UNIVERZA V LJUBLJANI   
FAKULTETA ZA DRUŽBENE VEDE

Andraž Gruden  
**NASLOV**Diplomsko delo

Ljubljana, 2018

UNIVERZA V LJUBLJANI   
FAKULTETA ZA DRUŽBENE VEDE

Andraž Gruden  
Mentor: izr. prof. dr. Jaroslav Berce

**NASLOV**

Diplomsko delo

Ljubljana, 2018

ZAHVALA



*Kardeljeva ploščad 5*

*1000 Ljubljana, Slovenija*

*telefon 01 58 05 128*

**I Z J A V A O A V T O R S T V U**

diplomskega dela

Spodaj podpisani/-a      , z vpisno številko      , sem avtor/-ica diplomskega dela z naslovom:      .

S svojim podpisom zagotavljam, da:

* je predloženo diplomsko delo izključno rezultat mojega lastnega raziskovalnega dela;
* sem poskrbel/-a, da so dela in mnenja drugih avtorjev oz. avtoric, ki jih uporabljam v predloženem delu, navedena oz. citirana v skladu s fakultetnimi navodili;
* sem poskrbel/-a, da so vsa dela in mnenja drugih avtorjev oz. avtoric navedena v seznamu virov, ki je sestavni element predloženega dela in je zapisan v skladu s fakultetnimi navodili;
* sem pridobil/-a vsa dovoljenja za uporabo avtorskih del, ki so v celoti prenesena v predloženo delo in sem to tudi jasno zapisal/-a v predloženem delu;
* se zavedam, da je plagiatorstvo – predstavljanje tujih del, bodisi v obliki citata bodisi v obliki skoraj dobesednega parafraziranja bodisi v grafični obliki, s katerim so tuje misli oz. ideje predstavljene kot moje lastne – kaznivo po zakonu (Zakon o avtorskiin sorodnih pravicah (UL RS, št. 16/07-UPB3, 68/08, 85/10 Skl.US: U-I-191/09-7, Up-916/09-16)), prekršek pa podleže tudi ukrepom Fakultete za družbene vede v skladu z njenimi pravili;
* se zavedam posledic, ki jih dokazano plagiatorstvo lahko predstavlja za predloženo delo in za moj status na Fakulteti za družbene vede;
* je elektronska oblika identična s tiskano obliko diplomskega dela ter soglašam z objavo diplomskega dela v zbirki »Dela FDV«;
* je diplomsko delo lektorirano in urejeno skladno s fakultetnim Pravilnikom o diplomskem delu.

V Ljubljani, dne       Podpis avtorja/-ice:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Proces izdelave video iger: Proces razvoja video iger: sistematična izgradnja procesnega modela za razvoj iger s pomočjo meta-metode MetaME in praks Agilnega modeliranja

Proces razvoja video iger: sistematični pristop inženiringa procesa za razvoj iger z uporabo meta-metode MetaME in Agilnega modeliranja

Game Development Process: A Systematic Approach towards Process Engineering with the use of meta-method MetaME and Agile Modeling

Ključne besede: proces, metoda, inženiring.

VSEBINA

1 UVOD

Računalniška tehnologija je danes prisotna v domala vseh sferah družbenega življenja. Eskalacija tehnološkega razvoja je v zadnjih letih privedla do tega, da se je računalniška tehnologija iz laboratorijev preselila v naše domove in znašla v sleherni sredini našega socialnega življenja. Hedonično dimenzijo uporabnosti računalniške tehnologije v prvi vrsti predstavljajo video igre. Igrajo se z namenom doživljanja novih izkušenj, pridobivanja občutka dosežka, za interakcijo s prijatelji, družino in za preživljanje prostega časa (Fullerton, 2014, 1). Video igre so vsestranski informacijski sistemi (Hamari in Keronen, 2017, 136), ki poleg zabave služijo tudi kot orodja za informiranje, oskrbo zdravja, medicine, usposabljanja, in učenja.

Igre kot učni pripomočki[[1]](#footnote-1) pozitivno vplivajo na analitične, prostorske, psihomotorične sposobnosti, kratkoročni in dolgoročni spomin, vizualno pozornost, identifikacijo, reševanje problemov in socialne spretnosti (Madani, Pierce, in Mirchi, 2017, 4). Igre imajo velik potencial pri podpori učenja konceptualnega razumevanja, procesov, praks, epistemologije in ne nazadnje obnašanje (Clark, Tanner-Smith, in Killingsworth, 2016, 79).

Igre so se izkazale kot avtentična orodja za ohranjanje in spodbujanje zdravja. Spodbujajo lahko kognitivne in praktične spretnosti profesionalcev. Igra *911 Paramedic* zajema 35 medicinskih primerov in 40 realističnih medicinskih orodij. Igralcu omogoča realistično izvajanje protokolov zdravljenja (Gareth Schott in Darrin Hodgetts, 2006, 311). Spodbujajo se simulacije z VR[[2]](#footnote-2) pripomočki z namenom praktičnega izobraževanja neizkušenih kirurgov pri katerih je ogrožanje zdravja pacientov med procesom izključeno (Ma in Zheng, 2011, 171). Obstajajo tehtni dokazi, da lahko igre posredujejo vplive, ki uporabnika za določena zdravstvena stanja naredijo bolj učinkovitega (Ma in Zheng, 2011, 311). Igralni pripomočki kot so *Dance Dance Revolution, Sony Eyetoy, Nintendo Wii* spodbujajo zdravje s ponujanjem alternativnih pristopov fizičnih aktivnosti. Igra *Rex Ronan* spodbuja zdrav življenjski slog s spodbujanjem proti kadilskih nagnjenj (Gareth Schott in Darrin Hodgetts, 2006, 311). Ulbin (2017) predstavi nevsiljiv način spremembe življenjskega sloga z izdelavo igre, ki bi pomagala otrokom s prekomerno telesno težo. Številne raziskave nakazujejo tudi na terapevtske potenciale. Igre pozitivno vplivajo na kognitivne dejavnosti, mobilnost, rehabilitacijo bolnikov, ki trpijo za izgubo ali oslabitev motoričnih funkcij (kap, parkinsonizem) (Salmon in drugi, 2017, 45). Dokaz sta rešitvi BREATHING+, ki je nastal na pobudo podjetja Zdrav dih in Bimeo, slovenskega podjetja Kinestica. Obe rešitvi temeljita na rehabilitaciji zdravja s pomočjo igranja iger (''Kinestica'', b.d.; ''Breathing Labs'', b.d.).

Prihodki industrije iger vztrajno rastejo. Samo v ZDA so leta 2012 dosegli 14,8 milijard dolarjev, kar je zasenčilo filmsko industrijo (Fullerton, 2014, XXV). Zadnje poročilo ocenjuje prihodek industrije iger na 30,4 milijarde dolarjev (''ESA'', b.d.). Na porast vpliva tudi vzpon mobilne tehnologije. V letu 2010 je bilo prodanih več mobilnih naprav kot osebnih računalnikov. V zadnjem četrtletju je bilo po svetu prodanih 100,9 milijonov telefonov (Rakestraw, Eunni, in Kasuganti, 2013, 2). S porastom prodaje teh naprav je sunkovito poskočilo povpraševanje po igrah. Leta 2011 je bilo vsak teden izdanih 15.000 novih aplikacij (prav tam). Na podlagi popularnosti mobilnih iger se vlagajo velike vsote v razvoj in promocijo, s čimer se je drastično dvignila tudi konkurenčnost na trgu (Soomro, Ahmad, in Sulaiman, 2013, 247).

Industrija iger je nedvomno velika industrija, ki ustvarja visokokvalitetna delovna mesta. Gamasutra navaja, da so razvijalci iger leta 2014 samo v Evropi, v povprečju zaslužili več kot 45,000 dolarjev (''Gamasutra'', b.d.). Povprečje za EU28, ki ga navaja Eurostat pa je 38,000 dolarjev (''Eurostat'', 2017). EY za leto 2017 navaja, da je v Evropi zaposlenih 108,000 ljudi v industriji iger, ki ustvarijo preko 16 milijard evrov dobička. (''EY'', b.d.). Tako postane toliko bolj mikavno za vlagatelje kot tudi za razvijalce, da usmerijo svoje moči v razvoj iger. Rast je privedla do tega, da je tehnologija postala priročnejša in dostopnejša. Vedno več posameznikov je motiviranih za izdelavo lastnih iger (Aleem, Capretz, in Ahmed, 2016, 55). Zaradi želje po hitrem vstopu na trg, vlagatelji pogosto pritiskajo na razvijalce. Posledično se zato podaljšujejo roki in slabo definirajo ocene zaključka projektov. (O’Hagan, Coleman, in O’Connor 2014, 182). Na izdajne roke vplivajo tudi slaba organizacija, upravljanje (Fábio Petrillo in drugi, 2009, 4) in izbira metodologije. Igre s slabo razvojno metodologijo bolj verjetno presežejo finančna sredstva in časovne termine (Kanode in Haddad, 2009, 556). Standish group je ocenil, da je v povprečju 30% projektov nedokončanih in 53% projektov, ki presežejo časovni termin s 189% presežkom sredstev (Petrillo in drugi, 2008, 707). Postalo je izredno pomembno, da se proces izdelave iger izboljša zavoljo konkurenčnosti in finančnih ciljev organizacij (Aleem, Capretz, in Ahmed, 2016a, 55).

Kompleksnost iger je eskalirala (Blow, 2004, 29) in z njo razvojni procesi (O’Hagan, Coleman, in O’Connor, 2014, 182). Organizacije zaradi neprilagojenih procesov pogosto razvijejo lastne procese, ki pa lahko vsebujejo zastarele tehnike in ne uporabljajo najboljših praks industrije razvoja (Sommerville 2010, 28, 30). Moramo poudariti, da ne obstaja en način razvoja iger (Aktaş in Orçun 2016, 249). Razlog je, da proces video iger ni podrobno definiran (McAllister in White, 2015, 14). Po naravi je proces razvoja iger nestrukturiran, intuitiven in organski proces, ki temelji na razvojnih praksah. To je bilo zadovoljivo v času, ko so bile razvojne ekipe in količine potrebnih virov za izgradnjo iger manjše (Aktaş in Orçun, 2016, 240). Danes je napoved razpona projekta skoraj nemogoča (Kanode in Haddad, 2009, 555). V nekaterih primerih lahko projekt vključuje tudi do tisoč ljudi in lahko traja več let (Kanode in Haddad, 2009). Težavnost se lahko le še stopnjuje, če so ekipe geografsko razdeljene (O’Hagan, Coleman, in O’Connor, 2014, 182). Igre so oblikovane s strani ekip izkušenih posameznikov, ki lahko vključujejo več visoko usposobljenih strokovnjakov z različnih področij, računalništva, umetnosti, medijskega oblikovanja, poslovanja (prav tam) in izobraževanja (Aslan in Balci, 2015, 307). Za načrtovanje in upravljanje tako kompleksnih multidisciplinarnih projektov je potrebna metodologija, kjer ad hoc načini upravljanja ne pridejo v poštev (prav tam). Čeprav so bile določene dobre prakse prevzete od tradicionalnega razvoja programske opreme se te razlikujejo od razvoja iger (O’Hagan, Coleman, in O’Connor, 2014, 182). V razvoju iger je močno vključeno ustvarjanje kreativnih vsebin, ki definirajo izgled igre (Aktaş in Orçun, 2016, 249). Vendar igre niso povsem le umetnost kot tudi ne produkt popolnega inženiringa. Razvoj igre je bolj podoben ustvarjanju izdelka s prepletanjem aspektov umetnosti, glasbe, programiranja, igranja in poslovnega upravljanja (Ramadan in Widyani, 2013, 95). Igre bolj stremijo k proizvajanju uporabniške izkušnje kot uporabnosti. (O’Hagan, Coleman, in O’Connor, 2014, 182). Poudarek je na evalvaciji uporabniške izkušnje in uporabo povratnih informacij, ki nato vodijo tok iteracij razvoja (O’Hagan, Coleman, in O’Connor, 2014, 183). Pressman zagovarja, da so igre programska oprema, ki ponujajo zabavo (Ramadan in Widyani, 2013, 95), vendar ni metode za določanje subjektivnega elementa zabave, na katero se osredotočajo oblikovalci iger. Zato je razvoj iger toliko bolj kompleksen v nasprotju s tradicionalnim razvojem programske opreme. Potrebno je razširiti podedovane tradicionalne tehnike razvoja programske opreme, da bi lahko podprli kreativni proces razvoja video iger (Fábio Petrillo in drugi, 2009, 19). Razvoj iger potrebuje specifične usmeritve, ki temeljijo na najboljših praksah in ocenjevalni model za reševanje izzivov, s katerimi se soočajo razvijalci pri izvajanju trenutnih procesov (Ramadan in Widyani, 2013, 95; Aleem, Capretz, in Ahmed, 2016, 55). V kratki zgodovini discipline, usmeritve in prakse še niso bile povsem raziskane (Aleem, Capretz, in Ahmed, 2016, 27). Posledično ni bilo izdelanega procesnega modela, ki bi temeljil na najboljših praksah razvoja iger. Takšen model, bi lahko zmanjšal čas razvoja, čas vstopa na trg ali celo izboljšal kvaliteto iger (O’Hagan in O’Connor, 2015, 15).

Prav tako do nedavnega še ni bilo zrelostnega modela, ki bi direktno naslovil težave ocenjevanja in izboljšanja procesov (Aleem, Capretz, in Ahmed, 2016, 58).

To je privedlo v motivacijo identifikacije usmeritev in praks ter predlaganje procesnega modela za igre, ki ga bomo v delu optimizirali s prvim[[3]](#footnote-3) primernim zrelostnim modelom za igre[[4]](#footnote-4). Delo je prikaz sistematičnega pristopa k izgradnji domenskega procesa, kateri kljub sistematizaciji in optimizaciji ohranja prostor za kreativnost.

2 METODOLOGIJA

Metodološki pregled pretekle literature je ključnega pomena vsakega akademskega raziskovanja. Področja kot so inženiring, so zaradi podcenjevanja pomembnosti metode pregleda literature kronično utrpela pomanjkanje raziskovalnih del, kar je otežilo teoretski in konceptualni napredek (Levy in J. Ellis, 2006, 181). Metoda nam omogoča pregled preteklih dognanj, ki služijo kot dobri temelji prihodnjim raziskavam (prav tam). Industrija video iger ponuja veliko literature z usmeritvami razvoja video iger vendar je na to temo malo akademske (Ruonala, 2016, 1). Razvoj iger je inherentno agilna aktivnost (prav tam). Večina organizacij je podedovalo agilne prakse (prav tam) za katere postoji veliko literature, vendar ta ni usmerjena v razvoj iger (Barbosa, 2017, 292). S pregledom literature želimo disciplini razvoja video iger prispevati trdnejše temelje nadaljnjim raziskavam na področju raziskovanja razvojnih procesov.

2.1 NAMEN RAZISKOVANJA

Namen raziskovanja je postavitev temeljev za sistematično razumevanje procesov in procesnih aktivnosti pri razvoju iger. Cilj naloge je sestaviti procesni model, ki bi bil primeren za uporabo in optimizacijo s prvim zrelostnim modelom za igre.

Na podlagi tega smo formulirali naslednji raziskovalni vprašanji:

**RV1:** Katere procesne modele, se uporablja pri razvoju video iger?

**RV2:** Kakšen bi na podlagi literature lahko bil procesni model, ki bi bil primeren za optimizacijo z zrelostnim modelom za igre?

2.2 ZBIRANJE PODATKOV

Kvaliteta literature vpliva na celostno napredovanje znanja tematike (Levy in J. Ellis, 2006, 183). Za zagotavljanje kvalitete iskanja literature smo uporabljali sistematične tehnike iskanja. Proces zbiranja podatkov smo razdelili na digitalne in analogne tehnike.

**Digitalne tehnike** vključujejo zbiranje literature iz najbolj stabilnih in priznanih podatkovnih skladišč, knjižnic in izjemoma spletnih strani, ki jih urejajo večje organizacije. Za iskanje literature smo izvajali poizvedbe na spletnih straneh in iskalnikih:

* google.si,
* amazon.com,
* link.springer.com,
* tandfonline.com,
* sciencedirect.com,
* onlinelibrary.wiley.com,
* uk.sagepub.com,
* Web of Knowledge,
* ACM Digital Library,
* IEEE Xplore.

Večina podatkovnih skladišč ponuja lastne grafične vmesnike s funkcijami omejevanja (Booth, Papaioannou, in Sutton, 2012, 77), ki omogočajo modifikacijo poizvedb. V kolikor je bilo to mogoče smo se posluževali teh. V veliko primerih smo z namenom strnitve zadetkov bili primorani uporabiti naprednejše tehnike iskanja s prostim tekstom. Elektronske baze tipično dovoljujejo iskanje s prosim tekstom. Pristop, ki se velikokrat izkaže za pomanjkljivega smo podprli z uporabo krajšav ('\*','$', '%') in nadomestnih ('?', '!') simbolov. Poleg iskanja s prostim tekstom smo uporabili iskanje s tezavri[[5]](#footnote-5), ki služijo podatkovni bazam za naslavljanje predmetov ali opisov (Booth, Papaioannou, in Sutton, 2012, 74). Naprednejše tehnike je vključevala tudi uporaba logičnih operatorjev (AND, OR, NOT):

* OR združuje termine z istim konceptom in s tem razširi iskanje,
* AND združuje termine z različnimi koncepti in s tem strni iskanje,
* NOT izključuje nepomembne termine in strni iskanje (Booth, Papaioannou, in Sutton 2012, 76).

