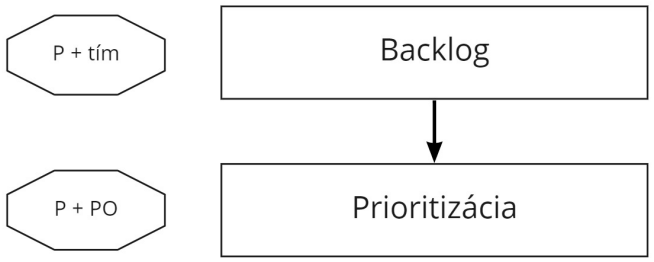
Akonáhle je dokumentácia dokončená a tím pozná detaily novej funkcionality je čas vytvoriť stories v backlogu.

Tieto stories väčšinou vytvárajú producenti sami na základe priorít a potreby implementácie. Pri každej story je veľmi dôležité, aby mala priradenú správnu prioritu a jasne špecifikovanú časť dokumentu na ktorú sa odkazuje.

Producenti nevkladajú informácie z dizajnového dokumentu priamo do story, pretože

dizajnové dokumenty sa zvyknú meniť počas rôznych iterácií a kopírovanie informácií môže len spôsobiť ďalšiu úroveň, ktorú bude treba kontrolovať. Okrem samotného dokumentu producenti pridávaju aj výstižnú informáciu, čím sa story zaoberá. A to z dôvodu, aby aj keď sa k takémuto tasku dostane vývojár po niekoľkých týždňoch, stále bude jasné, čo v rámci úlohy treba urobiť.

Akonáhle producenti vytvoria relevantné stories, dohodnú si stretnutie s feature tímom a pridajú časové odhady pre tieto funkcionality. Ak sú už známe priority pre jednotlivé šprinty, pomocou tímu tieto stories rovno zaplánujú do relevantných šprintov.



Obr. 11.4 Príprava backlogu. Zdroj (vlastný).

11.6.3 Produkcia

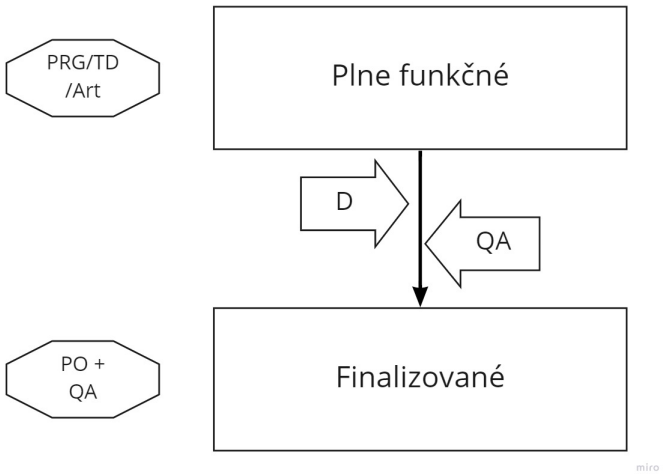
Produkcia funkcionality môže byť rozdelená do viacerých častí. Rovnako býva rozdelená aj do iterácií.

Iteráciou sa v tomto prípade rozumie implementácia, ktorá sa môže skladať aj z niekoľkých častí no na konci odpovedá hodnote minimálnej verzie funkcionality, ktorá môže byť zevaluovaná. Väčšinou prvá implementácia funkcionality obsahuje časti s prioritou must-have, no môže obsahovať aj iné priority.

Po prvej implementácií si funkcionalitu primárne kontrolujú dizajnéri, ktorí v prípade potreby do nej vložia dáta a robia jej prvú evaluáciu. Aby sa ale mohla prvá implementácia označiť za dokončenú, posiela sa na testovanie. Na implementácii sa následne iteruje pokiaľ hodnota, ktorú prináša nie je dostačujúca pre hru. Funkcionalita je označená za finálnu product ownerom a testermi.

11.7 Vyhodnotenie výsledkov a diskusia

V čase písania tejto prípadovej štúdie vývoj na vertical slice miľníku neustále pokračuje. Tím sa snaží redukovať rozsah a oneskorenie za pomoci iteratívneho vývoja. Každá



Obr. 11.5 Produkcia. Zdroj (vlastný).

funkcionalita prechádza dlhým schvaľovaním a prioritizovaním, aby sa zaručilo, že sa naimplementujú tie najdôležitejšie časti ako prvé.

