4 RAZVOJ VIDEO IGER

Veliko je interesa na temo raziskovanja discipline metodologij razvoja video iger (Ampatzoglou in Stamelos 2010, 889). Opaziti je bilo povečano število raziskav na tem področju. Poleg tega aktivnost raziskovanja inženiringa video iger v primerjavi z raziskovanjem razvoja programske opreme raste hitreje (Ampatzoglou in Stamelos 2010, 893). Od leta 2008 je bilo opaziti več študij na temo upravljanja razvoja video iger (Ampatzoglou in Stamelos 2010, 894), ki zajema tehnike upravljanja s procesi (Novak 2012, 351). To potrdi tudi sistematičen pregled literature, ki so ga izvedli Aleem in kolegi (2016).

Razvidno je da je število raziskav doseglo vrhunec med leti 2008-2012 (PRILOGA A). Kljub porastu raziskav smo opazili, da se organizacije poslužujejo različnim pristopom in njihovim hibridom predstavljenih v prejšnjem poglavju. Našli smo le nekaj raziskav, ki so definirale razvojni proces, metodologijo ali življenjski cikel izključno za razvoj video iger. Morda je krivo to, da v literaturi še ni bilo podrobneje predstavljenih najboljših praks ali strategij za razvoj iger. Poleg tega je bila disciplina okrnjena orodja za optimizacijo procesa, ki je ključno za izboljšanje procesa (Aleem, Capretz, in Ahmed 2016, 26). Menimo, da celovito podedovanje procesnih modelov in metod iz standardnega razvoja programske opreme za igre ni primerno. Zaradi svoje agilnosti, tendence po zmanjšanju dokumentacije, multidisciplinarnosti, kreativnosti, abstraktno definiranih zahtev (Ruonala 2016, 5), krajšega življenjskega cikla (O’Hagan in O’Connor 2015, 3) posedujejo lastnosti, ki jih evidentno ločuje od standardnega razvoja programske opreme.

4.1 ZNAČILNOSTI IGER

Za razvoj iger je jasno, da poseduje značilnosti, ki niso prisotne v ostalih disciplinah inženiringa programske opreme (Abrahamsson in dr. 2017, 1).

Igre se osredotočajo na igranje (Esposito 2005, 3). Igranje predstavlja element, ki emocionalno pritegne igralce. Je nekakšno stanje, ki je podobno zabavi (Fullerton 2008, 91). Posledično se igre smatrajo za najbolj ekstremne hedonične informacijske sisteme, pa vendar se pri utilitarnih namenih uporabe uporabna motivacija premakne iz zabave v uporabnost (Hamari in Keronen 2017, 136).

Igre lahko vodijo tako intrinzične kot ekstrinzične motivacije (Novak 2012, 198). Številne študije kažejo pozitivne učinke učnih iger na področju motivacije do učenja in učnega izkoristka. Motivacija in pozitivne emocije sta pomembna faktorja pri učenju (Imlig-Iten in Petko 2014, 151), katerih uravnoteženost igre predstavlja efektivnost, ki je pridobljena s konsistenco zahtevnosti in zabave pri igranju[[1]](#footnote-1) (Novak 2012, 202). To predstavlja zabavno izkušnjo, ki je najpomembnejše merilo uspeha iger (Cooper in Scacchi 2015, 12). V razvoju iger se uporablja termin uporabniška izkušnja[[2]](#footnote-2), ki jo z iterativnim procesom upoštevanja uporabniških povratnih informacij oblikovalci izboljšujejo tekom procesa razvoja (Fullerton 2008, 2). Potrebno je razumevanje, da igra ni izkušnja vendar je izkušnja nekaj kar igra omogoča (Schell 2008, 10). S povratnimi informacijami je povezana tudi pozornost na defekte. Zavoljo tega osrednja aktivnost razvoja video iger navadno postane konstantno testiranje igranja (Cooper in Scacchi 2015, 10). To aktivnost predstavlja testni protokol in velja za iterativni proces med načrtovanjem, razvojem in testiranjem (Sylvester 2013, 295). Zato se razvoji iger nagibajo od tradicionalnih življenjskih ciklov razvoja k bolj inkrementalnim, kateri na podlagi uporabniških povratnih informacij izpopolnjujejo ali izboljšujejo verzije programa (Cooper in Scacchi 2015, 10).