**Analogne tehnike** so vključevale iskanje avtorjev in metodo snežene kepe. Z iskanjem po bibliografijah smo spoznavali avtorje in konstrukte, ki so najbolj povezani s tematiko raziskave. Sledili smo verigam citatov in odkrivali zgradbo literatur. To nam je pomagalo pri spoznavanju vplivnih literatur, avtorjev in spoznanju terminologije tematike.

Kot vplivne literature smo identificirali dela:

1. Ruonala, Henna-Riikka. 2016. Agile Game Development: A Systematic Literature Review.
2. Aleem, Saiqa, Luiz Fernando Capretz, in Faheem Ahmed. 2016. A Digital Game Maturity Model (DGMM).

Prvo delo je sistematični pregled literature, ki vsebuje analize člankov in njihove povezave. Slednje delo je prvi primer zrelostnega pristopa za igre, ki smo ga uporabili tudi v našem, delu. Iz terminologije smo razbrali gesla, ki so nam pomagala pri kreiranju poizvedb. Gesla sledijo po pomembnosti:

* Game,
* Process,
* life-cycle,
* method,
* engineering,
* development.

Iz teh smo sestavljali kombinacije, ki veljajo za indekse. Sledita primera:

* Game Development Process OR Life-Cycle.
* Game Process OR Method OR Life-Cycle.

2.3 POTEK DELA

Uvodoma smo predstavili potencial, ki ga predstavljajo video igre. Iz literature je razvidno, da je bilo na temo razvoja procesov za video igre opravljenih malo raziskav. Ker spada razvoj video iger med razvoj programske opreme (Bates, 2004, 217) bomo v tretjem poglavju poleg predlaganih procesnih modelov za igre predstavili tudi vse standardne procesne modele za razvoj programske opreme. V četrtem poglavju bomo identificirali procesne modele, ki se uporabljajo v praksi in prikazali njihove pomanjkljivosti. Nato bomo v nadaljevanju sestavili predlog ustreznejšega procesnega modela za razvoj video iger. Definiranje predloga modela bo potekala sistematično:

* pog. 5: izgradnja metode za definiranje procesnega modela,
* pog. 6: apliciranje metode in predlaganje procesnega modela na podlagi literature,
* pog. 7: spoznanje pristopov optimizacije in poskus izvedbe.

Raziskava je bila pisana v urejevalniku besedil Word. Zbrano literaturo smo upravljali z odprtokodno programsko opremo Zootero. Vsi procesni modeli so bili narisani z odprtokodno rešitvijo ArgoUML. Za znižanje tveganj smo uporabljali spletno rešitev spremljanja izvorne kode GitHub (https://github.com/andrazg/game\_development\_process), ki smo jo uporabili za shranjevanje dokumentacije raziskave.

2.4 OMEJITVE

Upoštevati moramo, da smo pri iskanju literature v večini uporabljali digitalne tehnike. Ker je raziskovalno področje razvojnih procesov slabo raziskano je velika možnost nekonsistentne terminologije ali nepravilne rabe. V literaturah se prepletajo trije termini, ki definirajo procese. Čeprav vsak termin posamezno definira drugačen obseg lastnosti in namenov (pog. 3) so jih avtorji enačili. Ti termini si sledijo po pojavnostih: proces (process), življenjski cikel (life-cycle) in metoda (method). Poleg tega smo opazili nekonsistentnost pri poimenovanju procesnih modelov. Navedeno je nekaj ekvivalenčnih izrazov procesnih modelov (življenjskih ciklov, metod):

1. Code and fix (Tsui, Karam, in Bernal, 2013, 58) **~** Build and fix (Sabharwal, 2009, 8).
2. Stage gate (Peters 2008, 113) **~** Phased-release (Lethbridge in Laganiere, 2005, 430).
3. Incremental Software Development Life Cycle (Sabharwal, 2009, 16) **~** Incremental delivery (Sommerville, 2010, 47) **~** Incremental Process (Pressman in Maxim, 2014, 43)

Nevarnost rezultatov poizvedb predstavljajo tudi ključne besede, ki s svojim pomenom ne opisujejo vsebine literature. Tukaj smo si pomagali s sopomenkami (inženiring, razvoj). Za prevajanje strokovnih izrazov v slovenščino smo uporabljali Računalniški slovarček[[6]](#footnote-6). Besed, ki jih nismo našli smo prevedli sami in podprli z opombo v izvorni obliki.

3 PROCESNI MODELI

Razvoji video iger so v prvi vrsti projekti razvoja programske opreme (Bates, 2004, 217). Zato se za upravljanje razvoja priporočajo tehnike, ki jih definira disciplina programskega inženiringa (prav tam). Programski inženiring je uporaba discipliniranega, merljivega in sistematičnega pristopa v razvoj, zagon in vzdrževanje programske opreme. Temelj programskega inženiringa predstavlja proces (Pressman in Maxim, 2014, 15). Proces je zbirka aktivnosti, del in nalog, ki se izvajajo ob ustvarjanju produkta (Pressman in Maxim, 2014, 16) in lahko vsebujejo tudi sebi podrejene procese (Sommerville, 2010, 28). Predstavlja primarni pristop, ki projekt organizira v aktivnosti (Lethbridge in Laganiere, 2005, 428). Te aktivnosti bivajo v okvirjih ali modelih, ki definirajo njihove povezave s procesom in drugimi aktivnostmi (Pressman in Maxim, 2014, 31). Procesi iste narave so razdeljeni v procesne modele (Rolland, 1998, 8), ki so poenostavljena predstavitev procesa razvoja (Sommerville, 2010, 29). Zaradi svoje uporabnosti, narave ali drugih lastnosti so procesni modeli razdeljeni v kategorije, ki sledijo.

3.1 AD HOC PROCESNI MODELI

Ti modeli so povsem odvisni od znanj in izkušenj razvijalca ali razvojne ekipe. Inštitut programskega inženiringa je tak način razvoja označil za nepredvidljivega. Vsakršna izguba člana razvoja ima negativen učinek na proces razvoja (Muffatto, 2006, 73). Med ad hoc modele spadajo Slam dunk, Big bang in priložnostni model (Peters, 2008, 109; Jalote in drugi, 2004, 2; Lethbridge in Laganiere, 2005, 428). Ker se pri slednjem kodiranje smatra za osrednjo aktivnost tega procesa ga v praksi imenujejo tudi (code-and-fix) kodiraj in popravi (Tsui, Karam, in Bernal, 2013, 58) ali (build and fix) gradi in popravi (Sabharwal, 2009, 8).

3.2 SEKVENČNI PROCESNI MODELI

Procesni modeli iz te družine stremijo k vzpostavitvi strukture in ureditve razvoja programske opreme (Pressman in Maxim, 2014, 41). Aktivnosti in naloge se izvajajo sekvenčno z definiranimi časovnimi mejniki (prav tam). Spadajo tudi med predpisujoče, ker predpisujejo niz procesnih elementov; okvirnih aktivnosti, dejanj programskega inženiringa, nalog, zagotavljanja kvalitete in mehanizme sprememb (prav tam). Vsak model predpisuje določen delovni tok[[7]](#footnote-7) , ki odraža medsebojno povezanost procesnih elementov (prav tam). Med sekvenčne spadajo kaskadni, v-model, fazni in inkrementalni procesni model.

*Kaskadni model* predstavlja klasični življenjski cikel s sistematičnim pristopom (Pressman in Maxim, 2014, 42). Model je primer načrtno-usmerjenih procesov, ker je v praksi vsako aktivnost potrebno načrtovati. (Sommerville, 2010, 31). *V-model* predstavlja variacijo kaskadnega modela (Pressman in Maxim, 2014, 43). Imenovan tudi kot variacijski in validacijski model (''Tutorialspoint'', b.d.). Verifikacija in validacija (V&V) sta namenjeni za potrditev ustreznosti programske opreme na podlagi specifikacije in pričakovanj naročnika. (Sommerville, 2010, 41). Ta model prikazuje odnose med aktivnostmi zagotavljanja kvalitete na eni strani in komunikacije na drugi. *Fazni model* (phase-release model) izvira iz proizvodnega sektorja. Ima veliko poimenovanj, eden izmed njih je tudi stopenjski (stage gate) (Peters, 2008, 115). Model izboljša nekaj težav kaskadnega modela in predstavlja koncept inkrementalnega razvoja. Predlaga delitev projekta na posamezne faze po definiranju zahtev in načrtovanju. Vsako fazo se ob končanju sprosti stranki. Tako so lahko nekateri deli sistema vidni prej kot bi bili z uporabo kaskadnega modela (Lethbridge in Laganiere, 2005, 430). *Inkrementalni model* združuje elemente linearnega in paralelnega procesnega toka. V časovnem toku se odvijajo linearne sekvence na različnih stopnjah. Vsaka sekvenca predstavlja del sistema programa v razvoju. Posamezno sekvenco imenuje inkrement (Pressman in Maxim, 2014, 44). Sekvenca korakov predstavlja faze procesa. Ta pristop je v osnovi sestavljen iz prepleta specifikacije, razvoja in validacije, ki sestavljajo sosledje različic projekta (Sommerville, 2010, 30).

3.3 EVOLUCIJSKI PROCESNI MODELI

Med razvojem se zahteve pogosto spreminjajo kar onemogoča linearnost procesov. Tesni razvojni cikli onemogočajo dokončanje zapletenih programskih paketov. Zavoljo poslovnih in konkurenčnih pritiskov pa mora razvoj vseeno izdati omejeno različico paketa. Kadar imamo produkt, ki se konstantno razvija in spreminja, ko so temeljne zahteve programske opreme dobro poznane, razširitve slednjih pa še v načrtovanju so najprimernejši evolucijski procesni modeli. Ti modeli so ponavljajoči, iterativni. Sem spadajo prototipiranje, spiralni in vzporedni procesni model (Pressman in Maxim, 2014, 45).

*Prototipiranje* poteka po iteracijah. Načrt in izvedba prve iteracije sta hitra. Investitor po iteraciji prototip pregleda in poda povratno informacijo. Takšen potek omogoča natančnejše definiranje zahtev. Iteracija se zaključi ob rektifikaciji protitipa in izpolnitvi želja investitorjev, sočasno pa pripomore k boljšemu razumevanju prihodnjih. Navadno ta model služi kot idealen mehanizem za prepoznavo programskih zahtev (Pressman in Maxim, 2014, 45). *Spiralni model* združuje iterativno naravo prototipiranja in kontroliran sistematični pogled kaskadnega modela. Omogoča hiter razvoj z izdajanjem vse bolj popolne različice programa. Zgodnje izdaje predstavljajo modele ali prototipe kasnejše iteracije pa vedno volj popolne različice (Pressman in Maxim, 2014, 48). *Vzporedni model* omogoča uporabo iterativnih in vzporednih elementov v kombinaciji z vsemi spoznanimi modeli v tem poglavju. Vse razvojne aktivnosti se odvijajo vzporedno, vendar je lahko vsaka aktivnost na svoji stopnji razvoja. (Pressman in Maxim, 2014, 50). Model prestavlja abstrakcijo dejanskega procesnega cikla. V njem se definira serijo sekvenčnih dogodkov vseh aktivnosti, dejanj in opravil. (prav tam). Vsaka aktivnost, dejanje ali opravilo se odvijajo vzporedno z drugo aktivnostjo, dejanjem in opravilom (Pressman in Maxim, 2014, 51).

3.4 SPECIALIZIRANI PROCESNI MODELI

Ti modeli povzemajo karakteristike enega ali več tradicionalnih modelov. Lahko jih definiramo tudi kot skupek posamičnih tehnik ali metodologij za doseganje specifičnih ciljev razvoja (Pressman in Maxim, 2014, 52). Sem spadajo komponentni, formalni, unificiran, sinhronizacijsko in stabilizacijski model in rapidni razvoj.

*Komponentni model* nastavlja proces razvoja programske opreme z že prevedeno programsko opremo (Pressman in Maxim, 2014, 53). Komercialne komponente (COTS[[8]](#footnote-8)) zagotavljajo namenske funkcionalnosti z dobro definiranimi vmesniki, ki omogočajo integracijo v programsko opremo v razvoju (prav tam). Po naravi je model evolucijski, ki zahteva iterativni pristop k razvoju (prav tam). *Formalni model* zajema zbirko aktivnosti, ki vodijo v formalno, matematično specifikacijo programske opreme. Ta model omogoča specifikacijo, razvoj in verifikacijo sistemov z apliciranjem strogih matematičnih notacij (prav tam). Primer formalnega razvojnega procesa je model čiste sobe[[9]](#footnote-9). Vsak inkrement v razvoju ima formalno specifikacijo na podlagi katere se izvede implementacija (Sommerville, 2010, 32). *Unificiran model* je poskus združitve najboljših značilnosti procesnih modelov z umestitvijo njihovih najboljših praks v agilne procese. Je primer modernega procesnega modela, ki je bil izpeljan iz unificiranega jezika modeliranja[[10]](#footnote-10) in povezanega unificiranega razvojnega procesa programske opreme (Sommerville, 2010, 50). *Sinhronizacijsko stabilizacijski model* je podoben inkrementalnemu. Začne se z zahtevami. Nato razvojna ekipa izdela specifikacijo, določi prioritete in razdeli razvoj na štiri večje izdaje. Izdaje[[11]](#footnote-11) so kandidat za javnost (RC) 1 do 3 in izdaja v proizvodnjo. Kandidat za javnost je prvi večji izid in predstavlja mejnik v razvojnem ciklu z možnostjo širše modifikacije kode in vsebine (Peters, 2008, 120). S časom vsak naslednji izid postane izhodišče, odstotek dovoljenih sprememb v kodi, število novih funkcionalnosti in napake pa se zmanjšujejo. Največji ali celo najmanjši defekti so z izidom RC3 odpravljeni (Peters, 2008, 121). *Rapidni razvoj* predvideva fragmentacijo programskih zahtev v posamezne module. Ti se nato razvijajo in neodvisno integrirajo v produkt. Najpomembnejši atribut modela je hitrost poteka razvoja od analize zahtev do končnega sistema. Čas izdaje enega modula navadno traja 60 do 90 dni in ga imenujemo časovni okvir (Sabharwal, 2009, 19). Modularna fragmentacija omogoča uporabo komponent na podlagi katerih se skrajša časovni okvir razvojnega cikla in stroške razvoja (Sabharwal, 2009, 20).

3.5 AGILNI PROCESNI MODELI

Agilni procesi so družina razvojnih metodologij, ki proizvajajo programsko opremo s kratkimi iteracijami in dovoljujejo večje spremembe v načrtovanju (Tsui, Karam, in Bernal, 2013, 84). Agilni razvoj je pravzaprav nabor najboljših praks zbranih iz drugih življenjskih ciklov in uspešnih praks kodiranja (Peters, 2008, 116). Veliko jih izhaja iz leta izkušenj, ki sta jih prinesla uporaba iterativnih in inkrementalnih procesov. Ne obstaja končna definicija kaj sestavlja Agilno metodo, vendar obstaja kar nekaj karakteristik, ki so metodam sorodne (Tsui, Karam, in Bernal, 2013, 84). Čeprav agilni modeli slonijo na inkrementalnem razvoju in izdaji, predlagajo drugačne procese za doseganje tega. Ne glede na raznolikost v procesih si delijo načela predstavljena v agilnem manifestu. (Sommerville, 2010, 59). Tabela 3.1 spodaj jih podrobno opisuje.

Tabela 3.1 podrobno opisana načela agilnih modelov  
  
Vir: (Sommerville, 2010, 60)

Med agilne modele spadajo XP[[12]](#footnote-12), Scrum, Crystal, Lean, kanban, DSDM[[13]](#footnote-13), FDD[[14]](#footnote-14), ASD[[15]](#footnote-15), TDD[[16]](#footnote-16), DAD[[17]](#footnote-17), odprtokodni model in Scrumban.