K iteratívnemu prístupu, redukovaniu ceny a rozsahu jednotlivých funkcionalít dopomáha práve dokumentácia. Zároveň je to jedna zo zmien, s ktorou sa tím ťažšie stotožňoval. Udržať zdokumentované každé rozhodnutie a zmenu nie je vôbec jednoduché. Taktiež aktualizovať všetky dokumenty na základe spätnej väzby. Práve z toho dôvodu museli aj samotní producenti stráviť mnoho času čítaním a úpravou rôznych dokumentov. Pri vysokom tlaku a návale požiadaviek na preprodukciu nových funkcionalít sa veľmi jednoducho zabúda na aktualizáciu dokumentácie funkcionalít, ktoré sú momentálne implementované. Preto sa aj producenti rýchlo dostali do stavu, kedy museli čítať jednotlivé dokumenty hneď po dopísaní prvej iterácie a následne ich kontrolovať počas jednotlivých schvaľovaní, či obsahujú všetky dôležité informácie. Udržovanie aktualizovanej dokumentácie nemá len dlhodobé výhody ako udržanie aktuálnych informácií o funkcionalitách, ktoré sú naimplementované, aby sa nezabudlo ako majú fungovať. Ale taktiež aj krátkodobé, ako napríklad, dôvera tímu počas implementácie v tieto dokumenty. V určitej dobe sa občas stávalo, že ak dokumentácia nebola aktualizovaná, tak programátor namiesto komunikácie z dizajnérmi alebo produkciou jednoducho naimplementoval danú funkcionalitu podľa seba. Čo samozrejme spôsobilo zbytočnú prácu naviac a oneskorenie s implementáciou.

Veľmi zaujímavé boli situácie, v začiatkoch zavádzania rozsiahlych dokumentácií, počas ktorých sa niektorí programátori rozhodli implementovať funkcionality bez čítania dokumentácie. Táto implementácia pozostávala len na základe nejakej krátkej diskusie s dizajnérom. Toto je len jeden z mnohých príkladov od producentov, ktorý ukazuje, že už len zavedenie fundamentálnych procesov ako písanie kvalitnej a aktualizovanej dokumentácie nie je jednoduché ako pre manažment tak ani pre tím.

Ďalšou výzvou pre producentov a aj pre samotný tím bola práca s backlogom. Na začiatku projektu bolo veľmi časovo náročné vytvoriť stories pre všetky funkcionality, ktoré boli v danej chvíli známe. Okrem tvorby backlogu, museli producenti prísť aj so spôsobom a postupom, akým budú ostatní členovia tímu pracovať s backlogom. Následne prišla časť naučiť tím pracovať s nástrojom, ktorý slúžil na prácu s backlogom a predstaviť mu nové postupy. Rovnako ako v predchádzajúcich prípadoch to nebola jednoduchá úloha. Mnohé oddelenia až na pár jednotlivcov tieto procesy nerešpektujú. Bežnou praxou produkcie je kontrolovanie commitov od jednotlivých členov tímu a na základe nich aktualizovať stories v backlogu.

Taktiež zavádzanie iteratívnej alebo čiastočnej implementácie podľa priorít nebolo jednoduché. Zo začiatku bolo pre technických ľudí náročné implementovať len určitú časť funkcionality. Takouto časťou mohlo byť napríklad UI bez zapojenej funkcionality. Podobne aj v prípade iteratívnej implementácie, kedy sa testoval základný koncept funkcionality. Je pochopiteľné, že ak niekto vytvára novú funkcionalitu, chce, aby vyzerala čo najlepšie a dokázala čo najviac vecí pred tým, ako ju vyskúša niekto iný. Preto bolo potrebné naučiť tím aj tento minimalistický prístup, počas ktorého implementujú len jasne definovanú story.

V čase písania tejto práce jednotlivé procesy stále nefugovali v tíme dokonale. Zavádzanie takýchto zmien vyžaduje čas a veľa trpezlivosti, či už zo strany tímu tak aj produkcie. Tréning ľudí je dlhodobá záležitosť, ktorá si vyžaduje podporu zo strany manažmentu a ich pochopenie.

12 ODPORÚČANIA

V predchádzajúcich dvoch štúdiach sú opísané spôsoby a postupy, akými môžu byť vedené herné projekty. Obidva tímy, na ktorých sa vykonávali štúdie, boli rozdielne či už z pohľadu skúseností tak aj zvykov. V prípade tímu pracujúcom na mobilnej hre si manažment zvolil ráznejšiu cestu zavedenia agilných metodík, presnejšie scrumu. Počas jedného vydania projektu implementovali scrum do celého tímu a začali používať všetky jeho procesy a postupy. Na základe zistení z tejto iterácie následne tvorili dokumentáciu a znalosti, ktoré aplikovali v ďalších iteráciach. V druhom tíme zase manažment začal budovať agilné zvyky tímu pomalšie pomocou mentorovania ohľadom najzásadnejších postupov a procesov. Medzi ktoré, napríklad, patrilo písanie dokumentácie, alebo práca v manažérskymi nástrojmi.