Zanimivost igre je tudi v njenem življenjskem ciklu razvoja. Njen življenjski cikel se nikoli ne konča. Navadno manj izkušena ekipa ostane in skrbi za posodabljanje virtualnega sveta (Bartle 2003, 93). Izdajajo se popravki, posodobitve in razširitve v glavnem z namenom daljšanja življenjske dobe igre (Novak 2012, 365). Posebnost je tudi vključenost uporabnikov v življenjski cikel razvoja, ki se bistveno razlikuje od standardnega razvoja programske opreme. Uporabniki so lahko vključeni v beta testiranja (odprto beta testiranje) programske opreme in pripomorejo k zagotavljanju kvalitete v zameno za brezplačno igranje (Levy in Novak 2009, 54) ali pa modificirajo[[3]](#footnote-3) igre s čimer se vključijo v primitiven razvojni proces igre (Cooper in Scacchi 2015, 5).

4.2 PROCESI V PRAKSI

Po podatkih spletne ankete v Avstrijski industriji iger 23% podjetij razvija igre z ad-hoc pristopi, 77% pa jih uporablja Scrum ali XP. Verjetno je k rezultatu botrovala velikost samih podjetij, saj ima 85% podjetij vsaj 4 zaposlene medtem, ko ima le 15% podjetij 15 ali več zaposlenih. Ne glede na rezultate so vsa podjetja nakazala uporabo nekakšnih fleksibilnih, sekvenčnih ali agilnih pristopov (Musil in dr. 2010, 5).

Raziskava, ki so jo naredili O'Hagan in kolegi je pokazala, da se pri razvoju iger uporablja 47% agilnih in 53% hibridnih procesov. Izmed 404 študij so identificirali 23 procesnih modelov izmed katerih so bili vidnejši: XP, Scrum, Kanban, rapidni, inkrementalni in komponentni razvojni modeli. Medtem, ko so prvi trije našteti strogo agili, vsi bazirajo na iteracijah. Vsi modeli so se razlikovali le po številu iteracij, katerih število je bilo večje pri agilnih in manjše pri hibridnih procesih (O’Hagan, Coleman, in O’Connor 2014, 187).

Koutonen in Leppänen (2013) sta izvedla raziskavo o uporabi agilnih procesov v finski industriji igre. Raziskava je zajemala najmanjša podjetja kot tudi enega največjih (npr. Rovio Entertainment). Vsa podjetja razen enega so v vsaj enem koraku procesa uporabili agilne metode (Koutonen in Leppänen 2013, 12).

Na podlagi dokumentov obdukcij razvoja sta Petrillo in Pimenta (2010) naredila raziskavo, ki je ugotavljala uporabo agilnih praks v razvoju. Ugotovila sta, da se v razvoju uporabljajo agilne prakse. Uporaba se lahko s prakticiranjem agilnih praks izvaja povsem nezavedno in celo neformalno (Petrillo in Pimenta 2010, 14).

Prav tako agilne pristope podpira Fullerton, saj meni, da je Scrum primeren za reševanje zapletenih problemov oblikovanja iger (Fullerton 2014, 369). Svoje mnenje podaja John Comes[[4]](#footnote-4), ki meni, da naj opustimo kaskadne procese in se sprejmemo agilne (John Comes v Novak 2012, 366).

4.3 PROCESNI MODELI ZA IGRE

Babu in Maruthi (2013) podrobneje definirata faze življenjskega cikla za igre. Faze si sledijo linearno: izdelava zgodbe, razvoj skripte, študija izvedljivosti, promocijski demo, oblikovanje, oblikovanje postavitev, modeliranje, teksturiranje, animiranje, oblikovanje stopenj, kodiranje, testiranje, razhroščevanje, integracija, testiranje igranja (Babu in Maruthi 2013, 1491). Z definicijo življenjskega cikla podata razširjen vpogled v faze razvoja iger (Babu in Maruthi 2013, 1502).