*Ekstremno programiranje*, uporablja paradigmo objektno-orientiranega razvoja (Pressman in Maxim, 2014, 72). Posebnost modela je izdelava zgodb (Pressman in Maxim, 2014, 73) ali scenarijev na podlagi zbiranja zahtev (Sommerville, 2010, 65). Te opisujejo potrebne funkcionalnosti in lastnosti programske opreme (Pressman in Maxim, 2014, 73). *Scrum model* je agilna metoda, ki se fokusira na upravljanje inkrementalnega razvoja. Inovativna lastnost modela so sprint cikli (Sommerville, 2010, 73). Sprint je enota načrtovanja, kjer se oceni potrebno delo, izberejo funkcionalnosti za razvoj in opravijo implementacije programske opreme (Sommerville, 2010, 73). *Kristalno družino modelov* je razvil Alistair Cockburn z namenon prilagoditve metodologij projektom. Primeri modelov so: Crystal Clear, Crystal Orange, Crystal Orange Web (Cockburn, 2006, 94). Metodologije so definirane z barvami. Temnejša kot je barva primernejša je metodologija za zahtevnejše projekte. *Lean ali vitka* metoda je bila ustvarjena za potrebe avtomobilske industrije. Povzema sedem principov; znižanje odpadkov, kvaliteto produkta, ustvarjanje znanja, proizvodnjo v pravem trenutku[[18]](#footnote-18), spoštovanje ljudi in konstantno optimizacijo. Medtem, ko se ostale agilne metode nagibajo k agilnosti se ta metoda k efektivnosti (Janes in Succi, 2014, 144). *Kanban* metoda se izvaja z uporabo kanban sistema, ki omejuje delo v teku z uporabo vizualnih signalov (Anderson in Carmichael, 2016, 1). Vizualizacija se prične s Kanban tablo, ki mora imeti vidno označene in definirane točke zaveze in jasno opredeljene meje dela v teku. Zaveza je eksplicitni dogovor med naročnikom in razvojem, ki narekuje razvoju delo na tistih postavkah katere želi naročnik. Zahteve ali postavke se nahajajo v bazenu idej, ki so lahko izbrane ali pa tudi ne. Čas od izbrane postavke do izdaje definira naročnikov dobavni rok. Dobavni rok razvoja pa je čas od izbranih postavk do njihove dostave naročniku. Ta čas definira vse postavke, ki so v danem trenutku obravnavane in ga označuje kot delo v teku[[19]](#footnote-19). *DSDM* je metoda za razvoj sistemov z zahtevnimi časovnimi omejitvami s pomočjo inkrementalnega prototipiranja v kontroliranem okolju. Filozofija modela sloni a modificiranem Paretovem principu, da je mogoče narediti 80% aplikacije v 20% časa, ki bi bil potreben za izdajo celotne aplikacije (Pressman in Maxim, 2014, 80). *Funkcionalno usmerjen razvoj* je zgrajen okoli jedra najboljših praks. Zbrane zahteve naročnika predstavljajo razredi s svojimi metodami (funkcijami), ki skupaj tvorijo funkcionalnosti. Funkcionalnosti se ovrednotijo glede na uporabno vrednost naročnika. Ovrednotene služijo kot vodilo in sledenje napredku razvoja (Palmer in Felsing, 2002, 39). *Prilagodljiv razvojni model* se fokusira na razvoj velikih in kompleksnih sistemov. Metoda spodbuja uporabo inkrementalnih in iterativnih tehnik s konstantnim prototipiranjem (Abrahamsson in drugi, 2017, 72). *Testno usmerjen razvoj* je evolucijski proces pri katerem se izdela testni protokol in test pred produkcijsko kodo (Duka in Hribar, 2010, 1). Razlog tega je, da testni scenariji vodijo načrtovanje, saj ti določajo potrebe (Duka in Hribar, 2010, 2). *Disciplinirana agilna dostava* je metoda, katere osrednji del je izbira dostavnih (izdajnih) pristopov na podlagi problematike. DAD pozna štiri različice razvojnega dela procesa. Prva je agilna/bazična različica, ki razširja cikel izgradnje po Scrum metodi. Druga je naprednejša/Lean, ki ima za osnovo Kanban. Tretja je neskončni cikel izdajanja in zadnja četrta je raziskovalna, ki ima za osnovo Lean Start-up[[20]](#footnote-20) pristop. *Odprtokodni model* brezplačno distribuira programsko opremo z izvorno kodo. Model pričakuje, da bodo člani skupnosti zato imeli razlog za izboljšanje programa. Zagotavljanje kakovosti izvaja skupnost sama (Lethbridge in Laganiere, 2005, 434). Različice se izdajajo na podlagi specifikacij pridobljenih preko e-pošte, oglasnih desk in drugih neformalnih medijev. Integracije se dogajajo pogosto, preko spleta. Razvijalci imajo skupne vizije, saj opremo potrebujejo in jo razvijajo za lastne potrebe (Tsui, Karam, in Bernal, 2013, 98). Scrumban je hibrid Scrum in Kanban agilnih modelov namenjen obvladovanju dinamičnih sprememb naročniških zahtev in frekvenčnih težav z izvorno kodo (Yilmaz in O’Connor, 2016, 238). Od metode Scrum je povzel tehnike dnevnih sestankov, definiranja zgodb in samo organizacijski pogled na razpored dela (prav tam). Zaradi boljše organizacije dela, preglednosti napredka in sprememb se za koordinacijski mehanizem uporablja Kanban tehniko WiP (prav tam). Ta omogoča koordinacijo vlečenja dela v nasprotju z Scrum, ki prakticira potiskanje dela (prav tam). Zaradi potiskanja dela v praksi velikokrat prihaja do zastojev zaradi večopravilnosti razvoja. Tehnika vlečenja pomeni, da je toliko enot dela v teku kolikor jih lahko razvoj zaključi (prav tam). Optimizacija dela v teku je odvisna od prave izbire nalog za dosego optimalne pretočnosti. WiP pomaga razvijalcem omejiti večopravilnost z namenom povečanja produktivnosti (Yilmaz in O’Connor, 2016, 238).

3.6 PROCESNI MODELI ZA IGRE

Procesni modeli za igre so predlogi modelov, ki poskušajo zapolniti vrzeli potreb razvoja video iger, ki ostanejo z apliciranjem standardnih procesnih modelov.

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[[21]](#footnote-21). 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[[22]](#footnote-22) 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[[23]](#footnote-23). PRILOGA B prikazuje življenjski cikel DEG[[24]](#footnote-24), 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 delovnih produktov (Aslan in Balci, 2015, 310). Za 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, evalvacijae 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[[25]](#footnote-25) (PRILOGA I, pod Scrum). 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 analize[[26]](#footnote-26) razvoja, ki služi za identifikacijo pomanjkljivosti v preteklem razvojem procesu (Barbosa, 2017, 294).

4 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 drugi, 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 med katerimi so vidnejši: XP, Scrum, Kanban, rapidni, inkrementalni in komponentni razvojni modeli (O’Hagan, Coleman, in O’Connor, 2014, 187). 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 iger. Raziskava je zajemala najmanjša podjetja kot tudi enega največjih (Rovio Entertainment). Vsa podjetja razen enega so v vsaj enem koraku procesa uporabili agilne metode (Koutonen in Leppänen, 2013, 12). Prav slednja avtorja povzemata tudi delo avtorjev Tran in Biddle (2008)[[27]](#footnote-27), ki označujeta Scrum, XP in Kanban za najpogosteje uporabljene modele (Koutonen in Leppänen, 2013, 4).

Stacey in Nandhakumar (2008) sta naredila študijo treh razvojnih ekip, ki je pokazala, da niso aplicirale agilnih metod kot takih ampak so se posluževale agilnim praksam.

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

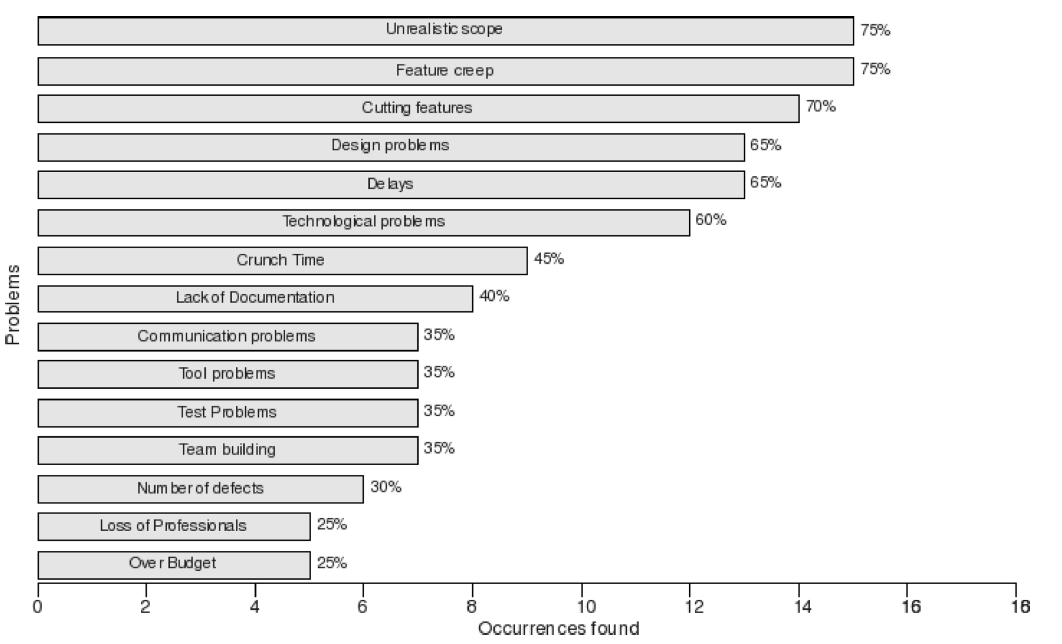
Prav tako agilne pristope podpira Fullerton, saj meni, da je Scrum primeren za reševanje zapletenih problemov oblikovanja iger (Fullerton, 2014, 369). Svoje mnenje podaja John Comes[[28]](#footnote-28), ki meni, da naj opustimo kaskadne (sekvenčne) procese in sprejmemo agilne (John Comes v Novak, 2012, 366).

4.1 POMANKLJIVOSTI AGILNIH PRISTOPOV

Agilni pristopi niso usmerjeni v izdelavo dokumentacije ampak na zgodaj delujočo verzijo igre (O’Hagan in O’Connor, 2015, 5). Slabo definirana specifikacija vodi v defekte, vpliva na funkcionalnost, vire in časovnice (Kanode in Haddad, 2009, 263). Med produkcijo ti pristopi nadaljujejo s prototipiranjem in izdelavo inkrementov igre. Kar privede do sprememb, ki drastično vplivajo tudi na GDD (Kanode in Haddad, 2009, 263). Posledično se spremeni okvir projekta kar negativno vpliva na produkcijo (Kanode in Haddad, 2009, 263). Poleg tega so bili agilni modeli namenjeni za razvoj programske opreme in ne iger (O’Hagan in O’Connor, 2015, 6). Več kot polovica študij primerov omenja kreativnost, kot glavni izziv uporabe agilnih pristopov (Ruonala, 2016, 29). Multidisciplinarne ekipe se pogosto delijo med umetnike in programerje, kar privede do težav v komunikaciji (Kanode in Haddad, 2009, 263). Umetniki imajo večji odpor po sprejemanju agilnih metod in ostalih uveljavljenih procesnih kontrol kot tehnično orientirani zaposleni (Ruonala, 2016, 30).

Brazilska raziskava je pokazala, da agilne prakse tudi niso najučinkovitejše. V raziskavi je sodelovalo 58 podjetij, ki razvijajo video igre. Anketiranci so umestili svoj procesni model v enega od štirih kategorij in ocenili njegovo uspešnost glede na pogoste težave, ki se pojavljajo pri razvoju video iger. Te so odkrili Petrillo in kolegi (2009) v svoji raziskavi pri kateri so analizirali dokumente analiz razvoja (tab. 4.1).

Tabela 4.1 najpogostejše težave, ki se pojavljajo v razvoju iger na podlagi dokumentov analiz razvoja



Vir: (Petrillo in drugi, 2009, 18)

Kategorije procesov so bile definirane glede na naravo procesov in si sledijo: agilni, predpisujoči, ad-hoc in brez procesa (Politowski in drugi, 2016, 155). Agilne predstavljajo XP, Kanban, Scrum in veljajo za iterativne pristope konstantnega posodabljanja procesa. Predpisujoči so derivati kaskadnih, sekvenčnih procesnih modelov (Politowski in drugi, 2016, 155, 156). Med ad-hoc spadajo modeli, ki so visoko modificirani in ne spadajo med agilne in ne med predpisujoče (prav tam). Zadnja kategorija brez procesa definirala pristope brez metodološke podlage (prav tam). Pogoste težave so bile predstavljene na podlagi raziskave, ki so jo naredili Petrillo in kolegi o pogostih težavah med razvojem igre (prav tam). Rezultati so pokazali, da so zakasnitve izdajnih rokov, nerealističen okvir, pomanjkanje dokumentacije najpogostejši problemi (Politowski in drugi, 2016, 160). Poleg tega so ugotovili, da uporaba sistematičnih pristopov pripomore k boljšim produktom ne glede na tip pristopa (prav tam) pri čemer so se visoko modificirani procesni modeli izkazali z največjo stopnjo uspešnosti projektov (Politowski in drugi, 2016, 158).

4.2 POMANKLJIVOSTI MODELOV ZA IGRE

Kljub temu, da so v poglavju 3.6 predstavljeni predlogi modelov, ki so eksplicitno sestavljeni za potrebe razvoja video iger jih spremljajo številne pomanjkljivosti. 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 se lansirajo 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 definiranje 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. Kar vodi v to, da celotna produkcija sloni na enem dokumentu. V PRILOGA E pa je prikaz več artefaktov dokumentacije, 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 predlog igre, GDD, TDD[[29]](#footnote-29), vodnik umetniškega sloga[[30]](#footnote-30) (UB), projektni načrt (Novak, 2012, 380-394). Vsi so predstavljeni kot samostojni dokumenti, medtem, ko nekatere ekipe razdelijo celo GDD na 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 E) kot pri definiranju faz razvoja (PRILOGA F), ki je 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[[31]](#footnote-31), ki pomeni sprejemanje nemogočih zahtev (Rucker, 2002, 27) in je nasprotje iterativnemu razvoju (Novak, 2012, 367).

Neuradne različice modelov se pojavijo tudi v neznanstveni literaturi. Zaradi pomanjkanja dokumentacije jih nismo mogli predstaviti v poglavju 3.6. Prav tako pa posedujejo pomanjljivosti. GUP[[32]](#footnote-32) 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, 2012, 22). Flood pri definiranju GUP predstavi in komentira GWP[[33]](#footnote-33) (''Gamedev'', 2003). Kljub številnim pomanjkljivostim ta proces prvič predstavi artefakte, ki so specifični za igre (Wilson Brotto Furtado, 2012, 21) (specifikacija igre, umetniška biblija, tehnična specifikacija) (''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. Pri tem procesu 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 (''EcuRed'', 2018). Razvoj iger ni linearen proces (Flynt in Salem, 2004, 151). V praksi je tendenca po izogibanju uporabi 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, na kar najverjetneje vpliva tudi tajna narava industrije iger (Wilson Brotto Furtado, 2012, 22). Med neuspešne spada tudi model, ki ga objavi Demachy (2003) na Gamasutra. Interpretira XP metodologijo in predlaga XGD[[34]](#footnote-34) za video igre. Vendar pristop ne zagotavlja strukturiranih usmeritev, deloma zato, ker gre za agilno metodologijo (Wilson Brotto Furtado, 2012, 22).

Poleg naštetih se v akademski literaturi pojavita tudi dve pobudi procesov AGP[[35]](#footnote-35) 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, 2012, 22). Slednji pa je neustrezen, ker 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 (Fabio Petrillo in Pimenta, 2010, 10). Zato kakršnikoli predpisujoči (sekvenčni) pristopi niso priporočljivi.

4.3 UGOTOVITVE IN USMERITVE MODELIRANJU

V času pred-produkcije se priporoča prototipiranje, saj služi za lažje definiranje igre (Kanode in Haddad, 2009, 261). Definiranje igre je mišljeno odkrivanje zabave, ki je osrednja zahteva pri razvoju iger za katero pa ni merila. Zabavo je potrebno validirati na vsakem koraku razvojnega procesa, kar vodi v apliciranje visoko iterativnega upravljanja procesa (Koutonen in Leppänen, 2013, 3). To potrjuje dejstvo, da razvoj iger sloni na iteracijah (Novak, 2012, 366). Manjuka in kolegi (2016) potrjujejo, da agilne prakse dobro delujejo v fazi pred-produkcije zaradi hitrih iteracij in prototipiranja (Manjuka, Chakradhar Raju, in Sai Chand, 2016, 558). Kanode in Haddad (2009) poleg tega predlagata uporabo enostavnih agilnih metod v fazi pred-produkcije, ki omogočajo eksploracijo igralnosti in uporabniških interakcij.

Za izboljšanje procesa se priporoča združevanje razvojnih procesov (Manjuka, Chakradhar Raju, in Sai Chand, 2016, 558). Podobno meni tudi Keith (2010), ki navaja, da je Scrum primeren za pred-produkcijo medtem, ko priporoča Lean in Kanban pristope v produkcijski fazi (Keith, 2010, 298). V produkciji je mogoče uporabiti tudi bolj formalni proces, ker je bila večina eksperimentiranja v pred-produkcijski fazi že zaključenega (Kanode in Haddad, 2009, 264).

Zanimivost razvoja iger je tudi v življenjskem ciklu razvoja. Življenjski cikel se nikoli ne konča. Virtualni svet se po koncu produkcije posodablja (Bartle, 2003, 93). V tem času se izdajajo 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 programske opreme in pripomorejo k zagotavljanju kvalitete v zameno za brezplačno igranje (Levy in Novak, 2009, 54). Prav tako z modificiranjem[[36]](#footnote-36) igre s čimer se vključijo v primitiven razvojni proces (Cooper in Scacchi, 2015, 5).

Glavna razlika med razvojem programske opreme in razvojem igre je v razširjeni uporabi in integraciji multimedijskih sredstev[[37]](#footnote-37) (Kanode in Haddad, 2009, 261). Kanode in Haddad (2009) jasno navajata, da razvoj video iger potrebuje inženirsko prakso, ki bo znala upoštevati specifične karakteristike upravljanja multimedijskih sredstev in privlačnega igranja (Kanode in Haddad, 2009, 261). Z razliko od drugih domen razvoja aplikacij, sredstva pri razvoju iger predstavljajo izziv. Izdelava efektivnega cevovoda za upravljanje sredstev je esencialnega pomena in lahko predstavlja samostojen projekt (Kanode in Haddad, 2009, 261). Potek dela umetnikov in programerjev je različen in potrebuje natančno sinhronizacijo (Ruonala, 2016, 30). Zato Manjuka in kolegi (2016) priporočajo kaskadni model za izdelavo sredstev, ker so za njih zahteve jasno definirane, kar pa pokrije samo upravljanje kreativnega procesa in ne sinhronizacijo z inženiringom.