Rovnako aj spomenuté hry predstavovali rozdielne výzvy pre manažment spoločností. Mobilná hra, ktorá v opisovanom čase už bola na trhu niekoľko mesiacov, vyžadovala prístup rýchlych iterácií, na konci ktorých bolo vydanie novej verzie priamo koncovému zákazníkovi. Jednotlivé iterácie pracovali s aktuálnymi dátami z trhu, na základe ktorých následne tím prioritizoval backlog. Taktiež bolo veľmi dôležité, aby sa hra udržiavala v každej iterácií alebo šprinte v čo najvyššej možnej kvalite.

V druhom prípade sa pracovalo na hre, ktorá bola v začiatkoch vývoja. Tento tím pracoval v dlhších iteráciách a prioritizácia backlogu sa diala primárne podľa prieskumu trhu.

Na základe štúdií z týchto dvoch projektov a tímov môžeme vidieť, že by sme len veľmi ťažko hľadali procesy, ktoré by vyhovovali každej spoločnosti. Každý tím je iný, má iné zvyky, skúsenosti a potreby. Rovnako procesy môžu závisieť aj od projektu. Ako je v teoretickej časti napísané, agilné metodiky poskytujú súbor nástrojov, ktoré pomáhajú riešiť vývojárom rôzne problémy a výzvy počas vývoja. Niektorým tímom vyhovuje, keď používajú všetky nástroje, iným nie.

Keďže neexistujú ideálne procesy, ktoré by riešili všetky problémy moderných tímov vyvýjajúcich hry, v nasledujúcich riadkoch si na základe predchádzajúcich prípadových štúdií spíšeme zopár doporučení, ktorých aplikáciou dokážu jednotlivé tímy zvýšiť efektivitu.

12.1 Kvalitná dokumentácia

V teoretickej časti uvádzame dva druhy dokumentácie ako súčasť kapitoly 4 a to technickej dokumentácie a dizajnového dokumentu. Tieto dva druhy dokumentácie nám pokrývajú požiadavky na funkcionality a technické aspekty jednotlivých implementácií. Keďže aj tvorba hier je primárne vývoj softvéru, odporúča sa aj tvorba doku-

mentácie architektúry, ktorá sa zameriava primárne na systémové komponenty, alebo dokumentácia uživateľských návodov na systémy [40].

Dokumentácia je mnohokrát videná ako jedna z najfrustrujúcejších častí herného vývoja, to ale neuberá na jej dôležitosti. Vďaka dokumentácii môžeme uchovávať dôležité rozhodnutia na rôznych úrovniach počas projektu. Základné typy dokumentácie môžu byť nasledujúce:

- dokumentácia požiadaviek v podobe dizajnového dokumentu,
- dokumentácia technických aspektov funkcionalít v podobe technického dokumentu,
- dokumentácia architektúry,
- dokumentácia uživateľských návodov na systémy,
- dokumentácia všetkých stretnutí a rozhodnutí na stretnutiach.

Každé štúdio alebo spoločnosť môže tvoriť vlastné druhy dokumentácie podľa potrieb. Dôležitým odporučením v tomto prípade je dokumentovať všetky dôležité rozhodnutia, na ktoré sa bude môcť tím odkázať kedykoľvek počas vývoja.

Na otázku, prečo by sme mali chcieť ukladať tieto rozhodnutia je mnoho odpovedí no najednoduchšia z nich je, že ľudia zabúdajú. V čase, keď tím diskutuje určitý problém, tak sa všetko zdá, že je nadmieru jasné a nie je potreba o tom písať dokument. Horší prípad nastáva, keď sa k prediskutovaným rozhodnutiam treba vrátiť o pár mesiacov neskôr.

Každý druh dokumentácie má svoj primárny účel, na základe ktorého by mala byť písaná. Napríklad jedným z účelov technickej dokumentácie môže byť, aby programátor premýšľal nad návrhom implementácie a v prípade potreby si na ňu urobil výzkum. Taktiež účelom tejto dokumentácie je distribuovanie znalostí medzi jednotlivých členov tímu a tým znížiť riziko straty kľúčových vedomostí v prípade, že zamestnanec opustí spoločnosť alebo len odíde na dovolenku.

Ako máme uvedené vo vyhodnotení prípadovej štúdie v kapitole 11.7, počas aktívneho vývoja môže byť veľmi náročné udržiavať projektovú dokumentáciu aktuálnu. Hlavne, keď sa jedná o herný vývoj a funkcionality sa neustále menia na rôznych úrovniach počas testovania. Na vyriešenie tohoto problému majú väčšinou štúdiá vlastné procesy, jeden z postupov sme si ukázali v druhej štúdii prémiového projektu.