Ramadan in Widyani (2013) na podlagi ključnih aktivnosti različnih organizacij in raziskovalcev predlagata GDLC[[5]](#footnote-5). Pristop predlaga faze: iniciacijo, pred produkcijo, produkcijo, testiranje, beta in izdajo. Iniciacija služi za stvaritev koncepta igre in enostaven opis specifikacije. Pred produkcija zajema revidiranje oblikovanja in izdelava prototipa. Po tej fazi je končan GDD[[6]](#footnote-6) na podlagi katerega je izdelan prototip. V prvi iteraciji produkcije služi prototip za temelje in strukturo v nadaljnjih pa se ga izboljšuje. Produkcija se konča z izdelanim prototipom, ki predstavlja celoto (Ramadan in Widyani 2013, 98). Sledi testiranje, ki izda poročilo defektov in seznam izboljšav. Rezultat testiranja odloča ali se razvoj nadaljuje v beta fazo. Slednja predstavlja identifikacijo defektov in povratnih informacij uporabnikov. Iz beta lahko ponovno sledi faza produkcije z namenom izboljšanja igre ali faza izdaje v kolikor je rezultat beta testiranja zadovoljiv (Ramadan in Widyani 2013, 99).

Aslan in Balci (2015) predstavita metodologijo za kompleksni razvoj iger GAMED[[7]](#footnote-7). PRILOGA B prikazuje življenjski cikel DEG[[8]](#footnote-8), ki je osnova metodologije. DEG cikel je sestavljen iz štirih faz: oblikovanja igre, oblikovanja programske opreme, implementacije (izdaja) in učenja na podlagi iger (povratne informacije) (Aslan in Balci 2015, 309). Metodo označujeta za iterativno, saj pričakuje povratne tranzicije. V kolikor se pojavi potreba se postavimo en korak nazaj in ponovimo delo. Tipično se pomikamo naprej in nazaj med procesi dokler ne dosežemo zadovoljive kvalitete delavnih produktov (Aslan in Balci 2015, 310). Kot podporni proces metodologija uporablja spiralni model, ki ga aplicira v fazo oblikovanja igre (PRILOGA C). Oblikovanje poteka po spiralnem vzorcu pri čemer vsaka iteracija pomeni izpopolnjen dokument specifikacije ideje. Iteracija vsebuje aktivnosti: prototipiranja, testiranja igranja, evalvacija in analize tveganj. Vzorec se nadaljuje dokler kvaliteta oblikovanja ni potrjena v aktivnosti evalvacije. Končni dokument te aktivnosti predstavlja specifikacija oblikovanja igre (Aslan in Balci 2015, 313). Specifikacijo oblikovanja se v aktivnosti zbiranja zahtev nadgradi v dokument specifikacije potreb. Specifikacija potreb se po potrebi zahtevne arhitekture nadgradi v dokument specifikacije arhitekture. V zadnji aktivnosti faze oblikovanja se predhodni dokument (specifikacija arhitekture ali specifikacija potreb) posodobi v dokument specifikacije oblikovanja programske opreme iger (Aslan in Balci 2015, 315–316). Na podlagi tega dokumenta se v fazi implementacije izvede aktivnost programiranja v kateri se zgradi DEG ali njene komponente. Metoda GAMED predvideva vzdrževanje, ki se izvaja na podlagi povratnih informacijah. Po potrebi posodobitve se ponovno izvede celoten življenjski cikel (Aslan in Balci 2015, 317).

Barbosa (2017) predstavi Game-Scrum, ki je hibrid Scrum in XP procesnih modelov (PRILOGA D). Scrum skrbi za upravljanje projekta medtem, ko XP zagotavlja inženiring. Procesni hibrid je primeren za razvojne ekipe z malo ali nič izkušnjami (Barbosa 2017, 293). Proces se deli na faze: pred produkcija, faza izdelave GDD, produkcija in post produkcija. V pred produkciji poteka iskanje dejavnika zabave, izdelava idealnega koncepta in oblikovanja. Delo poteka po principu metode poskušanja in popravljanja napak. V tej fazi se predvideva izgradnja enostavnega prototipa, ki je zaradi narave svoje hitre konstrukcije navadno zavržen. Sledi izgradnja GDD, ki je v produkcijski fazi preveden v dnevnik zaostankov. Produkcija poteka iterativno, priporoča Kanban metodo za kreacijo umetnin in podajanje časovnih okvirjev, mejnikov njihove izdelave (Barbosa 2017, 293). V kolikor je v ekipi več programerjev se priporoča uporaba Scrum in XP tehnik. Po končani igri se uporabi testiranje igranja za zagotavljanje kvalitete in dejavnika zabave. Zadnje dejanje v razvoju predstavlja izdelava dokumenta obdukcije razvoja, ki služi za identifikacijo pomanjkljivosti v preteklem razvojem procesu (Barbosa 2017, 294).