Po pregledu literature lahko ugotovimo, da ne obstaja procesni model, ki bi upošteval multidisicplinarno sestavo razvojnih ekip in ni modela, ki bi definiral sinhronizacijo programskega inženiringa in izdelavo kreativnih sredstev. Prav tako ne obstaja definicija visokomodificiranega procesnega modela, ki bi temeljil na najboljših praksah in literaturi. Zato bomo v nadaljevanju izbrali metodo in prikazali sistematičen primer izgradnje procesnega modela, ki bi najbolje ustrezal domeni razvoja video iger.

5 IZBIRA METOD

Razvidno je, da se v praksi uporabljajo agilni pristopi in njim podobnim hibridi, ki slonijo na iteracijah. Iterativen in inkrementalen razvoj predstavljata temeljne principe agilnega modeliranja, ki je metoda za izgradnjo agilnih procesnih modelov (Ambler, 2002, 44). Spoznanja vodijo v prepričanje, da bi bila izbira metode agilnega modeliranja (AM) za izgradnjo procesnega modela upravičena. Vendar sama uporaba AM ne bi zadostovala za izgradnjo sistematičnega procesa. Poleg tega se priporoča formalizacija procesa (Rucker, 2002, 32). To pomeni, da mora biti proces dokumentiran in frekvenčno revidiran (Rucker, 2002, 32). AM pa je bolj produkcijska filozofija kot trdna zbirka pravil (Unger in Novak, 2011, 181). Predstavlja skupek najboljših praks, ki slonjo na principih in vrednotah predstavljenih v agilnem manifestu. AM ne predpisuje postopkov za izgradnjo določenega procesa ampak spodbuja konstantno in efektivno modeliranje skozi nasvete in dobre prakse (Ambler, 2002, 8). Zato bomo uporabili dodaten pristop k modeliranju procesa z namenom podpore dokumentaciji in sistematizaciji procesa. Obetaven pristop v smeri sistematičnega in strukturiranega razvoja programske opreme je uporaba tehnik meta-modeliranja metode MetaME za izgradnjo procesnih modelov (Engels in Sauer, 2010, 419).

5.1 AGILNO MODELIRANJE

Bistvo AM leži v njegovih praksah. Prakse AM so organizirane v štiri kategorije: iterativno in inkrementalno modeliranje, skupinsko delo, enostavnost in validacija (Ambler, 2002, 44–45).

Iterativno in inkrementalno kategorijo definirajo prakse:

* apliciranja pravih artefaktov,
* izdelave več vzporednih modelov,
* iteracije do naslednjega artefakta,
* izvajanja manjših inkrementov (Ambler, 2002, 45).

Artefakte predstavljajo UML grafikon stanja, izvorna koda, diagram poteka podatkov, primeri uporabe in drugi. Pri modeliranju je pomembno razumevanje kdaj je smotrno uporabiti določen artefakt in kdaj ne. V nekaterih primerih je bolj efektivno uporabiti diagram kot pa napisati 1024 vrstic kode (Ambler, 2002, 46). Vzporedno modeliranje omogoča simultani zajem informacij več različnih artefaktov (Ambler, 2002, 48). V primeru, da informacije za določen artefakt postanejo neprimerne, se izvede iteracija do naslednjega artefakta. S tem se informacije prenesejo na drug artefakt in posledično omogočijo napredovanje v procesu. Agilno modeliranje si prizadeva k fragmentaciji kompleksnejših nalog v manjše obvladljive entitete, ki se izvajajo v krajših intervalih. Agilni procesi se bolj fokusirajo na efektivne ure razvoja in ne na njihovo količino (Unger in Novak, 2011, 180).

Skupinsko delo definirajo prakse:

* skupinskega modeliranja,
* aktivna participacija vlagateljev,
* kolektivno lastništvo,
* javni prikaz modelov (Ambler, 2002, 44–45).

Skupinsko modeliranje omogoča boljše razumevanje idej in ustvarjanje skupne vizije projekta. Prav tako pripomore k izboljšani komunikaciji, izgradnji skupnega besednjaka in povečanju možnosti izvajanja kvalitetnega dela (Ambler, 2002, 52). Aktivna participacija spodbuja prisotnost uporabnikov ali vlagateljev na lokaciji z namenom izmenjave informacij glede zahtev in pričakovanj projekta (Ambler, 2002, 53). Za kolektivno lastništvo velja, da lahko vsakdo sodeluje pri izgradnji določenega modela, kar predstavlja priložnost večje identifikacije napak (Ambler, 2002, 54). V ta namen se uporabljajo tudi table za javni prikaz modelov, kar spodbuja odprto komunikacijo med zaposlenimi (Ambler, 2002, 55).

Enostavnost definirajo prakse:

* ustvarjanja preproste vsebine,
* enostavnega prikaza modela,
* uporaba enostavnih orodij (Ambler, 2002, 45).

Vsebino je priporočljivo poenostaviti do mere, da še vedno zadovoljuje projektu. Priporoča se izogibanje: križanim tranzicijam, zavitim tranzicijam, diagonalnim tranzicijam, različnim velikostim balonov, velikemu številu balonov (ne več kot 7 +/−2) in nepotrebnim podrobnostim (Ambler, 2002, 57). Uporaba enostavnih orodij deluje, saj je večina diagramov za enkratno uporabo (Ambler, 2002, 58).

Validacijo definirajo prakse:

* upoštevanja preverljivosti,
* dokazovanja s kodo (Ambler, 2002, 45).

Moderni procesi vključujejo aktivnosti testiranja in zagotavljanje kvalitete čez celoten življenjski cikel. Nekateri modeli celo priporočajo definiranje testnih konceptov pred izgradnjo samega sistema (Ambler, 2002, 58). K temu stremi praksa upoštevanja preverljivosti. Ob zgrajenem modelu pa je potrebno preveriti ali je mogoča implementacija poslovnega pravila v model, torej ali ga bo mogoče izvajati. To lahko preverimo z uporabo kode. V praksi se zato priporoča izvajati cikel modeliranja, kodiranja in testiranja (Ambler, 2002, 59).

AM podaja tudi dopolnilne prakse, ki jih lahko poljubno apliciramo in se delijo v tri kategorije: produktivnost, dokumentacija in motivacija (Ambler, 2002, 60). Vključujejo apliciranje standardov modeliranja med katerimi je najbolj poznan UML (Ambler, 2002, 61). Spodbujajo postopno vpeljevanje in uporabo že obstoječih modelov (Ambler, 2002, 64) in modeliranje z namenom razumevanja problema in izboljšanja komunikacije (Ambler, 2002, 69, 70).

5.2 METODA METAME

Metoda MetaME definira potek izgradnje procesnega modela, ki ga lahko razberemo iz slike 5.1. V prvem koraku moramo definirati domeno uporabe metode in njene discipline (Engels in Sauer, 2010, 429). Discipline lahko predstavljajo korake ali faze procesa (zahteve, analiza, razvoj, itd.) (Engels in Sauer, 2010, 429–30). Nato je zgrajen model iz konceptov ali temeljnih aktivnosti domene, katere so organizirane na podlagi disciplin (Engels in Sauer, 2010, 429-30). Ta model predstavlja produktni model meta-metode. Sledi izbira notacije, ki skrbi za primerno predstavitev konceptov. V tem koraku je potrebno identificirati jezike, podjezike in elemente jezika (Engels in Sauer, 2010, 430). Naslednji korak predstavlja definiranje artefaktov. V tem koraku se jeziki in njegovi elementi pripišejo konceptom z namenom izražanja njihovih lastnosti. Medtem, ko domenski model predstavlja semantično domeno (pomen) konceptov, jeziki predstavljajo sintaktično domeno (oblika). Hierarhiji domenskega modela konceptov in artefaktov morata biti kompatibilna. Temu sledi definiranje procesnega modela pri katerem je potrebno definirati aktivnosti, ki so potrebne za izpolnitev nalog. Procesni model je strukturiran iz aktivnosti, mejnikov in elementov nadzora toka. Zadnji korak predstavlja izbira orodij, tehnik in pripomočkov skupaj s koncepti uporabe, ki so potrebni za usmerjanje in poenostavitev izgradnje artefaktov (Engels in Sauer, 2010, 431).

Slika 5.1 temeljni proces meta-metode za izgradnjo metode inženiringa programske opreme



Vir: (Engels in Sauer, 2010, 430)

5.3 UPORABA METOD

Za izgradnjo predloga procesnega modela bomo torej uporabili dve metodi. Sistematično izgradnjo procesnega modela bo vodila metoda MetaME in bo predstavljala rigidno dimenzijo delovnega toka, medtem, ko bodo prakse AM v posameznih fazah delovnega toka predstavljale fleksibilno dimenzijo predlaganega procesa. Ambler (2002) podaja usmeritve uporabe AM z definiranimi procesi kot so XP, SCRUM, DSDM ali UP, ki jih bomo z MetaMe povezali v enoten proces. Kot kaže slika 5.2 bo metoda MetaME predstavljala hrbtenico osnovnega procesnega toka razvoja video iger, AM pa bo s svojimi praksami služila za podporo bazičnemu agilnenu procesu, ki ga bomo izbrali v nadaljevanju.

Slika 5.2 izboljšan temeljni proces meta-metode z uporabo Agilnega modeliranja



Vir: (Ambler, 2002, 10)

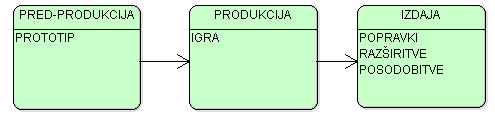
5.4 OPTIMIZACIJA PROCESA

Ker želimo v sklopu RV2 ugotoviti ali predlagani model ustreza optimizaciji z DGMM bomo po izvedbi izgradnje predloga procesnega modela izvedli simulacijo optimizacije s prvim zrelostnim pristopom za igre DGMM. Slednji upošteva faktorje, ki temeljijo na razvojni, potrošniški in poslovni perspektivi (Aleem, Capretz, in Ahmed, 2016a, 57–58). Na podlagi teh perspektiv so avtorji definirali 18 različnih faktorjev (PRILOGA K), ki jih imenujejo procesne aktivnosti razvoja video iger (GDPA)[[38]](#footnote-38) (Aleem, Capretz, in Ahmed, 2016a, 59). Zrelost procesa razdelijo na pet stopenj (naraščajoče): ad-hoc, priložnosten, konsistenten, organiziran in optimiziran (Aleem, Capretz, in Ahmed, 2016a, 61). Vsako stopnjo zrelosti se preverja z namenskim vprašalnikom, katerega izjave se nanašajo na izvajanje GDPA. Ocenjevanje zrelosti poteka tako, da posamezno izjavo (PRILOGA M) potrdimo, če model zagotavlja njeno izvajanje in zavrnemo, če model izvajanja ne podpira. Na koncu seštejemo število potrjenih izjav in s tem dobimo število[[39]](#footnote-39) GDPA, ki se izvajajo. V kolikor je število izvajanih aktivnosti večje od določenega praga[[40]](#footnote-40) pomeni, da je procesni model dosegel zrelosti na stopnji, ki smo jo preverjali. Za doseganje praga zrelosti velja, da se v procesu izvaja 80% aktivnosti glede na vse izjave, ki so podane za določeno stopnjo zrelosti (PRILOGA L) (Aleem, Capretz, in Ahmed, 2016a, 68).

6 PREDLOG MODELA ZA RAZVOJ VIDEO IGER

V tem poglavju bomo izvedli sistematično izgradnjo procesnega modela, ki hkrati podaja usmeritve in dobre prakse apliciranja predlaganega procesnega modela. Na podlagi 4 poglavja praks in usmeritev smo sestavili okvirni predlog, ki zajema osnovne faze procesa in njegove vidnejše lastnosti. Tako je na sliki 6.1 predstavljen model, ki nakazuje osnovne faze in artefakte, ki se v teh fazah pojavijo. Ta model predstavlja najvišjo abstrakcijo procesnega predloga, ki ga bomo podrobneje sestavili v poglavjih, ki sledijo. Podpoglavja si sledijo na podlagi teorije uporabe metode MetaME (pog 5.2).

Slika 6.1 predlagani model na podlagi faz in artefaktov v praksi



6.1 DEFINIRANJE DOMENE IN DISCIPLIN

Domeno predstavlja razvoj video iger. Osnovne faze pri razvoju programske opreme predstavljajo analiza, načrtovanje, kodiranje in testiranje (Ramadan in Widyani, 2013, 95). Faze, ki jih posamezni avtorji navajajo so prikazane v PRILOGA F. Na podlagi literature smo konsolidirali faze razvoja iger in jih kronološko razvrstili: formalizacija problema, razvoj koncepta, pred produkcija, produkcija, testiranje, post produkcija.

6.2 IZGRADNJA DOMENSKEGA MODELA

Da bi s predlaganim modelom lahko vodili tudi izdelavo resnih iger smo v model umestili prvo fazo, ki jo definirata Aslan in Balci pri definiranju procesa za idelavo učnih iger. Aslan in Balci (2015) jo imenujeta formulacija problema. Ta faza definira probleme različnih domen in načine reševanja, ki privedejo do efektivnega učenja na podlagi iger (Aslan in Balci, 2015, 310). Njuna prva faza je unikatna, ker gre za specifično fazo pri razvoju učnih iger in je nismo zasledili pri drugih avtorjih. Naslednja faza, ki jo definirata že sledi standardnim potekom razvoja video iger. Imenujeta jo generacija idej (Aslan in Balci, 2015, 311), ki pa predstavlja prvo fazo po Bates (2004) in Novak (2012). Widyani (2013) to fazo imenuje iniciacija. V tej fazi je predstavljena ideja igre v pisni obliki (Novak, 2012, 352). Čeprav Unger in Novak (2011) označujeta pred-produkcijo za prvo fazo, pa navajata, da se začne po uspešno sprejetem konceptu. Zato bomo pred-produkcijo kronološko postavili za fazo razvoja koncepta. Pred-produkcija je faza kjer se sprejme odločitev kakšna bo igra in kako bo izdelana (Unger in Novak, 2011, 176). Predstavlja fazo načrtovanja (Novak, 2012, 353). Vrhunec pred-produkcije predstavlja dokaz koncepta, ki ga predstavlja prototip. Namen je dokazati zmožnost premagovanja tehničnih zahtevnosti (Bates, 2004, 88). Po potrditvi prototipa se lahko začne produkcija, ki predstavlja fazo razvoja igre (Novak, 2012, 358). Produkcija je kompleksna in časovno potratna. Za uspešno izvajanje faze je priporočena fragmentacija opravil v manjše naloge, ki se jim se strogo sledi (Bates, 2004, 212). Priporočen je tedenski pregled napredka (prav tam). Čeprav se testiranje v smislu produkcijskega testiranja dogaja tudi v fazi produkcije se v industriji iger ločijo posebne faze testiranja. Ko je razvoj igre tako daleč, da je mogoče igro igrati od začetka do zaključka se začnejo faze testiranja (Novak, 2012, 359). Testiranje se deli na Alfa in Beta faze. V razvoju mobilnih iger sta fazi navadno združeni (Unger in Novak, 2011, 176). Alfa faza predstavlja točko v kateri je mogoče testirati celotno igranje. Testna ekipa v tej fazi testira vse module igre, zapiše defekte v bazo in ustvari testni načrt. Beta faza je namenjena popravilu defektov in implementaciji vseh sredstev (zvok, umetnine) v igro. S tem se popolnoma konča produkcijski proces. Namen faze je stabilizacija projekta in eliminacija večine ali vseh defektov pred izdajo. Beta faza se deli tudi na zaprto in odprto beta fazo (Levy in Novak, 2009, 54–55). Za zaprto beta fazo velja privatno testiranje v kateri se razvoj fokusira na poliranje igre (prav tam). Odprta faza pa je navadno uporabljena pri igrah ki vsebujejo spletne komponente (prav tam). V odprti beta fazi so povabljeni zunanji igralci, ki v zameno za brezplačno igranje služijo kot testerji (prav tam). V tej fazi se preveri delovanje strežnikov, uravnoteži igranje in identificira defekte (prav tam). Ko igra prestane beta fazo sledi njena izdaja. V tej fazi ravnateljstvo opravi še en pregled produkta in pridobi seznam prisotnih defektov. Po pregledu sledi izdaja produkta na trg. V industriji iger je za to fazo pogost izraz zlata (Novak, 2012, 362). Po fazi izdaje sledi post-produkcijska faza. Faza predstavlja nadaljevalno produkcijo, ki jo Fullerton (2014) imenuje tudi vzdrževanje. Slika 6.2 tako predstavlja domenski model, ki je zgrajen iz razvojnih faz in hkrati predstavlja podlago za procesni model.

Slika 6.2 domenski model na podlagi razvojnih faz



6.3 IZBIRA NOTACIJ

V tem koraku bomo definirali jezik uporabe, ki nam bo podal notacije s katerimi bomo lahko izrazili natančen potek delovanja procesa. Jezik, ki ga bomo uporabili je UML z izbranimi notacijami v tabeli 6.1. Z notacijami bomo povezali stanja in artefakte v procesni model. UML omogoča več različnih prikazov diagramov. Pri izgradnji bomo uporabili UML diagram aktivnosti, ki služijo za prikaz kontrole poteka med aktivnostmi (Rumbaugh, Jacobson, in Booch, 2004, 37).