12.2 Iteratívny vývoj

Vývoj v iteráciách pri správnom aplikovaní dokáže priniesť mnoho výhod do vývoja herného projektu. Ako je uvedené v kapitole 6.2, jedným z primárnych cieľov herného vývoja je nájsť zábavu pre určitú skupinu cieľových zákazníkov. Jednotlivé návrhy dizajnových dokumentov môžu vyzerať zaujímavo. Po implementácii a otestovaní s ostatnými hernými mechanikami a funkcionalitami môžeme rýchlo zistiť, že máme problém a že hra nie je zábavná. Práve vďaka iteratívnemu vývoju dokážeme hľadať zábavu po častiach a na základe výstupov jednotlivých iterácií robiť ďalšie rozhodnutia o smerovaní produktu.

Napríklad po implementácii možnosti skočiť v hre si môžeme uvedomiť, že je to zábavné a na základe toho reprioritizovať tvorbu jednotlivých levelov a preplánovať výrobu, aby podporovali tento typ zábavy. Samozrejme, takéto razantné rozhodnutia je lepšie robiť na začiatku vývoja, no aj počas plnej produkcie nám iterácie dokážu pomôcť robiť rozhodnutia o smerovaní projektu.

V prípadových štúdiách či už mobilného projektu v kapitole 10, alebo počítačového projektu v kapitole 11, môžeme vidieť, že akákoľvek časť vývoja môže byť robená v iteráciách. Funkcionality môžu byť implementované v iteráciách. Takýto spôsob implementácie pomáha tímu uvedomiť si hodnotu jednotlivých častí funkcionality, alebo ju prípadne zahodiť ešte pred tým, ako na nej programátori strávia veľa času. Taktiež na základe štúdií môžeme vidieť, že rôzne tímy majú rozdielne postupy a procesy, ktoré podporujú iterácie. Tieto procesy musia odzrkadlovať potreby tímu a projektu.

Pri zavádzaní iteratívneho vývoja sa odporúča, aby jednotlivé iterácie boli čo najkratšie [2]. Samozrejme, závisí to znovu na potrebách a skúsenostiach tímu, no krátke iterácie so sebou prinášajú mnoho výhod. Jednou z nich je rýchle učenie sa tímu [2]. Keď tím začína s iteráciami musí sa mnohokrát naučiť veľa lekcií. Či už sa jedná o samotný odhad práce, ktorý je schopný doručiť v iterácii, iteratívny prístup k implementácii, alebo potrebné postupy a procesy pri iteráciách. Pri prvých iteráciách sa predpokladá, že sa tímu nepodarí všetko tak, ako si naplánoval. V prípade dlhých iterácií tímu mže trvať pokiaľ zlyhá a keď už zlyhá tak sa ťažko identifikuje dôvod, keďže počas týchto iterácií sa toho môže veľa stať. Krátke iterácie dovoľujú novému tímu rýchlejšie zlyhávať a jednoduchšie analyzovať dôvody zlyhania. Je veľmi dôležité analyzovať s tímom, prečo zlyhal alebo uspel počas iterácie, aby bolo možné sa so svojich chýb, alebo úspechov poučiť.

12.3 Multidisciplinárne tímy

Komunikácia medzi jednotlivými oddeleniami dokáže byť mnohokrát pomalá, primárne ak oddelenia komunikujú medzi sebou prostredníctvom manažmentu. Pokiaľ sa infor-

mácia dostane do ďalšieho tímu môže každá drobnosť ktorá vyžaduje spoluprácu medzi oddeleniami vyžadovať veľa času na vybavenie. Okrem pomalej komunikácie, môže takýmto spôsobom dôjsť aj ku skresleniu predávaných informácií.

Vďaka tímom poskladaným z rôznych disciplín potrebných na doručenie určitej časti produktu sa komunikácia v tíme dokáže zrýchliť enormným spôsobom [2]. Keďže tím pozostáva z viacerých oddelení, jednotliví členovia dokážu vyriešiť mnoho problémov medzi sebou. Taktiež tieto multidisciplinárne tímy by mali mať zodpovednosť za doručenie priradených funkcionalít. Mnohokrát tieto tímy k takejto zodpovednosti treba vycvičiť. Kvalitný multidisciplinárny tím nielenže dokáže doručiť danú časť produktu, ale dokáže aj navrhnúť procesy, ktoré na doručenie bude potrebovať.