4.4 POMANJKLJIVOSTI PROCESNIH MODELOV ZA IGRE

Različice modelov se pojavijo tudi v sivi literaturi. GUP[[9]](#footnote-9) predstavi rang procesnih pristopov od RUP do XP. Vendar kljub vsemu GUP ne definira poteka aktivnosti niti ne poda osnovnih usmeritev kako združiti estetske discipline razvoja z inženirskimi (Wilson Brotto Furtado, 22). Flood (2003) pri definiranju GUP predstavi in komentira GWP[[10]](#footnote-10). Kljub številnim pomanjkljivostim ta proces prvič predstavi artefakte, ki so specifični za igre (Wilson Brotto Furtado, 21) (specifikacija igre, umetniška biblija, tehnična specifikacija) („Game Unified Process - EcuRed“ 2018). GWP omenjata tudi Flynt in Salem (2004) kot pristop, ki so ga uporabili razvijalci igre Ankh. GWP ima težavo, da predstavlja linearen proces. Ocenjevalci, testerji, ravnatelji ne vidijo produkta dokler je ta v razvoju. V realnosti je tako, da se po evalvaciji navadno zahtevajo spremembe. To zahteva, da razvoj ponovi faze, ki so bile predvidoma že končane („Game Unified Process - EcuRed“ 2018). Razvoj iger ni linearen proces (Flynt in Salem 2004, 151). V praksi je tendenca po ogibanju uporabe kaskadnih pristopov, ker je nemogoče definirati in načrtovati program vnaprej (Rucker 2002, 38). V praksi je GWP (razvoj igre Ankh) predstavljal napor. Razvoj igre potrebuje fleksibilnost zato, da lahko ekipa vključi spoznanja, ki so se identificirala tekom razvoja z namenom konstantne izboljšave kvalitete (Flynt in Salem 2004, 18). GWP in prav tako GUP ne podajata priporočila uporabe ali referenčne modele za aplikacijo teh procesov, k temu najverjetneje vpliva tajna narava industrije iger (Wilson Brotto Furtado, 22).

Med neuspešne spada tudi model, ki ga objavi Demachy (2003) na Gamasutra. Interpretira XP metodologijo in predlaga XGD[[11]](#footnote-11) za video igre. Vendar pristop ne zagotavlja strukturiranih usmeritev, deloma zato, ker gre za agilno metodologijo (Wilson Brotto Furtado, 22).

Poleg naštetih se v akademski literaturi pojavita tudi dve pobudi procesov AGP[[12]](#footnote-12) in Predpisujoča metodologija za razvoj video iger, ki sta nastala v okviru zaključnih nalog. Obema predlogoma primanjkuje evalvacijski proces za določanje njune ustreznosti (Wilson Brotto Furtado, 22). Poleg tega igre niso predpisujoče (Koster 2013, 156). Tipične lastnosti razvoja iger kot so oblikovanje ali definiranje igranja je težko ali celo nemogoče specificirati brez različice sistema, ki jo lahko preizkusimo (Petrillo in Pimenta 2010, 10). Zato kakršnikoli predpisujoči (sekvenčni) pristopi niso priporočljivi.