Tabela 6.1 prikaz izbranih notacij UML jezika za definiranje procesa

|  |  |  |
| --- | --- | --- |
| Naziv | Namen | Notacija |
| Začetno stanje | Stanje začetka aktivnosti |  |
| Izbira | Stanje, ki omogoča izbiro |  |
| Tranzicija | Prikazuje smer tranzicije |  |
| Stanje | Predstavlja stanje |  |
| Vilice ali združitev | Deli in združuje dejavnosti |  |

Vir: sestavljeno po (Rumbaugh, Jacobson, in Booch, 2004, 35, 90)

6.4 IZGRADNJA ARTEFAKTNEGA MODELA

Če želimo razviti učno igro mora skozi fazo, formalizacije problema. V tej fazi se kreira dokument specifikacije učnega problema (Aslan in Balci, 2015, 310).

Nato sledi faza razvoja koncepta, ki je namen definiranja osrednje funkcionalnosti igre, predstavitev grafičnega izgleda in zgodbe. Dokumente, ki jih razvoj koncepta proizvede so: višji koncept, predlog igre (''pitch doc'') in koncept (Bates, 2004, 203). Vsak tip dokumenta, ki predstavlja koncept ima svoj namen, zato ni potrebno, da uporabimo vseh. V našem primeru bomo uporabili poimenovanja po Novak (2012), ki končni dokument koncepta imenuje predlog igre. Predlog igre definira: zgodbo, like, koncepte umetnin, žanr, igralni pogon, igranje, trg, sestavo ekipe in analizo tveganj (Novak, 2012, 387, 389, 390; Bates, 2004, 204, 205). V kolikor želi razvijalec pridobiti investitorja ali sredstva, mora koncept vsebovati tudi načrt proračuna. Načrt proračuna zajema: napoved stroškov (plače, marketing, sredstva, orodja, prodaja) in napoved prihodkov (Novak, 2012, 390; Bates, 204, 206).

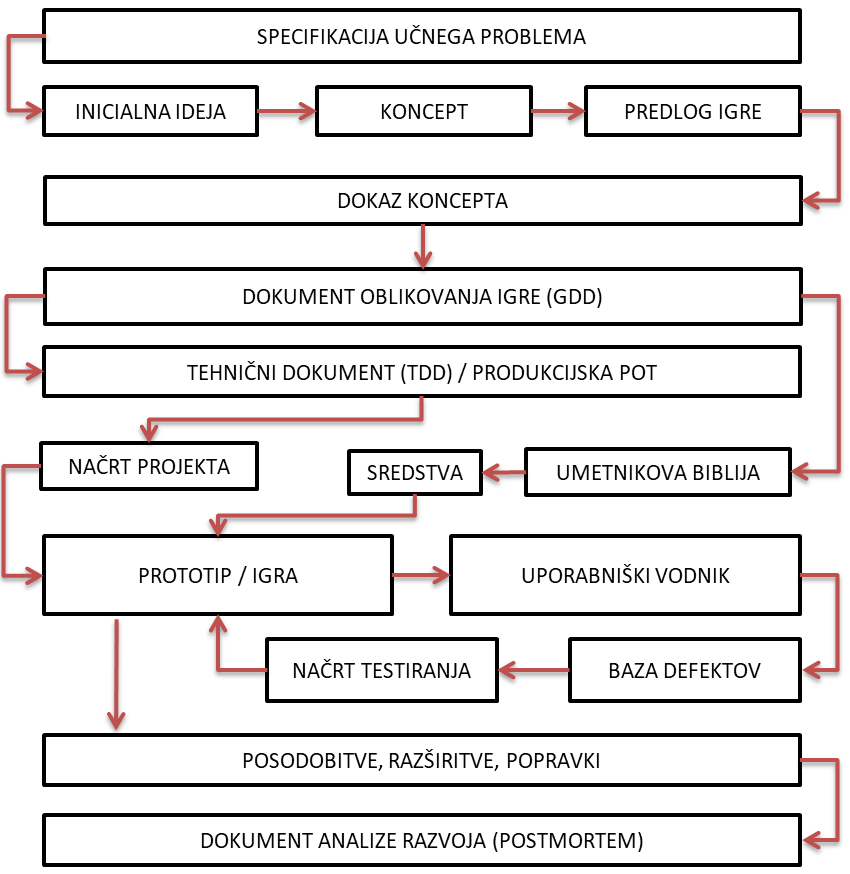
Na podlagi predloga igre se izdela prvi prototip[[41]](#footnote-41) igre. Predstavlja najpomembnejši material, ki ga lahko proizvedemo v tej fazi (Fullerton, 2014, 486). Velja za dokaz delovanja in zmožnost premagovanja tehničnih zahtevnosti (Bartle, 2003, 88). Če je dokaz koncepta uspešen se začne faza oblikovanja ali pred-produkcija (Novak, 2012, 353). Cilj je izdelava GDD, TDD, UB, definiranje produkcijske poti in stvaritev projektnega načrta (Novak, 2012, 353). GDD pa je najobsežnejši dokument v razvoju igre. Novak (2012) navaja, da se ta dokument konstantno posodablja. Schell (2008) dodaja, da se ga občasno posodablja in je ponavadi neurejen. Na polovici projekta se ga opusti, saj vsebuje igra sama vse pomembne podrobnosti (Schell, 2008, 382). Na podlagi GDD tehnična ekipa izdela TDD. Ta dokument predstavlja produkcijsko pot, ki vzpostavi načrt kako se bo razvoj premaknil od koncepta do programske opreme (Novak, 2012, 393). Nato sledi izdelava načrta projekta, ki se začne izdelovati po definiranju grobih nalog, ki jih podaja TDD. Ta dokument vsebuje načrt porabe virov, časovni načrt, mejnike, oceno, stroškov, seznam sredstev za izdelavo in izbrano programsko opremo (programi, igralni pogoni) (Novak, 2012, 393, 394; Bates, 2004, 209). Zadnje dejanje pred-produkcijske faze predstavlja izdelava prototipa (Bates, 2004, 207, 208). Ta artefakt je lahko analogne ali digitalne oblike. Pred izdelavo digitalnega je dobra praksa izdelava enostavnega analognega prototipa, ki ima namen zagotoviti zabavno in prepričljivo igralno mehaniko. Končan prototip je lahko dokaz delovanja produkcijske poti od ideje do realizacije (Novak, 2012, 354). Za kreiranje prototipa se navadno uporablja tehnika vertikalnega reza[[42]](#footnote-42), ki pomeni, da vsebuje ravno toliko vsebine, da poda jasen občutek igranja in takšni ponavadi vsebujejo veliko začasnih sredstev (Mitchell, 2012, 64, 107).

Nato sledi faza produkcije. V produkciji se kreirajo: različice igre (prototipi), uporabniški vodnik, UB in sredstva (Mitchell, 2012, 218). Za razvoj igre se priporoča pristop, ki je fokusiran na igralca (Fullerton, 2014, 10; Adams, 2013, XII). To pomeni, da s pomočjo introspekcije (Schell, 2008, 14) pred kodiranjem definiramo izkušnjo, katero želimo podoživeti ob igranju. To izvajamo iterativno z izdajanjem različic. Različica zajema vsa razpoložena sredstva in predstavlja verzijo igre katero lahko pregledamo in testiramo (Mitchell, 2012, 218). Izdelava UB se lahko začne že v pred-produkciji, vendar po definiranju žanra v GDD (Novak, 2012, 393). Navadno se UB gradi iterativno v času produkcije, ker se jo v fazi pred-produkcije ne potrebuje (Bartle, 2003, 87, 88). UB Služi za definiranje umetniškega sloga igre (Bates, 2004, 208) na podlagi katerega se v tej fazi prične razvoj končnih sredstev (Fullerton, 2014, 414). Dopolnjevanje priročnika je priporočljivo vzporedno z razvojem. Vsebuje razlago igre, vodnika po namestitvi, hiter začetek in detajlno razlago funkcionalnosti uporabniškega vmesnika (Rucker, 2002, 6).

Produkciji sledi faza testiranja. Ta faza predstavlja zagotavljanje kakovosti[[43]](#footnote-43) in skrbi, da igra zadovoljuje vse potrebe preden se izda na trg (Levy in Novak, 2009, 57). V tem koraku se izdelata podatkovna baza defektov in načrt testiranja. Po testiranju sledi faza post-produkcije. Artefakte v tej fazi predstavljajo: popravki, posodobitve in razširitve. Popravki so brezplačne verzije in so ustvarjene z apliciranjem popravkov na originalno različico izdaje. Popravki, so lahko aplicirani tudi z namenom reševanja ostalih produkcijskih defektov. Posodobitve izboljšujejo originalno različico igre. Te so večinoma ustvarjene z namenom podaljšanja življenjske dobe igre. Razširitve lahko delujejo kot samostojne igre ali pa potrebujejo originalno igro za delovanje (Novak, 2012, 365). Po izdaji igre se izdela dokument analize razvoja, ki pripomore k izboljšanju procesa v prihodnjih projektih. McAllister in White (2015) zaradi težav analiziranja podatkov priporočata standardizacijo tega dokumenta.

Slika 6.3 prikazuje model artefaktov, ki je ustrezno kronološko razvrščen.

Slika 6.3 model artefaktov pri razvoju video iger zgrajen na podlagi pregleda literatur in praks



6.5 DEFINIRANJE PROCESNEGA MODELA

S pomočjo definiranih UML notacij in združevanjem artefaktnega in domenskega modela smo definirali UML diagram aktivnosti, ki predstavlja procesni model za igre (sl. 6.4). Bela stanja so aktivnosti, ki predstavljajo fazo razvoja koncepta. Ta se konča z dokazovanjem koncepta. Slednjega predstavlja prototip in velja za poenostavljeno različico igre. Oblikovalci iger jih uporabljajo za testiranje funkcionalnosti (Adams, 2013, 49). Uporaben je tudi za dokazovanje izvedljivosti idej, saj predstavlja delujoči model ideje. Z njim si omogočimo formalizacijo ideje in izolacijo problemov (Fullerton, 2014, 197). V tem procesnem modelu se prototip pojavi že zgodaj v konceptualni fazi, ki dokazuje delovanje ideje preden se začne dolga faza oblikovanja. Ta lahko traja več mesecev (Adams, 2013, 49). Proces se nadaljuje v želeno stanje, ki predstavlja pred-produkcijo. Ta se navadno konča, ko je investitor že videl igralno različico igre in je zadovoljen z ekipo in delom. Če investitor projektu poda zeleno luč se začne produkcijska faza (Adams, 2013, 46).

Produkcijska faza se odvija med aktivnostmi načrtovanja, kodiranja in evalvacije (Novak, 2012, 367). Fullerton (2014) jo opisuje kot iterativni proces med testiranjem igranja, evalvacije in razvoja. V našem modelu se testiranje igranja izvaja v stanju evalvacije.

Po načrtovanju iteracije se izvede izbira orodja in izdela prototip v aktivnosti kodiranja. V fazi evalvacije ekipa igro testira in se odloči ali se vrne v fazo načrtovanja in nadgradi prototip. Ta proces se izvaja dokler igra ne predstavlja več prototipa ampak končno igro (Novak, 2012, 367). Vsaka iteracija mora imeti svoj cikel razvoja skupaj z načrtovanjem potreb, artefaktov in urnika (Novak, 2012, 367), zato ima model na sliki 6.3 v produkcijski fazi dve aktivnosti (načrtovanje iteracij, izbire orodij), ki vsebujeta prakse in orodja, ki podpirajo te potrebe. Stanja produkcije so označena z rdečo barvo in se odvijajo iterativno dokler igra ni končna. Vzporedno se v produkciji odvijajo tudi procesi posodabljanja baze defektov, sredstev in dokumentacije. Baza defektov je označena z vijolično barvo in predstavlja aktivnost, ki se izvaja v več fazah (produkcija, testiranje, post-produkcija).

Posebnost predlaganega procesnega modela je implementacija toka razvoja sredstev, kar priporoča tudi Bates (Bates, 2004, 171). Aslan potrjuje, da se to prične izvajati v fazi elaboracije (produkcija) (Adams, 2013, 13), to potrjuje tudi Mitchell (2012). Sredstva se proizvajajo do faze testiranja. Ponavadi v alfa fazi še niso povsem izdelana (Bates, 2004, 214; Novak, 2012, 359), vendar morajo biti do začetka beta faze permanentna, drugače se beta ne izvede (Levy in Novak, 2009, 53). Mesta neizdelanih sredstev zaradi razvoja zapolnjujejo začasna sredstva[[44]](#footnote-44) (Levy in Novak, 2009, 52; McAllister in White, 2015, 16). Začasna sredstva se med razvojem pogosto uporablja za testiranje uporabnosti brez distrakcije estetike (Novak, 2012, 261). Takšna praksa se pogosto uporablja in je priporočljiva, saj onemogoča ozka grla produkcije (Zagal in Altizer, 2015, 746). Za zvočne efekte ali določene teksture se navadno ne priporoča uporaba začasnih sredstev. Velikokrat se zgodi, da se odkrije manjkajoča tekstura globoko v zaključni beta fazi (Levy in Novak, 2009, 79, 82). Še težje pa je odkriti manjkajočo zvočno podlago (Levy in Novak, 2009, 79, 82). Za izdelavo sredstev se priporoča linearni proces z izboljševanjem serije prototipov. To vodi v ponavljajoč kaskadni način razvoja sredstev (Bates, 2004, 227), kar priporočajo tudi Manjuka in kolegi (2016). Proces razvoja sredstev se začne z razvejanjem tranzicije iz aktivnosti oblikovanja igre. Ena tranzicija prenese začasna sredstva v razvoj, druga gre v prvo aktivnost kreiranja konceptualnih sredstev. Za koncepte se priporoča, da so enostavni, saj služijo za predstavitve (Mitchell, 2012, 85). Po sprejetju koncepta se proces nadaljuje v aktivnost izvedbe, kjer se sredstvo izdela (Mitchell, 2012, 103). Izdelano sredstvo se preveri s testiranjem v različici igre (Levy in Novak, 2009, 79). Ko se sredstvo odloži v razvoj se mora izvesti celoten cikel razvoja, ki predstavlja testiranje sredstva. Cikel izdelave sredstva se vedno konča v posodobitvi dokumentacije (seznam sredstev v projektnem načrtu ali v GDD PRILOGA H – SECTION XI ASSET LIST). Če je sredstvo sprejeto je na voljo za posodobitev igre drugače se vrne v aktivnost izvedbe.

Dokumentacija se lahko zaradi iterativnega in inkrementalnega razvoja drastično spremeni (O’Hagan in O’Connor, 2015, 5). Slabo nadzorovane spremembe v dokumentaciji lahko prerastejo v večje težave, ki lahko negativno vplivajo na funkcionalnost igre, sredstva ali časovne mejnike ((O’Hagan in O’Connor 2015, 5). Aleem in kolegi (2016) priporočajo vpeljavo dokumentacije, ki beleži spremembe in povezane težave. Ker razvoj iger stremi k agilnim praksam, ki se ogibajo presežku dokumentacije (O’Hagan in O’Connor, 2015, 5) predvideva naš model sprotno posodobitev dokumentacije po vsakem inkrementu igre ali izdelave sredstva. Podjetja imajo lahko različne standarde dokumentacije. Nekatera ustvarijo več dokumentov, druga manj. Navadno je dokumentacija obratno proporcionalna stopnji zaupanja založnika in izkušenosti razvojne ekipe. Ne glede na izkušnje ekipe, dobra dokumentacija pripomore k boljšem razvoju (III, 2004, 319).V času produkcije se zaradi dobro definiranih zahtev število iteracij zmanjša, kar vodi v spiralno orientiran razvoj, ki ga priporoča več avtorjev (Kanode in Haddad, 2009, 264; Manjuka, Chakradhar Raju, in Sai Chand, 2016, 557–58; Ruonala, 2016, 9). Predlagani procesni model to omogoča, saj izbor agilnih praks v fazi načrtovanja iteracije ni nujen. Za lažjo definiranje dokumentacije priporočamo uporabo osnutkov. V PRILOGA H je primer osnutka GDD.

Na sliki 6.4 modra stanja predstavljajo faze testiranja. V fazo testiranja preidemo, ko igra predstavlja končni izdelek ali če nas obvezujejo kakršnekoli časovne ali pogodbene obveznosti. Ta faza se deli na alfa in beta fazo testiranja. Alfa stremi k zaklepanju funkcionalnosti medtem, ko se beta faza osredotoča na poliranje igre in reševanje defektov (Levy in Novak, 2009, 52–53). V obeh fazah se proces nadaljuje iterativno dokler obstajajo defekti ali dokler ti niso označeni za nepomembne (WNF, NAB)[[45]](#footnote-45) (Levy in Novak, 2009, 103), zato lahko v procesu pridemo do izdaje tudi preko baze defektov. Izdaja predstavlja zadnjo fazo v procesu in je v modelu označena z rumeno barvo. Izdelava popravkov po končani igri je skoraj neizogibno. Vzrok ne leži nujno v rani izdaji temveč v tisoče različnih strojnih konfiguracijah katere je nemogoče v celoti predvideti in testirati (Bates, 2004, 216). Prav tako kot popravki predstavljajo posodobitve majhne projekte, ki zahtevajo načrtovanje mejnikov, testiranja in druge elemente dobrih praks (prav tam). Procesni model to zagotavlja, saj se ob potrebi po posodobitvah (nadgradnje, popravki, razširitve) ponovno začne iterativni proces razvoja. Proces razvoja igre se tako nikoli ne konča. Razvijalci kljub izdanem produktu na podlagi povratnih informacij objavljajo popravke in včasih celo dodajo funkcionalnosti (Fullerton, 2014, 421).