Poskladanie takéhoto tímu nemusí byť vôbec jednoduché. Aby tím efektívne fungoval, je dôležité zhodnotiť a nakombinovať rôzne úrovne technických a komunikačných znalostí [41]. Podľa kapitôl 10 a 11 môžeme vidieť, že nie vždy je možné poskladať tím z ideálnych členov majúcich potrebné znalosti a kvality. V takomto prípade je veľmi dôležité, aby boli tímy podporené neustálym mentorovaním či už manažmentom alebo skúsenejšími vývojármi.

12.4 Implementácia procesov

Agilita pri vývoji je veľmi dôležitá. Samotné agilné procesy a metodiky, ktoré sú v dnešnej dobe dostupné sú výbornou pomôckou dosiahnuť spomínanú agilitu. No nie sú nevyhnutnosťou na používanie. Jednotlivé metodiky nám ponúkajú rozhrania, za pomoci ktorých môžeme vyvíjať herné produkty. Rovnako ako pri používaní akéhokoľvek rozhrania v programovacom jazyku, tiež nemusíme využiť všetky jeho funkcie. To isté platí pre agilné procesy. Nie je nutnosťou využívať všetky procesy rovnako ako je opísané v príručkách. Tieto procesy boli vytvorené, aby podporovali tím a projekt, na ktorý sú aplikované. Agilný manifest z kapitoly 6.1 taktiež hovorí "jednotlivci a interakcie nad procesmi a nástrojmi"[26]. Vždy je dôležité porozumieť potrebám tímu rovnako ako aj potrebám produktu. Zároveň je veľmi dôležité brať do úvahy samotné skúsenosti tímu a na základe všetkých týchto premených dizajnovať produkčný proces hry.

V prípadových štúdiách je uvedené ako rozdielne spoločnosti aplikovali rôzne procesy a metodiky. V spoločnosti zaoberajúcej sa mobilnými hrami v kapitole 10 si vybrali spôsob plnej implementácie scrumu a jeho následného prispôsobovania potrebám tímu. Ako sme si aj v tejto štúdii písali, samotní manažéri to ohodnotili ako neefektívny spôsob v danej situácii. Čiže prvotnú implementáciu môžeme nazvať neagilnú. Tým, že manažment a tím neustále vynakladali úsilie za účelom zlepšovania sa, vytvorili správne podmienky pre zefektívnenie a prispôsobenie procesov potrebám tímu a vytvorenie

agilného myslenia v tíme.

V druhej spoločnosti z kapitoly 11 môžeme povedať, že manažment si implementáciu agilných procesov a metodík viac strategicky naplánoval na základe možností tímu a potrieb projektu. Cieľom bolo zachovať momentálnu efektivitu tímu no zároveň ho začať pomaly trénovať k vzájomnej spolupráci medzi oddeleniami. Okrem iného bolo treba definovať a zaviesť základné postupy, ako napríklad navrhovanie technických riešení a schvaľovanie dokumentácie. Tieto zmeny neboli pre tím jednoduché a preto bolo dôležité, aby manažment postupoval pomaly.

Ako je uvedené na začiatku tejto kapitoly, pred zavádzaním nových procesov je veľmi dôležité premyslieť si cieľ, ktorý chceme dosiahnuť a plán, akým sa k takémuto cieľu dostaneme. Existuje veľa príručiek, ktoré pomáhajú tvoriť túto stratégiu alebo opisujú už vytvorené a osvedčené stratégie.

12.5 Reakcia na zmenu

Reakcia na zmenu patrí k riadeniu projektov a je jednou zo základných hodnôt agilného vývoja, ako je uvedené v kapitole 6.1.

Jedná sa o samotný spôsob myslenia počas celého procesu vývoja projektu a zahŕňa všetky aktivity projektového manažmentu. Je to schopnosť zmeniť zavedené normy na základe externých faktorov za účelom úspešného doručenia projektu.

V jednotlivých štúdiach rozoberáme ako sa manažment počas vývoja snažil neustále prispôsobiť rôznym potrebám projektu aj tímu. Tieto potreby sa neustále menia, dôležité je, aby sa na ne nejakým spôsobom reagovalo.

V prípade mobilnej hry v kapitole 10 producenti neustále analyzovali postup vývoja funkcionalít a porovnávali ho s počiatočným plánom. Akonáhle zistili nejaké potenciálne riziko, snažili sa prispôsobiť priority backlogu, rozsah jednotlivých funkcionalít, prípadne plán tak, aby sa riziko eliminovalo. Ak nie je možné eliminovať riziko úplne, mali by sme sa ho pokúsiť znížiť na minimum. Aj keď prvotná implementácia procesov bola pre väčšinu tímov rovnaká, na základe spätnej väzby vývojárov ohľadom zvýšenia ich efektivity sa neustále prispôsobovali jednotlivé postupy.