Kljub temu, da predlogi procesov v prejšnjem poglavju predstavljajo bolj strukturirane jih spremljajo številne pomanjkljivosti. V tem razdelku smo že večkrat komentirali linearnost procesov, ki se izražajo s predpisovanjem poteka razvoja. Kljub temu, da Babu in Maruthi (2013) podrobneje definirata življenjski cikel in opišeta posamezne faze je model linearno orientiran. GDLC je v nasprotju s tem visoko iterativen, ki se lahko iz faze beta testiranja vrne v pred produkcijo, medtem, ko med produkcijo poteka iterativna izboljšava prototipa. Čeprav pristop podpira post produkcijsko fazo, ki jo označuje za izdajo, zelo slabo opisuje aktivnosti, ki se izvajajo v tej fazi. V razvoju je postala nekakšna praksa, da so popravki takoj po izdaji, saj zaradi hitenja ni mogoče testirati vseh funkcionalnosti (Bates 2004, 216). Glavni razlog je ujemanje dobavnih rokov in najbolj priljubljenih časov izdaj (božič) (Ruonala 2016, 8). Zato je bistvenega pomena, da je proces nadaljnje podpore dobro definiran. Poleg tega GDLC-ju primanjkuje definiranja artefaktov predvsem dokumentacije. Poleg prototipa, GDD obstajajo še umetniška sredstva, ki lahko že med produkcijo vplivajo na testiranje igranja. (Sylvester 2013, 286). Posledično GDLC tako ne definira estetskih disciplin produkcije. V nasprotju z GDLC je metoda GAMED podprta z dokumentacijo. Težava je le v tem, da je dokumentacija zgrajena na dopolnjevanju. V PRILOGA F je prikaz artefaktov, ki jih avtorji identificirajo med razvojem igre. Čeprav ni določenih standardov za kreiranje dokumentacije veljajo določeni dokumenti za osnovne komponente trdnih temeljev. In te predstavljajo koncept, predlog igre, GDD, TDD[[13]](#footnote-13), vodnik umetniškega sloga projektni načrt (Novak 2012, 380-394) Vsi so predstavljeni kot samostojni dokumenti, nekatere ekipe lahko celo razdelijo GDD tudi več dokumentov, ki posebej predstavljajo napredovanje po stopnjah, igranje in zgodbo (III 2004, 319). GAMED predstavlja odlično podlago za definiranje učnih iger vendar je pristop terminološko oslabljen. Pri drugih avtorjih se določena terminologija ponavlja tako pri definiranju artefaktov (PRILOGA F) kot pri definiranju faz razvoja (PRILOGA G), katere pa pristop ni podedoval. Največjo težavo pristopa predstavlja obnašanje cikla. Faza oblikovanja je iterativen proces (Schell 2008, 79), kar predvideva tudi GAMED, vendar se razvoj prototipa konča s fazo oblikovanja in definiranja dokumenta, ki po opisu predstavlja GDD. Prototip je ob koncu zavržen, razvoj programske opreme pa se ponovno izvede tokrat linearno na podlagi podrobno definirane dokumentacije (GDD), ki služi kot funkcionalna specifikacija. V nasprotju s funkcionalnimi specifikacijami je GDD zaradi svoje narave bolj organski in dinamičen. (III 2004, 310) in se v toku produkcije dnevno spreminja (Novak 2012, 391). Ker je prototip zavržen je opravljen presežek dela, saj je potrebno produkt ponovno razvijati. Prav tako se za iterativni razvoj smatra izvajanje iteracij med oblikovanjem igre, prototipiranjem in testiranjem (Novak 2012, 367), kar je v nasprotju z linearno realizacijo zahtev iz dokumentacije. Ta lahko privede do pozlačevanja[[14]](#footnote-14), ki pomeni sprejemanje nemogočih zahtev (Rucker 2002, 27) in je nasprotje iterativnemu razvoju (Novak 2012, 367).

1. To predstavlja game flow. [↑](#footnote-ref-1)
2. (angl.) User Experiece-UX. [↑](#footnote-ref-2)
3. (angl.) Modding. Razvijalci izdajo orodja za modificiranje igre katere uporabniki manipulirajo za ustvarjanje novih vsebin v igri. Za začetnika moddinga se smatrajo razvijalci Doom. [↑](#footnote-ref-3)
4. Avtor igre Wolfshade MUD, senior oblikovalec na projektu Earth & Beyond, pri Electronic Arts Los Angeles izda: Command & Conquer: Generals – Zero Hour in Lord of the Rings: The Battle for Middle-Earth, pri Powered Games izda: Superme Commander in Demigod. [↑](#footnote-ref-4)
5. Game development lifecycle. [↑](#footnote-ref-5)
6. Game design document. [↑](#footnote-ref-6)
7. diGital educAtional gaMe dEvelopment. [↑](#footnote-ref-7)
8. Digital education game. [↑](#footnote-ref-8)
9. Game Unified Process. [↑](#footnote-ref-9)
10. Game Waterfall Process. [↑](#footnote-ref-10)
11. eXtreme Game Development Process. [↑](#footnote-ref-11)
12. Agile Game Process. [↑](#footnote-ref-12)
13. Tehnical Design Document. [↑](#footnote-ref-13)
14. Gold-plating. [↑](#footnote-ref-14)