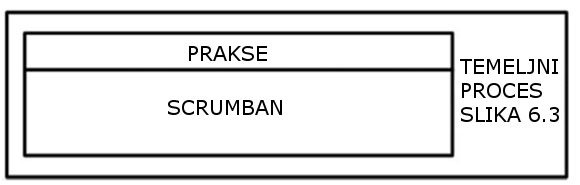
6.4 procesni model za video igre zgrajen na podlagi združevanja procesnega in artefaktnega modela   


6.5 IZBIRA ORODIJ

Izbira se odvija v času produkcije ali post produkcije (posodobitve) pred vsakim začetkom iteracije razvoja. Ta korak omogoča AM v stanjih načrtovanja iteracije in izbire orodij. Orodja niso determinirana vendar se izbirajo po potrebi projekta ali same naloge. Tukaj se izrazi praksa, da se z aplikacijo AM v proces modelira več kot prej. V vsakem koraku aplikacije AM je cilj pridobiti globlje razumevanje enega ali več perspektiv sistema in s tem prakso, ki nam ustreza (Ambler, 2002, 9).

AM nam ponuja različne prakse, ki so se razvile na podlagi osnovnih kategorij znotraj posameznih agilnih procesnih modelov. Nekaj vidnejših praks posameznih procesnih modelov je zbranih v PRILOGA I. Te prakse bomo uporabili za podporo našemu temeljnemu procesu na sliki 6.4 v kombinaciji s popolnim agilnim procesom kot to prikazuje slika 6.5 (Ambler, 2002, 10).

Slika 6.5 abstrakcija aplikacije predlaganega procesnega modela



V razvoju video iger obstajajo tehnične in ne tehnične usmeritve, ki omogočajo uspeh projektov. Tehnične so zagotoviti: tehnični, časovni in projektni načrt, razumevanje tveganj in znanje ponastavitve projekta v težavah (Bates, 2004, 219). Ne tehnične predstavljajo: vzdrževanje komunikacije, sledenje stroškom, vzdrževanje ekipnega duha in identitete (Bates, 2004, 213). Vse te usmeritve lahko zadovoljimo z izbiro pravega orodja ali več njih. Če se orodje ne izkaže za pravo je praksa AM v naslednji iteraciji izbrati drugega (pog 5.1).

Pri načrtovanju razvoja je priporočljiva fragmentacija večjih nalog v manjše obvladljive naloge (Bates, 2004, 212). Za takšno opravilo je priporočljiva uporaba Scrum prakse, fragmentacije produkta na obvladujoče entitete (Sommerville, 2010, 74). Vendar Scrum ne definira tehničnih aspektov agilnega pristopa ampak se osredotoča le na upravljanje inkrementalnega razvoja (Sommerville, 2010, 72). Zato so najverjetneje podjetja Scrum metodologijo posvojila v različnih oblikah.

Dr. Lennart E. Nacke za Novak (2012) navaja:

*»V naši ekipi smo dodali samolepilne liste na tablo za spremljanje nalog, kar omogoča izogibanje presežku poročanja in vizualno predstavo napredka posameznikov ali njihovih težav vsem v razvoju.« Dr. Lennart E. Nacke.*

To nakazuje na uporabo hibrida Scrum in Kanban metodologije imenovano Scrumban, ki jo predstavita Yilmaz in O'Connor (2016). S Kanban dimenzijo metodologije vsak trenutek vemo koliko dela je v obtoku (Yilmaz in O’Connor, 2016, 242). Poleg tega lahko na podlagi obremenitve izračunamo stroške in napredek projekta (prav tam). Vodenje projekta se izvaja s Scrum praksami, kakor tudi fragmentacija dela, ki se izvaja z uporabo definiranja zgodb (prav tam). Tudi v našem primeru predlagamo uporabo Scrumban pristopa. Ta izboljša Kanbanov WiP z vpeljavo sistema vlečenja, ki omeji količino dela v teku (prav tam). Obremenitev razvoja je tako minimalna (prav tam). Ocena hitrosti dela tako ni odvisna od števila zgodb, ki jih razvoj proizvede ampak od dnevnika zaostankov (prav tam). Prav tako model posodobi dnevne sestanke, katere nadgradi v pol strukturirane intervjuje (prav tam). V realnosti veliko razvojnih ekip zgreši pri napovedi trajanja projekta (Novak, 2012, 358). Zavoljo tega se proti koncu produkcijskega cikla velikokrat pojavi čas krize[[46]](#footnote-46) (prav tam). Z uporabo praks tega procesnega modela pridobimo na reševanju težav slabih časovnih napovedi razvoja. Za podporo Scrumban predlagamo agilni praksi XP pristopa. Konstantno prisotnega naročnika (vlagatelja) in fiksnega 40 urnega tedenskega delavnika, ki še podpre reševanje kriznega časa in onemogoča preobremenitev zaposlenih.

Menimo, da so zaposleni najpomembnejši viri in jih moramo tako tudi obravnavati. Razvojne ekipe lahko štejejo enega do treh razvijalcev pri majhnih neodvisnih razvojnih timih, medtem, ko lahko ekipa šteje več sto zaposlenih pri velikih organizacijah (Mitchell, 2012, 56). Čeprav je v teoriji mogoče izdelati igro le z umetnikom in programerjem je potrebno veliko več za sestavo temeljev uspešne igre (Novak, 2012, 319). Vsako igro je potrebno upravljati, oblikovati, programirati, testirati. Prav tako igra potrebuje zvok, umetnine in glasbo (Bates, 2004, 151). Vloge zaposlenih se delijo glede na discipline: produkcija, oblikovanje, programiranje, umetnost, zvok in testiranje.

Tako osnovno ekipo lahko sestavljajo producent, oblikovalec, programer in kreator umetniških sredstev, (Novak, 2012, 319–39; Bates, 2004, 153–79) pri čemer lahko glede na velikost ekipe ena oseba pokriva več vlog (Novak, 2012, 319; Bates, 2004, 151). Pri vodenju takšnih multidisciplinarnih ekip so potrebne usmeritve v obliki mehkih veščin. Priporoča se spodbujanje idej in sodelovanje vse ekipe v oblikovanju igre (Schell, 2008, 375). To omogoča večjo izbiro idej, njihovo hitro ustvarjanje in zadovoljstvo ekipe (prav tam). Sem spadajo še veščine spodbujanja komunikacije in njene kvalitete: objektivnost, jasnost, vztrajnost, spoštovanje, zaupanje, iskrenost, zasebnost in skupnost (Schell, 2008, 376–79)

Proces je priporočeno podpreti tudi z uporabo digitalnih programskih paketov. Ravnateljstvo že dolgo uporablja takšna orodja za administracijo individualnih ali skupine projektov. Orodja kot so: Scrumwise, Kanbanery, Yodiz, ZenHub, Leankit, Jira, Trello, so idealna za spremljanje nalog, saj omogočajo kreiranje Kanban tabel. Poleg teh poznamo še Fat Panda, Pivotal Tracker, Active Collab, HacknPlan, Asana in Slack. Orodja lahko omogočajo tudi definiranje t.i. Gantt[[47]](#footnote-47) ali PERT[[48]](#footnote-48) grafov, ki zadostijo definiranjem časovnih načrtov.

Novi paketi takšnih orodij prinašajo prednosti, saj omogočajo upravljanje s tveganji, upravljanje z najboljšimi praksami, e-poštne notifikacije in kolaboracije.

Za izbiro ustreznega programskega paketa predlagamo analitični hierarhični proces[[49]](#footnote-49), ki omogoča izbiro paketa na podlagi teorije izbire po Saatyu. Primarna naloga postopka je kvantifikacija relativnih prioritet za podano zbirko alternativ (PRILOGA J) na podlagi sodb, ki jih poda odločevalec (Ahmad in Laplante, 2006, 76). Kriteriji se spreminjajo in predlagani predstavljajo le ilustracijo za procesa izbora. Kriterije je potrebno definirati na podlagi hierarhije, ki predstavlja za nas najbolj primerno orodje (Ahmad in Laplante, 2006, 81).

Ker lahko projekti zaradi količine defektov postanejo neobvladljivi se priporočajo rešitve za spremljanje izvorne kode (McShaffry in Graham, 2012, 111). Komercialne rešitve predstavljajo SourceSafe, Perforce in AlienBrain, odprtokodne pa Subversion, TortoiseSVN in Git (McShaffry in Graham, 2012, 15–114).

6.6. OPTIMIZACIJA PROCESA

V sklopu RV2 smo želeli ugotoviti ali je predlagani model primeren za optimizacijo. Zato smo z DGMM izvedli simulacijo optimizacije. V prvem delu smo izbrali vprašalnik, ki vsebuje izjave za ocenjevanje najvišjega nivoja zrelosti. Nato smo izjavam, ki se nanašajo na izvajanje GDPA (PRILOGA M) za ocenjevanje optimalne zrelosti po DGMM podali odgovore. V kolikor predlagani model zagotavlja izvajanje posamezne GDPA smo v tabelo zapisali DA, v kolikor ne zagotavlja izvajanja smo zapisali NE. Rezultati so v PRILOGA N.

Ugotovili smo, da naš model ne zagotavlja izvajanja vseh GDPA, ki jih predvideva DGMM. Pri posodabljanju dokumentacije smo uporabili drugačen pristop kot ga predvideva DGMM v izjavi S.5.1.4. Pri našem modelu smo upoštevali najmanjši potreben obseg dokumentacije, da lahko vzdržujemo kvaliteten razvoj. Dodatna dokumentacija predstavlja odvečno delo, poleg tega pa zavira agilnost katero smo želeli doseči. Izjavi S.5.8.1, S.5.8.2 se nanašata na aktivnosti povezane z razvojem in vzdrževanjem igralnih pogonov. Za to nalogo je odgovoren programer (Novak, 2012, 334). V literaturi je razbrati, da se je pri številčno manjših ekipah bolje usmeriti na oblikovanje in izdelavo igre kot na izgradnjo igralnega pogona (Novak, 2012, 344). Sama izdelava pogona, vzdrževanje, nudenje podpore več platformam je časovno in stroškovno potratno opravilo za manjše ekipe. (McShaffry in Graham, 2012, 22). Ti dve izjavi sta predvsem odvisni od številčnosti ekipe in njihove usposobljenosti, na kar procesni model ne more vplivati. Izjavo S.5.15.2 smo zavrnili, ker se v prvi vrsti fokusiramo na igralca in uporabniško izkušnjo. Procesni model omogoča sicer oba pristopa strategij. Vendar se igre bolj fokusirajo na uporabniško izkušnjo katero želimo doseči z iterativnim razvojem prototipov (Fullerton 2014, 15)

Rezultati (PRILOGA N) so pokazali, da predlagani model od skupno 43 GDPA omogoča izvajanje 39 GDPA. Po metodi DGMM je za ocenjevanje najvišje stopnje zrelosti potrebno izvajanje vsaj 36 GDPA (PRILOGA L). Zato kljub nekaj zavrnjenim izjavam potrjujemo, da bi bil procesni model v praksi primeren za optimizacijo z DGMM.

7. ZAKLJUČEK

Vsestanski potencial iger, predstavlja veliko motivacijo za zagon razvoja številnim organizacijam in posameznikom. Vendar se optimistični načrti razvoja iger, kljub tehnično podkovanim kadrom in jasnim poslovnim ciljem, zaradi slabo definiranega procesa, pogosto končajo v neobvladovanju projekta ali finančnih izdatkih, ki zadušijo razvoj.

Da bi našli najustreznejši razvojni model smo naredili pregled vseh procesnih modelov, ki so na voljo za razvoj iger. Ugotovili smo, da obstaja šest kategorij modelov: ad hoc, sekvenčni, evolucijski, specializirani, agilni in procesni modeli za igre. Nato smo v sklopu raziskovalnega vprašanja – 1 (RV1) izvedli pregled uporabe modelov v praksi. Ugotovili smo, da se v praksi pretežno uporabljajo modeli, ki spadajo v agilno kategorijo ali vsaj indicirajo lastnosti agilnih praks. Čeprav obstajajo tudi procesni modeli za igre se najpogosteje uporabljata XP in Scrum, ki sta pogosto združena z drugimi modeli. To potrjujejo študije uporabe, ki nakazujejo uporabo hibridnih pristopov kjer je vsaj en korak v procesu podprt ali izražen z agilnimi praksami. Kljub pogosti rabi pa so XP, Scrum in njuni hibridi prevzeti od standardnega razvoja programske opreme in niso specializirani za igre. Življenjski cikel razvoja iger zaradi svoje agilnosti, kreativnosti, tendence po spremembi dokumentacije, multidisciplinarnosti, abstraktno definiranih zahtev in krajšega življenjskega cikla, poseduje lastnosti, ki ga evidentno ločujejo od standardnega razvoja programske opreme. Nabor dokumentacije je pri razvoju iger širši in je lahko skupek več samostojih dokumentov na katerih lahko sočasno dela več različnih strokovnjakov. Ti dokumenti so terminološki konstrukti, ki so nastali v domeni razvoja video iger in jih standarden razvoj programske opreme ne definira. Neenotnost terminologije se je prikazala pri pregledu faz in določanju artefaktov. Niti dva avtorja nista definirala istih faz ali artefaktov. Nekateri avtorji so definirali več faz, drugi manj. Artefakti so bili definirani v različnih obsegih in njihovo poimenovanje ni bilo enotno. Šibkost terminologije in konceptov vpliva na to, da niti modeli, ki so specializirani za igre ne definirajo najpomembnejše aktivnosti v razvojnem procesu igre. Ne obstaja procesni model, ki bi definiral aktivnost razvoja kreativnih sredstev. Prav tako ne obstaja procesni model, ki bi definiral razvoj zabavne izkušnje, ki je ne moremo poustvariti brez načrtovanja teh sredstev. To je problematično ob dejstvu, da apliciranje delovnega toka razvoja sredstev v razvojni proces iger, ustvari dva vzporedna si procesa, ki morata biti usklajena. Takšno implementacijo in sinhronizacijo ne predvideva noben pregledan proces. Predvidevamo, da to povzroča ozka grla, ki so nato vzrok za nastanek v nalogi omenjenih (pog. 4.1). najpogostejših zapletov v razvoju.

Zato smo v skopu raziskovalnega vprašanja – 2 (RV2) predlagali procesni model za razvoj video iger, ki bi služil kot prikaz sistematične izgradnje procesnega modela z upoštevanjem lastnosti, zahtev in potreb discipline razvoja video iger. Predlagani procesni model smo sestavili z uporabo metode MetaME, ki ima prednost, da omogoča izgradnjo procesnega modela okoli konceptov, ki jih podaja disciplina. Koncepti so bili definirani z identifikacijo in konsolidacijo aktivnosti, ki jih predstavljajo posamezne faze discipline razvoja video iger. Tako smo dobili fiksni model, ki predstavlja delovni tok oz. procesno dimenzijo po metodi MetaME. Po istem vzorcu smo sestavili artefaktni model, ki predstavlja produktno dimenzijo po MetaME. Ta je prav tako kot procesni model kronološko razvrščen na podlagi najboljših praks in usmeritev. Z združitvijo teh dimenzij smo sestavili procesni model, ki povzema koncepte, prakse in terminologijo, ki ustrezajo disciplini razvoja video iger. Medtem, ko nam predlagan procesni model služi kot rigidni delovi toku nam znotraj posameznih faz uporaba agilnega modeliranja omogoča kreativno svobodo. Fleksibilnost predlaganega modela se izraža s praksami in usmeritvami implementacije modela in izbire orodij. Tako smo sestavili model, ki ima jasno določene faze znotraj katerih se izvaja fleksibilen razvoj, ki ga predstavljajo priljubljene agilne prakse. Rigiden delovni tok omogoča jasno definiranje poteka dela, napoved napredka razvoja in določitev jasnih mejnikov. Z uporabo predlaganega modela predvidevamo zmanjšanje tveganja, natančnejšo napoved trajanja projekta in onemogočanje kriznega časa, ki veljajo za najpogostejše težave v razvoju. Posebnost predlaganega procesa je definiranje dodatnega delovnega toka kreiranja sredstev in njegovo sinhronizacijo z razvojem. Razvoj poteka nemoteno z uporabo začasnih sredstev, katera posodablja glede na razpoložljivost in dostavo izdelanih s strani delovnega toka sredstev.

Z namenom pregleda ustreznosti smo predlagani model preverili s prvim zrelostnim modelom za igre DGMM. Predlagani procesni model velja za ustreznega, saj omogoča izvajanje zadostnega števila procesnih aktivnosti na podlagi DGMM. Kljub temu pa ne moremo potrditi uporabne vrednosti, ker ga nismo uporabili v praksi. Raziskava, ki bi ovrednotila uporabnost modela bi bila koristna za prihodnje optimizacije modela.

Čeprav smo v delu definirali model, ki je bil sestavljen iz več pristopov ne priporočamo hibridizacije modelov, saj vodi v mutirana orodja decentraliziranega upravljanja delovnega toka. Za to je predvsem odgovorna pomankljivost in neenostnost terminologije, ki otežuje prosperiteto discipline. Disciplina razvoja video iger potrebuje več tovrstnih študij s katerimi bo zgradila trdnejša temelja in omogočila teoretski in konceptualni napredek. S tem bo disciplina video iger pridobila na terminoloških temeljih, ki bodo služili za opis lastnega inženiringa metodologij.