Podobne to platí aj pre prémiovú hru v kapitole 11, kde sa jednotlivé procesy priamo prispôsobovali potrebám jednotlivých funkcionalít, alebo feature tímov.

ZÁVER

Cieľom diplomovej práce je poskytnúť súčastným a aj budúcim producentom nástroj, vďaka ktorému získajú základné povedomie o tom ako viesť herné projekty. Na dosiahnutie tohoto cieľa v teórii rozoberáme problematiku vývoja hier a jej jednotlivých častí. Vysvetľujeme jednotlivé fázy vývoja produktu v závislosti na platforme. Rozoberáme taktiež ciele a zodpovednosti oddelení potrebných pre vývoj herných projektov. Ďalšou témou teoretickej časti sú spôsoby vývoja hier. V rámci tejto témy opisujeme waterfall model a následne sa zameriavame na agilné metodiky. V praktickej časti testujeme teoretické základy v praxi. V rámci toho boli vykonané dve prípadové štúdie, v ktorých rozoberáme procesy a metodiky v dvoch rôznych spoločnostiach.

Prvá štúdia je zameraná na implementáciu scrumu v hernej spoločnosti pracujúcej na mobilnej hre. Skúmame časové obdobie približne dvoch až troch mesiacov, kedy manažment zavádzal nové procesy a metodiky do bežiaceho vývoja produktu. Procesy sa zavádzali počas práce na majoritnej aktualizácii, ktorá pre tím predstavovala výzvu nielen náročnosťou jednotlivých funkcionalít, ale aj veľmi tesnými termínmi. Okrem práce na nových funkcionalitách sa tím musel potýkať taktiež s výzvami spojenými so zlým technickým stavom niektorých častí hry, ktoré ho v mnohom spomaľovali. Manažment začal implementovať scrum so všetkými procesmi do jednotlivých multifunkčných tímov postavených na základe potrieb jednotlivých funkcionalít. Tieto procesy si tímy následne po iteráciách upravovali podľa potreby a preferencií.

V druhej štúdii opisujeme vývoj prémiového projektu pre počítačovú platformu. V tomto prípade skúmame časové obdobie kedy vyvíjaný projekt prechádzal z konceptovej fázy do vertical slice. Keďže sa jednalo o mladšiu spoločnosť, muselo byť od začiatku vytvorených veľa základných procesov. Medzi tieto procesy patrilo, napríklad, písanie dokumentácie, schvaľovanie dokumentácie, alebo vytvorenie postupu pre tvorbu jednotlivých funkcionalít, ktorý obsahoval ako preprodukciu tak aj produkciu. Manažment si v tomto prípade zvolil rozdielnu stratégiu vedenia projektu, kedy vytvoril feature tímy, ktoré pracovali v šprintoch za účelom trénovania ľudí k vzájomnej spoluprácii a iteratívnemu vývoju.

V poslednej časti spisujeme základné doporučenia pre vedenie herných projektov na základe teoretických znalostí a prípadových štúdií. Medzi tieto doporučenia partí dôraz na kvalitnú a aktuálnu dokumentáciu, iteratívny vývoj, práca s multidisciplinárnymi tímami, reakcia na zmenu a implementácia procesov na základe potrieb tímu a projektu.

ZOZNAM POUŽITEJ LITERATÚRY

[1] TRETKOFF, Ernie, CHODOS, Alan, ed. *This Month in Physics History: October 1958: Physicist Invents First Video Game.* In: APS Physics [online]. AMERICAN PHYSICAL SOCIETY, 1995, October 2008 [cit. 2022-04-24]. Dostupné z: `https://www.aps.org/publications/apsnews/200810/physicshistory.cfm`

[2] KEITH, Clinton. *Agile Game Development with Scrum: Deliver Better Games Faster, On Budget—And Make Game Development Fun Again!.* Pearson Education, May 23, 2010. ISBN 978-0-321-61852-8.

[3] *A GUIDE TO THE DIFFERENT TYPES OF VIDEO GAME PLATFORMS.* In: Glassy [online]. 2019 [cit. 2022-05-07]. Dostupné z: `https://glassyeyewear.com/blogs/article/a-guide-to-the-different-types-of-video-game-platforms`

[4] *Mobile, Console or VR: Game Development Cost Explained.* In: Game Ace [online]. 2017 [cit. 2022-05-07]. Dostupné z: `https://game-ace.com/blog/mobile-console-vr-game-development-cost-explained/`

[5] *WIJMAN, Tom. Mobile Revenues Account for More Than 50% of the Global Games Market as It Reaches $137.9 Billion in 2018.* In: Newzoo [online]. 2018 [cit. 2022-05-17]. Dostupné z: `https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile\-games-take-half/`

[6] *SHAREEF, Tashreef. 5 best cross-platform game engines for game developers.* In: Windows Report [online]. 2021 [cit. 2022-05-07]. Dostupné z: `https://windowsreport.com/cross-platform-game-engines/`

[7] *How to Make a Cross-Platform Game from Scratch.* In: Game Ace [online]. 2021 [cit. 2022-05-07]. Dostupné z: `https://game-ace.com/blog/make-cross-platform-game/`

[8] *Mobile Gaming Market to Reach $139.5 Billion by 2026.* In: PR Newswire [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://www.prnewswire.com/news-releases/global-mobile-gaming-market-to-reach-139-5-billion-by-2026–301498967.html

[9] ROTMAN, David. *We're not prepared for the end of Moore's Law: It has fueled prosperity of the last 50 years. But the end is now in sight.* In: MIT Technology Review [online]. 2021 MIT Technology Review, February 24, 2020 [cit. 2022-

04-24]. Dostupné z: https://www.technologyreview.com/2020/02/24/905789/were-not-prepared-for-the-end-of-moores-law/

[10] PICKELL, Devin. *The 7 Stages of Game Development.* In: G2 [online]. G2.com, október, 2019 [cit. 2022-04-24]. Dostupné z: https://www.g2.com/articles/stages-of-game-development

[11] FLORENCIO, Harry. *What you should take out of Pre-Production.* In: Game Developer [online]. Informa PLC Informa UK Limited [cit. 2022-04-24]. Dostupné z: https://www.gamedeveloper.com/blogs/what-you-should-take-out-of-pre-production

[12] FUTTER, Mike. *Why gamers don't fully understand alpha and beta tests.* In: The GameDev Business Handbook [online]. Bithell Games, október, 2017 [cit. 2022-04-24]. Dostupné z: http://www.gamedevbizbook.com/production/gamers-alpha-beta-tests/

[13] *Beta.* In: What Games Are [online]. [cit. 2022-05-08]. Dostupné z: https://www.whatgamesare.com/beta.html

[14] STEFYN, Nadia. *How video games are made: the game development process.* In: CG Spectrum [online]. CG Spectrum, október, 2019 [cit. 2022-04-24]. Dostupné z: https://www.cgspectrum.com/blog/game-development-process

[15] SOBOLEV, Jacob. *What Is A Soft Launch? How To Plan It?.* In: Gaming Shift [online]. Gaming Shift [cit. 2022-04-25]. Dostupné z: https://gamingshift.com/soft-launch/

[16] STREAMCHECKUP. Game development team structure: from design to publish. Medium [online]. [cit. 2022-03-15]. Dostupné z: https://medium.com/@streamcheckup/game-development-team-structure-from-design-to-publish-4a3225762ef4

[17] SCHELL, Jesse. *The art of game design: a book of lenses. Second edition.* Boca Raton, [2015]. ISBN 978-1-4665-9864-5.

[18] CHRON CONTRIBUTOR. *The Description of a Game Programmer.* Chron [online]. 4747 Southwest Freeway, Marec 04, 2021 [cit. 2022-03-15]. Dostupné z: https://work.chron.com/description-game-programmer-30762.html

[19] *What Are the Most Popular Game Engines?.* Perforce [online]. Copyright © 2022 Perforce Software, April 24, 2020 [cit. 2022-03-15]. Dostupné z: https://www.perforce.com/blog/vcs/most-popular-game-engines

[20] *Technical Design Document and Game Design Document.* In: Study Tonight [online]. Studytonight Technologies Pvt. [cit. 2022-04-25]. Dostupné z: https://www.studytonight.com/3d-game-engineering-with-unity/tdd-and-gdd

[21] *THE ROLE AND WORK OF A GAME ARTIST IN THE GAMES INDUSTRY.* Mages Institute [online]. Singapore 237978: MAGES Institute of Excellence Pte. [cit. 2022-03-15]. Dostupné z: https://mages.edu.sg/blog/the-role-and-work-of-a-game-artist-in-the-games-industry/

[22] FREEMAN, Will. *Career Guide: Becoming A Game Producer.* Ask About Games [online]. AAG, 23 február, 2018 [cit. 2022-03-16]. Dostupné z: https://www.askaboutgames.com/career-guide-becoming-a-game-producer

[23] HUGHEY, Douglas. *The Traditional Waterfall Approach.* In: Comparing Traditional Systems Analysis and Design with Agile Methodologies [online]. Copyright 2009 [cit. 2021-04-17]. Dostupné z: https://www.umsl.edu/~hugheyd/is6840/waterfall.html

[24] *The Game Creation Process: Part 2: Designing the Idea.* In: Game Jolt [online]. Game Jolt, 2014 [cit. 2022-04-25]. Dostupné z: https://gamejolt.com/p/the-game-creation-process-part-2-designing-the-idea\-designing-yo-viq5rk2t

[25] PETERSEN, Kai, Claes WOHLIN a Dejan BACA. *The Waterfall Model in Large-Scale Development* [online]. SE-37225 Ronneby, Sweden, 2009 [cit. 2022-04-25]. Dostupné z: https://www.diva-portal.org/smash/get/diva2:835760/FULLTEXT01.pdf. Blekinge Institute of Technology.