9 VIRI

Abrahamsson, Pekka, Outi Salo, Jussi Ronkainen, in Juhani Warsta. (2017). Agile Software Development Methods: Review and Analysis. Dostopno prek http://arxiv.org/abs/1709.08439.

Adams, Ernest. (2013). *Fundamentals of Game Design*. 3 edition. Berkeley, CA: New Riders.

Ahmad, Norita, in Phillip Laplante. (2006). *Software Project Management Tools: Making a Practical Decision Using AHP*. doi:10.1109/SEW.2006.30.

Aktaş, AZ, in E Orçun. (2016). A Survey of Computer Game Development. *The Journal of Defense Modeling and Simulation* 13 (2): 239–51. doi:10.1177/1548512914554405.

Aleem, Saiqa, Luiz Fernando Capretz, in Faheem Ahmed. (2016a). A Digital Game Maturity Model (DGMM). *Entertainment Computing* 17 (Supplement C): 55–73. doi:10.1016/j.entcom.2016.08.004.

Aleem, Saiqa, Luiz Fernando Capretz, in Faheem Ahmed. (2016b). Game Development Software Engineering Process Life Cycle: A Systematic Review. *Journal of Software Engineering Research and Development* 4 (1): 6. doi:10.1186/s40411-016-0032-7.

Ambler, Scott. (2002). *Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process*. 1 edition. New York: Wiley.

Anderson, David J., in Andy Carmichael. (2016). *Essential Kanban Condensed*. Blue Hole Press.

Aslan, Serdar, in Osman Balci. (2015). GAMED: Digital Educational Game Development Methodology. *SIMULATION* 91 (4): 307–19. doi:10.1177/0037549715572673.

Babu, K. Subhash, in R. Maruthi. (2013). Lifecycle for Game Development to Ensure Enhanced Productivity. *International Journal of Innovative Research in Computer and Communication Engineering* 1 (8): 1490–1503.

Barbosa, Ellen. (2017). Game-Scrum: An Approach to Agile Game Development. Dostopno prek https://www.academia.edu/15250630/Game-Scrum\_An\_Approach\_to\_Agile\_Game\_Development.

Bartle, Richard. (2003). *Designing Virtual Worlds*. New Riders Games.

Bates, Bob. (2004). *Game Design*. 2 edition. Boston, Mass: Cengage Learning PTR.

Blow, Jonathan. (2004). Game Development: Harder Than You Think. *Queue* 1 (10): 28–37. doi:10.1145/971564.971590.

Booth, Andrew, Diana Papaioannou, in Anthea Sutton. (2012). *Systematic Approaches to a Successful Literature Review*.

Breathing Labs. (b.d.). Dostopno prek https://www.breathinglabs.com/.

Clark, Douglas B., Emily E. Tanner-Smith, in Stephen S. Killingsworth. (2016). Digital Games, Design, and Learning: A Systematic Review and Meta-Analysis. *Review of Educational Research* 86 (1): 79–122. doi:10.3102/0034654315582065.

Cockburn, Alistair. (2006). *Agile Software Development: The Cooperative Game*. 2 edition. Upper Saddle River, NJ: Addison-Wesley Professional.

Cooper, Kendra M. L., in Walt Scacchi, ur. (2015). *Computer Games and Software Engineering*. 1 edition. Boca Raton: Chapman and Hall/CRC.

Duka, Denis, in Lovre Hribar. (2010). Test Driven Development Method in Software Development Process. Dostopno prek http://bib.irb.hr/prikazi-rad?rad=483416.

Engels, Gregor, in Stefan Sauer. (2010). A Meta-Method for Defining Software Engineering Methods. *Graph Transformations and Model-Driven Engineering*, 411–40. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-17322-6\_18.

Entertainment Software Association. (b.d.). Dostopno prek http://www.theesa.com/wp-content/uploads/2017/04/EF2017\_FinalDigital.pdf.

Eurostat. (2017). Dostopno prek http://ec.europa.eu/eurostat/web/labour-market/earnings/database.

Ey. (b.d.). Dostopno prek http://www.ey.com/Publication/vwLUAssets/Measuring\_cultural\_and\_creative\_markets\_in\_the\_EU/$FILE/Creating-Growth.pdf.

Flynt, Ph D. John P., in Omar Salem. (2004). *Software Engineering for Game Developers*. 1 edition. Boston, Mass: Course Technology PTR.

Fullerton, Tracy. (2014). *Game Design Workshop: A Playcentric Approach to Creating Innovative Games, Third Edition*. 3 edition. Boca Raton: A K Peters/CRC Press.

Gamasutra. (b.d.). Dostopno prek https://www.gamasutra.com/salarysurvey2014.pdf.

Gamedev. (2013). Dostopno prek https://www.gamedev.net/articles/programming/general-and-gameplay-programming/game-unified-process-r1940/.

Hamari, Juho, in Lauri Keronen. (2017). Why do people play games? A meta-analysis. *International Journal of Information Management* 37 (3): 125–41. doi:10.1016/j.ijinfomgt.2017.01.006.

EcuRed. (b.d.). Dostopno prek https://www.ecured.cu/Game\_Unified\_Process.

Gareth Schott, in Darrin Hodgetts. (2006). Health and Digital Gaming: The Benefits of a Community of Practice. *Journal of Health Psychology* 11 (2): 309–16. doi:10.1177/1359105306061189.

III, Richard Rouse. (2004). *Game Design: Theory and Practice*. 2 edition. Plano, Tex: Jones & Bartlett Learning.

Jalote, Pankaj, Aveejeet Palit, Priya Kurien, in V. T. Peethamber. (2004). Timeboxing: a process model for iterative software development. *Journal of Systems and Software* 70 (1): 117–27. doi:10.1016/S0164-1212(03)00010-4.

Janes, Andrea, in Giancarlo Succi. (2014). *Lean Software Development in Action*. 2014 edition. New York: Springer.

Kanode, Christopher M., in Hisham M. Haddad. (2009). Software Engineering Challenges in Game Development. *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, 260–265. ITNG ’09. Washington, DC, USA: IEEE Computer Society. doi:10.1109/ITNG.2009.74.

Keith, Clinton. (2010). *Agile Game Development with Scrum*. 1 edition. Upper Saddle River, NJ: Addison-Wesley Professional.

Kinestica. (b.d.). Dostopno prek http://www.kinestica.com/.

Koster, Raph. (2013). *Theory of Fun for Game Design*. 2 edition. Sebastopol, CA: O’Reilly Media.

Koutonen, Jussi, in Mauri Leppänen. (2013). *How Are Agile Methods and Practices Deployed in Video Game Development? A Survey into Finnish Game Studios*. doi:10.1007/978-3-642-38314-4\_10.

L. Rakestraw, Thomas, Rangamohan V. Eunni, in Rammohan Kasuganti. (2013). The mobile apps industry: A case study. *Journal of Business Cases and Applications* The mobile apps industry (september): 74–98.

Lethbridge, Timothy, in Robert Laganiere. (2005). *Object-Oriented Software Engineering*. 2 edition. London: McGraw-Hill Science/Engineering/Math.

Levy, Luis, in Jeannie Novak. (2009). *Game Development Essentials: Game QA & Testing*. 1 edition. Clifton Park, N.Y: Course Technology.

Levy, Yair, in Timothy J. Ellis. (2006). A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research. *Informing Science: The International Journal of an Emerging Transdiscipline* 9: 181–212. doi:10.28945/479.

Ma, Minhua, in Huiru Zheng. (2011). Virtual Reality and Serious Games in Healthcare. *Advanced Computational Intelligence Paradigms in Healthcare 6. Virtual Reality in Psychotherapy, Rehabilitation, and Assessment*, 169–92. Studies in Computational Intelligence. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-17824-5\_9.

Madani, Kaveh, Tyler W. Pierce, in Ali Mirchi. (2017). Serious games on environmental management. *Sustainable Cities and Society* 29 (februar): 1–11. doi:10.1016/j.scs.2016.11.007.

Manjuka, R, M Chakradhar Raju, in M Sai Chand. (2016). Software Engineering Challenges in Game Development. *International Journal of Research in Applied Science and Engineering Tehnology.* Dostopno prek https://www.ijraset.com/fileserve.php?FID=5768.

McAllister, Graham, in Gareth R. White. (2015). Video Game Development and User Experience. *Game User Experience Evaluation*, 11–35. Human–Computer Interaction Series. Springer, Cham. doi:10.1007/978-3-319-15985-0\_2.

McShaffry, Mike, in David Graham. (2012). *Game Coding Complete, Fourth Edition*. 4 edition. Boston, MA: Cengage Learning PTR.

Mitchell, Briar Lee. (2012). *Game Design Essentials*. John Wiley & Sons.

Muffatto, Moreno. (2006). *Open Source: A Multidisciplinary Approach*. London : Singapore ; Hackensack, NJ: Imperial College Press.

Musil, Juergen, Angelika Schweda, Dietmar Winkler, in Stefan Biffl. (2010). Improving Video Game Development: Facilitating Heterogeneous Team Collaboration through Flexible Software Processes. *Systems, Software and Services Process Improvement*, 83–94. Communications in Computer and Information Science. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-15666-3\_8.

Novak, Jeannie. (2012). *Game Development Essentials: An Introduction 3rd Edition*.

O’Hagan, Ann Osborne, Gerry Coleman, in Rory V. O’Connor. (2014). Software Development Processes for Games: A Systematic Literature Review. *Systems, Software and Services Process Improvement*, 182–93. Springer, Berlin, Heidelberg. doi:10.1007/978-3-662-43896-1\_16.

O’Hagan, Ann Osborne, in Rory V. O’Connor. (2015). Towards an Understanding of Game Software Development Processes: A Case Study. *Systems, Software and Services Process Improvement*, 3–16. Communications in Computer and Information Science. Springer, Cham. doi:10.1007/978-3-319-24647-5\_1.

Palmer, Stephen R., in John M. Felsing. (2002). *A Practical Guide to Feature-Driven Development*. 1 edition. Upper Saddle River, NJ: Prentice Hall.

Peters, Lawrence J. (2008). *Getting Results from Software Development Teams*. 1 edition. Redmond, WA: Microsoft Press.

Petrillo, Fabio, in Marcelo Pimenta. (2010). *Is agility out there? Agile practices in game development*. doi:10.1145/1878450.1878453.

Petrillo, Fabio, Marcelo Pimenta, Francisco M. Trindade, in Carlos Dietrich. (2008). Houston, we have a problem...: a survey of actual problems in computer games development. 707–11. doi:10.1145/1363686.1363854.

Petrillo, Fábio, Marcelo Pimenta, Francisco Trindade, in Carlos Dietrich. (2009). What Went Wrong? A Survey of Problems in Game Development. *Comput. Entertain.* 7 (1): 13:1–13:22. doi:10.1145/1486508.1486521.

Politowski, Cristiano, Daniel De Vargas, Lisandra Fontura, in Antonio Augutso Foletto. (2016). Software Engineering Processes in Game Development: a Survey about Brazilian Developers’ Experiences. Dostopno prek https://www.researchgate.net/publication/305653135\_Software\_Engineering\_Processes\_in\_Game\_Development\_a\_Survey\_about\_Brazilian\_Developers’\_Experiences.

Pressman, Roger S., in Bruce Maxim. (2014). *Software Engineering: A Practitioner’s Approach*. 8 edition. New York, NY: McGraw-Hill Education.

Ramadan, Rido, in Yani Widyani. (2013). Game development life cycle guidelines. 95–100. doi:10.1109/ICACSIS.2013.6761558.

Rolland, Colette. (1998). A Comprehensive View of Process Engineering. *Advanced Information Systems Engineering*, 1–24. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. doi:10.1007/BFb0054216.

Rucker, Rudy. (2002). *Software Engineering and Computer Games*. 1 edition. Harlow: Addison-Wesley.

Rumbaugh, James, Ivar Jacobson, in Grady Booch. (2004). *The Unified Modeling Language Reference Manual,*. 2 edition. Boston: Addison-Wesley Professional.

Ruonala, Henna-Riikka. (2016). Agile Game Development: A Systematic Literature Review.

Sabharwal, Sangeeta. (2009). *Software Engineering*. New Age International Pvt Ltd Publishers.

Salmon, Joshua P., Sarah M. Dolan, Richard S. Drake, Graham C. Wilson, Raymond M. Klein, in Gail A. Eskes. (2017). A survey of video game preferences in adults: Building better games for older adults. *Entertainment Computing* 21 (junij): 45–64. doi:10.1016/j.entcom.2017.04.006.

Schell, Jesse. (2008). *The Art of Game Design: A Book of Lenses*. 1 edition. Amsterdam ; Boston: CRC Press.

Sommerville, Ian. (2010). *Software Engineering*. 9 edition. Boston: Pearson.

Soomro, Sarmad, Wan Fatimah Wan Ahmad, in Suziah Sulaiman. (2013). Evaluation of Mobile Games Using Playability Heuristics. *Advances in Visual Informatics*, 264–74. Lecture Notes in Computer Science. Springer, Cham. doi:10.1007/978-3-319-02958-0\_25.

Stacey, Patrick, in Joe Nandhakumar. (2008). Opening up to Agile Games Development. *Communications of the ACM* 51 (12): 143. doi:10.1145/1409360.1409387.

Sylvester, Tynan. (2013). *Designing Games: A Guide to Engineering Experiences*. 1 edition. Sebastopol, CA: O’Reilly Media.

Tsui, Frank, Orlando Karam, in Barbara Bernal. (2013). *Essentials of Software Engineering*. Jones & Bartlett Publishers.

Tutorialspoint. (b.d.). Dostopno prek https://www.tutorialspoint.com/sdlc/sdlc\_v\_model.htm.

Ulbin, Martin. (2017). *Razvoj igre za pomoč otrokom s prekomerno težo*  (diplomska naloga). Dostopno prek https://dk.um.si/IzpisGradiva.php?id=66054&lang=slv.

Unger, Kimberly, in Jeannie Novak. (2011). *Game Development Essentials: Mobile Game Development*. 1 edition. Clifton Park, NY: Cengage Learning.

Wilson Brotto Furtado, André. (2012). *Domain-Specific Game Development* (doktorska disertacija). University of Pernambuco.

Yilmaz, Murat, in Rory O’Connor. (2016). *A Scrumban Integrated Gamification Approach To Guide Software Process Improvement: A Turkish Case Study*. Let. 23. doi:10.17559/TV-20140922220409.

Zagal, José P., in Roger Altizer. (2015). Placeholder Content in Game Development: Benefits and Challenges. *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, 745–750. CHI PLAY ’15. New York, NY, USA: ACM. doi:10.1145/2793107.2810319.

PRILOGE

PRILOGA A: ŠTEVILO TIPOV CITATOV NA LETO IZDAJE RAZISKAVE



Vir: (Aleem, Capretz, in Ahmed, 2016b, 14)

PRILOGA B: ŽIVLJENJSKI CIKEL METODOLOGIJE ZA RESNE IGRE GAMED

Vir: (Aslan in Balci, 2015, 309)

PRILOGA C: SPIRALNI MODEL ZA PODPORO OBLIKOVANJU IGRE PRI METODOLOGIJI GAMED



Vir: (Aslan in Balci, 2015, 313)

PRILOGA D: PREDLAGANI ŽIVLJENJSKI CIKEL RAZVOJA ZA VIDEO IGRE PO WIDYANI



Vir: (Ramadan in Widyani, 2013, 98)

PRILOGA E: PRIKAZ ARTEFAKTOV RAZVOJA VIDEO IGER PO AVTORJIH

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AVTOR | ARTEFAKTI |  |  |  |  |  |  |  |  |
| Novak (2012) | KONCEPT | PREDLOG IGRE | NAČRT IGRE | VODNIK UMETNIŠKEGA SLOGA | TEHNIČNI DOKUMENT | PROJEKTNI NAČRT | NAČRT TESTIRANJA |  |  |
| Bates (2004) | NAČRT IGRE | OPIS OBLIKOVANJA | SPECIFIKACIJA POTREB | PLAN KONFIGURACIJE | NAČRT INTEGRACIJE TESTIRANJA | NAČRT TESTIRANJA | UPORABNIŠKI PRIROČNIK |  |  |
| Rucker (2002) | SPECIFIKACIJA | NAČRT IGRE | ČASOVNI NAČRT | NAČRT OBLIKOVANJA | DOKUMENTACIJA | UPORABNIŠKI PRIROČNIK |  |  |  |
| Schell (2008) | OBLIKOVANJE (PREGLED OBLIKOVANJA, PODROBNI NAČRT IGRE, PREGLED ZGODBE) | INŽENIRING (TEHNIČNI DOKUMENT, PREGLED DELOVNEGA TOKA, OMEJITVE) | UMETNOST (UMETNIKOVA BIBLIJA, PREGLED KONCEPTOV UMETNIN) | UPRAVLJANJE(PRORAČUN, ČASOVNI NAČRT) | PISANJE (ZGODBA NARACIJA. UPORABNIŠKI PRIROČNIK) | IGRALCI (IGRALNI VODNIKI) |  |  |  |
| Richard Rouse III  (2004) | KONCEPT (PITCH, PREDLOG) | KONKURENČNA ANALIZA | NAČRT OBLIKOVANJA | DIAGRAM POTEKA | ZGODBA, NARACIJA | UMETNIKOVA BIBLIJA | TEHNIČNI DOKUMENT | ČASOVNI, POSLOVNI IN MARKETINŠKI DOKUMENTI |  |
| Adams (2013) | VIŠJI KONCEPT, ANALIZA IGRE | ANALIZA IGRE | OBLIKOVANJE OSREDNJEGA IGRALCA | OBLIKOVANJE SVETA | OBLIKOVANJE UPORABNIŠKEGA VMESNIKA | DIAGRAM POTEKA | ZGODBA IN NAPREDOVANJE PO STOPNJAH | TEKST IN AVDIO | SCENARIJ IGRE |
| Bartle (2003) | DOKUMENT VIZUALIZACIJE | NAČRT OBLIKOVANJA | TEHNIČNI NAČRT | UMETNIŠKA BIBLIJA | UPRAVLJANJE PRODUKCIJE | PROTOTIP |  |  |  |