[26] BECK, Kent, Mike BEEDLE, Arie van BENNEKUM, et al. *Agile Manifesto: The 12 Principles of Agile.* Agile Alliance [online]. The Agile Manifesto Authors, 2001, 1 [cit. 2021-04-17]. Dostupné z: https://www.agilealliance.org/wp-content/uploads/2019/09/agile-manifesto-download-2019.pdf

[27] *MoSCoW Prioritization.* In: ProductPlan [online]. [cit. 2022-05-08]. Dostupné z: https://www.productplan.com/glossary/moscow-prioritization/

[28] STARLOOP STUDIOS. *Best Agile Practices in Game Development.* In: Starloop [online]. Starloop Studios, 2020 [cit. 2022-04-25]. Dostupné z: https://starloopstudios.com/best-agile-practices-in-game-development/

[29] TUTORIALSPOINT. *Extreme Programming Tutorial* [online]. 2015 [cit. 2022-04-26]. ISBN TP00221. Dostupné z: https://www.tutorialspoint.com/ebook/extreme_programming_tutorial/index.asp

[30] BECK, Kent. *Test-Driven Development: By Example.* Boston: Addison Wesley, 2002. ISBN 0-321-14653-0.

[31] SHORE, Jim. *Continuous Design.* In: MartinFowler.com [online]. IEEE Software, 2004 [cit. 2022-04-26]. Dostupné z: `http://www.martinfowler.com/ieeeSoftware/continuousDesign.pdf`

[32] WILLIAMS, Laurie a Robert KESSLER. *Pair Programming Illuminated.* Boston: Addison Wesley, 2002. ISBN 0-201-74576-3.

[33] BECK, Kent a Cynthia ANDRES. *Extreme programming explained: embrace change.* 2nd ed. Boston: Addison-Wesley, 2005. ISBN 03-212-7865-8.

[34] *The Agile Journey: A Scrum overview.* In: Pm-partners [online]. PM-Partners Group, 2021 [cit. 2022-04-26]. Dostupné z: `https://www.pm-partners.com.au/the-agile-journey-a-scrum-overview/`

[35] TUTORIALSPOINT. *Scrum: Agile framework for completing complex projects* [online]. Tutorialspoint [cit. 2022-04-26]. Dostupné z: `https://www.tutorialspoint.com/scrum/scrum_tutorial.pdf`

[36] WEST, Dave. *Scrum roles and the truth about job titles in scrum.* In: Attlasian Agile Coach [online]. Atlassian [cit. 2022-04-26]. Dostupné z: `https://www.atlassian.com/agile/scrum/roles`

[37] *4 types of Agile ceremonies and how to manage them.* In: Monday [online]. 2021 [cit. 2022-04-26]. Dostupné z: `https://monday.com/blog/rnd/agile-ceremonies/?marketing_source`

[38] *Publish your app.* In: Google Help [online]. [cit. 2022-04-30]. Dostupné z: `https://support.google.com/googleplay/android-developer/answer/9859751?hl=en`

[39] *App Review.* In: App Store [online]. [cit. 2022-04-30]. Dostupné z: `https://developer.apple.com/app-store/review/`

[40] *JENA, Satyabrata. Overview Software Documentation.* In: GeeksforGeeks [online]. 2021 [cit. 2022-05-08]. Dostupné z: `https://www.geeksforgeeks.org/overview-software-documentation/`

[41] *PRATT, Mary K. Cross-functional team.* In: TechTarget [online]. [cit. 2022-05-08]. Dostupné z: `https://www.techtarget.com/searchcio/definition/crossfunctional`

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

F2P	Free to play
DLC	Downloadable content
GDD	Game design document
TDD	Test driven development
CI	Continuous integration
KPI	Key performance indicator
UI	User interface
LTV	Life time value

ZOZNAM OBRÁZKOV

ZOZNAM PRÍLOH

P I. CD s diplomovou prácou

PRÍLOHA P I. CD S DIPLOMOVOU PRÁCOU

Príloha obsahuje elektronickú verziu diplomovej práce.