PRILOGA F: PRIKAZ FAZ RAZVOJA PO AVTORJIH

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AVTOR | FAZE |  |  |  |  |  |  |  |  |  |  |
| Novak (2012) | KONCEPT | PRED PRODUKCIJA | PROTOTIP | PRODUKCIJA | ALFA | BETA | GOLD | POST PRODUKCIJA |  |  |  |
| Unger in Novak (2011) | PRED PRODUKCIJA | PRODUKCIJA | ALFA & BETA | GOLD | POST PRODUKCIJA |  |  |  |  |  |  |
| Ramadan in Widyani (2013) lasten podroben | INICIACIJA | PRED PRODUKCIJA | PRODUKCIJA | TESTIRANJE | BETA | IZDAJA |  |  |  |  |  |
| Bates (2004) | RAZVOJ KONCEPTA | PRED PRODUKCIJA | RAZVOJ | ALFA | BETA | PREKINITEV KODIRANJA | IZDAJA | POPRAVKI | POSODOBITVE |  |  |
| Aslan in Balci (2015) | FORMULACIJA PROBLEMA | IZDELAVA IDEJE | NAČRTOVANJE IGRE | RAZVOJ ZAHTEV | ARHITEKTURA | NAČRTOVANJE PROGRAMSKE OPREME | PROGRAMIRANJE | INTEGRACIJA | IZDAJA | UČENJE | POVRATNA INFORMACIJA |
| Bartle (2004) | PRED PRODUKCIJA | PRODUKCIJA | IZDAJA | PODPORA |  |  |  |  |  |  |  |
| Fulleton (2014) | KONCEPT | PRED PRODUKCIJA | PRODUKCIJA | ZAGOTAVLJANJE KVALITETE | VZDRŽEVANJE |  |  |  |  |  |  |
| Adams (2013) | FAZA KONCEPTA | FAZA ELABORACIJE | FAZA TUNINGA |  |  |  |  |  |  |  |  |
| Mitchell (2012) | PRED PRODUKCIJA | PRODUKCIJA | ALFA | BETA | GOLD | POST PRODUCIJA |  |  |  |  |  |
| McAllister in White (2015) | KONCEPT | PROTOTIPIRANJE | PRED PRODUKCIJA | PRODUKCIJA | ALFA | BETA | GOLD | PO IZDAJI |  |  |  |
| McAllister in White (2015) o Black Rock studiju | PROTOTIPIRANJE | PRED PRODUKCIJA | PRODUKCIJA | ALFA | BETA | SPREJETJE | IZDAJA | TESTIRANJE | MANUFAKTURA |  |  |
| Kanode in Hadadd (2009) | PRED PRODUKCIJA | PRODUKCIJA | TESTIRANJE |  |  |  |  |  |  |  |  |

PRILOGA G: PRIMERJAVA TEMELJINIH KONCEPTOV METOD INŽENIRINGA PROGRAMSKE OPREME



Vir: (Engels in Sauer, 2010, 418)

PRILOGA H: OSNUTEK DOKUMENTA OBLIKOVANJA IGRE GDD PO BATES (2004)

[1. GAME NAME 1](#_Toc509342358)

[a. Copyright Information 1](#_Toc509342359)

[2. TABLE OF CONTENTS 1](#_Toc509342360)

[3. SECTION I: PROJECT OVERVIEW 1](#_Toc509342361)

[a. Team Personnel (with contact information for each individual) 1](#_Toc509342362)

[i. Production Team 1](#_Toc509342363)

[ii. Design Team 1](#_Toc509342364)

[iii. Programming Team 1](#_Toc509342365)

[iv. Art Team 1](#_Toc509342366)

[vi. External Contractors 1](#_Toc509342367)

[b. Executive Summary 1](#_Toc509342368)

[i. High Concept 1](#_Toc509342369)

[ii. The Hook 1](#_Toc509342370)

[iii. Story Synopsis and Setting 1](#_Toc509342371)

[iv. Genre & Scope (such as number of missions or levels) 1](#_Toc509342372)

[v. Visual Style (2D? 3D? Isometric? etc.) 1](#_Toc509342373)

[vi. Engine (and editor?) 1](#_Toc509342374)

[c. Core Gameplay (What does the player *do*?) 1](#_Toc509342375)

[i. Single-player 1](#_Toc509342376)

[ii. Co-op? 1](#_Toc509342377)

[iii. Multiplayer? 1](#_Toc509342378)

[d. Game Features 1](#_Toc509342379)

[i. Gameplay innovations 1](#_Toc509342380)

[ii. Advances in AI 1](#_Toc509342381)

[iii. Artistic techniques and achievements 1](#_Toc509342382)

[iv. License tie-ins (if applicable) 1](#_Toc509342383)

[v. Other features that will make this game better than others like it on the market 1](#_Toc509342384)

[e. Project Scope 2](#_Toc509342385)

[i. Number of distinct locations 2](#_Toc509342386)

[ii. Number of levels/missions 2](#_Toc509342387)

[iii. Number of NPCs 2](#_Toc509342388)

[iv. Number of weapons 2](#_Toc509342389)

[v. Number of vehicles 2](#_Toc509342390)

[vi. Etc. 2](#_Toc509342391)

[f. Target Audience 2](#_Toc509342392)

[g. Delivery Platform(s) 2](#_Toc509342393)

[4. SECTION II: STORY, SETTING, AND CHARACTER 3](#_Toc509342394)

[a. Story 3](#_Toc509342395)

[i. Back story 3](#_Toc509342396)

[ii. In-game story (What happens during the game) 3](#_Toc509342397)

[b. Environments 3](#_Toc509342398)

[i. Area #1 3](#_Toc509342399)

[1. General description 3](#_Toc509342400)

[2. Physical characteristics 3](#_Toc509342401)

[3. List of levels that take place in this area 3](#_Toc509342402)

[ii. Area #2 3](#_Toc509342403)

[iii. Etc. 3](#_Toc509342404)

[c. Characters 3](#_Toc509342405)

[i. Player Character(s) 3](#_Toc509342406)

[1. Personality 3](#_Toc509342407)

[2. Back story 3](#_Toc509342408)

[3. “Look” 3](#_Toc509342409)

[4. Special abilities 3](#_Toc509342410)

[a. Ability #1 3](#_Toc509342411)

[i. When it’s acquired 3](#_Toc509342412)

[ii. How the player invokes it 3](#_Toc509342413)

[iii. Effect it has on the world 3](#_Toc509342414)

[iv. Graphic effect that accompanies it 3](#_Toc509342415)

[5. Weapon set 3](#_Toc509342416)

[6. Regular animations 3](#_Toc509342417)

[a. Walk, run, climb, roll, swim, crouch, crawl, idle, etc. 3](#_Toc509342418)

[7. Situation-specific animations 3](#_Toc509342419)

[8. Statistics (if applicable) 3](#_Toc509342420)

[ii. Allies 3](#_Toc509342421)

[1. Ally #1 4](#_Toc509342422)

[a. Personality 4](#_Toc509342423)

[b. Relationship to player character 4](#_Toc509342424)

[c. Back story 4](#_Toc509342425)

[d. “Look” 4](#_Toc509342426)

[e. Special abilities 4](#_Toc509342427)

[f. Weapon set 4](#_Toc509342428)

[g. Regular animations 4](#_Toc509342429)

[h. Situation-specific animations 4](#_Toc509342430)

[i. Statistics 4](#_Toc509342431)

[2. Ally #2 4](#_Toc509342432)

[3. Etc. 4](#_Toc509342433)

[iii. Bad Guys 4](#_Toc509342434)

[1. Ultimate bad guy 4](#_Toc509342435)

[a. Personality 4](#_Toc509342436)

[b. Relationship to player character 4](#_Toc509342437)

[c. Back story 4](#_Toc509342438)

[d. “Look” 4](#_Toc509342439)

[e. Special abilities 4](#_Toc509342440)

[f. Weapon set 4](#_Toc509342441)

[g. Regular animations 4](#_Toc509342442)

[h. Situation-specific animations 4](#_Toc509342443)

[i. Statistics 4](#_Toc509342444)

[2. Sub bosses 4](#_Toc509342445)

[3. Grunts 4](#_Toc509342446)

[iv. Neutrals 4](#_Toc509342447)

[1. World NPCs 4](#_Toc509342448)

[a. NPC#1 4](#_Toc509342449)

[i. Attitude towards player character 5](#_Toc509342450)

[ii. Function in the game 5](#_Toc509342451)

[iii. Animation set 5](#_Toc509342452)

[b. NPC#2 5](#_Toc509342453)

[c. Etc. 5](#_Toc509342454)

[d. Level Flow (A flowchart that summarizes the action of each level, and the cutscenes or mission briefings (if any) that take place between them) 5](#_Toc509342455)

[5. SECTION III: COMBAT 6](#_Toc509342456)

[a. Weapons 6](#_Toc509342457)

[i. Weapon #1 6](#_Toc509342458)

[1. General description and most effective use 6](#_Toc509342459)

[2. When it is first acquired 6](#_Toc509342460)

[3. Art (if available) 6](#_Toc509342461)

[4. Statistics (for both primary and secondary fire) 6](#_Toc509342462)

[a. Type of ammunition 6](#_Toc509342463)

[b. Shots per clip 6](#_Toc509342464)

[c. Fire rate 6](#_Toc509342465)

[d. Reload rate 6](#_Toc509342466)

[e. Damage inflicted 6](#_Toc509342467)

[f. Range 6](#_Toc509342468)

[ii. Weapon #2 6](#_Toc509342469)

[iii. Etc. 6](#_Toc509342470)

[b. Spells 6](#_Toc509342471)

[i. Spell #1 6](#_Toc509342472)

[1. Description 6](#_Toc509342473)

[2. When it is first acquired 6](#_Toc509342474)

[3. How the player invokes it 6](#_Toc509342475)

[4. Statistics 6](#_Toc509342476)

[a. Range 6](#_Toc509342477)

[b. “Refire rate” 6](#_Toc509342478)

[c. Damage 6](#_Toc509342479)

[d. Area of effect 6](#_Toc509342480)

[ii. Spell #2 6](#_Toc509342481)

[iii. Etc. 6](#_Toc509342482)

[c. Inventory Items/Gadgets 6](#_Toc509342483)

[i. Item #1 7](#_Toc509342484)

[1. Brief physical description of the object 7](#_Toc509342485)

[2. When it is first acquired 7](#_Toc509342486)

[3. What it does 7](#_Toc509342487)

[4. Art (if available) 7](#_Toc509342488)

[5. How the player equips it 7](#_Toc509342489)

[6. Statistics 7](#_Toc509342490)

[ii. Item #2 7](#_Toc509342491)

[iii. Etc. 7](#_Toc509342492)

[d. Powerups 7](#_Toc509342493)

[i. Powerup #1 7](#_Toc509342494)

[1. Brief physical description of how the object is represented in the world 7](#_Toc509342495)

[2. When it is first acquired 7](#_Toc509342496)

[3. Art (if available) 7](#_Toc509342497)

[4. What it does 7](#_Toc509342498)

[5. Statistics 7](#_Toc509342499)

[a. Effect 7](#_Toc509342500)

[b. Duration 7](#_Toc509342501)

[ii. Powerup #2 7](#_Toc509342502)

[iii. Etc. 7](#_Toc509342503)

[e. Melee (hand-to-hand) combat (if applicable) 7](#_Toc509342504)

[i. Attacks 7](#_Toc509342505)

[ii. Defensive moves 7](#_Toc509342506)

[iii. Combos 7](#_Toc509342507)

[f. Vehicles (if applicable) 7](#_Toc509342508)

[i. Capacity 7](#_Toc509342509)

[ii. Speed 7](#_Toc509342510)

[iii. Armor 7](#_Toc509342511)

[iv. Weaponry 8](#_Toc509342512)

[v. Combat statistics 8](#_Toc509342513)

[vi. Etc. 8](#_Toc509342514)

[6. SECTION IV: CONTROLS 9](#_Toc509342515)

[a. PC Keyboard/Mouse Commands 9](#_Toc509342516)

[i. Default keys for movement controls 9](#_Toc509342517)

[1. Move forward 9](#_Toc509342518)

[2. Move backward 9](#_Toc509342519)

[3. Strafe left 9](#_Toc509342520)

[4. Strafe right 9](#_Toc509342521)

[5. Jump 9](#_Toc509342522)

[6. Etc. 9](#_Toc509342523)

[ii. Default keys for using weapons 9](#_Toc509342524)

[1. Primary fire 9](#_Toc509342525)

[2. Alt-fire 9](#_Toc509342526)

[3. Reload 9](#_Toc509342527)

[4. Previous weapon 9](#_Toc509342528)

[5. Next weapon 9](#_Toc509342529)

[6. Etc. 9](#_Toc509342530)

[iii. Inventory access and manipulation 9](#_Toc509342531)

[iv. Menu access 9](#_Toc509342532)

[b. Console Platform #1 9](#_Toc509342533)

[i. A picture of the controller explaining what each button does 9](#_Toc509342534)

[ii. Movement controls 9](#_Toc509342535)

[iii. Weapon controls 9](#_Toc509342536)

[iv. Action controls 9](#_Toc509342537)

[v. Combos 9](#_Toc509342538)

[vi. Force-feedback options 9](#_Toc509342539)

[c. Console Platform #2 9](#_Toc509342540)

[d. Etc. 9](#_Toc509342541)

[7. SECTION V: INTERFACE 10](#_Toc509342542)

[a. The Camera 10](#_Toc509342543)

[i. Standard view 10](#_Toc509342544)

[ii. Alternate views 10](#_Toc509342545)

[iii. Player-controllable options 10](#_Toc509342546)

[b. HUD 10](#_Toc509342547)

[i. Worldview (what the player sees) 10](#_Toc509342548)

[ii. Status information 10](#_Toc509342549)

[1. Health 10](#_Toc509342550)

[2. Energy 10](#_Toc509342551)

[3. Armor 10](#_Toc509342552)

[4. Weapon equipped 10](#_Toc509342553)

[5. Ammo remaining 10](#_Toc509342554)

[6. Mission objectives? 10](#_Toc509342555)

[iii. Crosshairs (targeting reticule) 10](#_Toc509342556)

[iv. Radar or proximity map? 10](#_Toc509342557)

[c. Menus 10](#_Toc509342558)

[i. Game screen flow diagrams (schematic of how all the game’s various screens are accessed) 10](#_Toc509342559)

[ii. Start Menu 10](#_Toc509342560)

[1. Install 10](#_Toc509342561)

[2. Play game 10](#_Toc509342562)

[3. Explore CD (bonus features) 10](#_Toc509342563)

[4. Uninstall 10](#_Toc509342564)

[5. Quit 10](#_Toc509342565)

[iii. Main Menu 10](#_Toc509342566)

[1. Single-Player 10](#_Toc509342567)

[a. Load game 10](#_Toc509342568)

[b. Save game 10](#_Toc509342569)

[c. Play training level 11](#_Toc509342570)

[d. Set difficulty level 11](#_Toc509342571)

[2. Co-op 11](#_Toc509342572)

[3. Multiplayer 11](#_Toc509342573)

[a. Connection instructions 11](#_Toc509342574)

[b. Character/team selection 11](#_Toc509342575)

[iv. Game Menus 11](#_Toc509342576)

[1. Remap player controls 11](#_Toc509342577)

[2. Display (video) 11](#_Toc509342578)

[3. Audio 11](#_Toc509342579)

[4. Music 11](#_Toc509342580)

[5. Map 11](#_Toc509342581)

[6. Advanced 11](#_Toc509342582)

[7. Help screen 11](#_Toc509342583)

[8. Quit 11](#_Toc509342584)

[v. Inventory Menu 11](#_Toc509342585)

[vi. Credits 11](#_Toc509342586)

[8. SECTION VI: ARTIFICIAL INTELLIGENCE (AI) 12](#_Toc509342587)

[a. NPC #1 12](#_Toc509342588)

[i. Statistics 12](#_Toc509342589)

[1. Field of view 12](#_Toc509342590)

[2. Range of view 12](#_Toc509342591)

[3. Etc. 12](#_Toc509342592)

[ii. Internal states & the triggers that change them 12](#_Toc509342593)

[1. Idle 12](#_Toc509342594)

[2. Guarding an area 12](#_Toc509342595)

[3. Patrol 12](#_Toc509342596)

[4. Follow 12](#_Toc509342597)

[5. Search 12](#_Toc509342598)

[6. Etc. 12](#_Toc509342599)

[iii. Movement 12](#_Toc509342600)

[1. Pathing 12](#_Toc509342601)

[iv. Combat decisions 12](#_Toc509342602)

[1. Friend/foe recognition 12](#_Toc509342603)

[2. Targeting decisions 12](#_Toc509342604)

[3. Attack with ranged weapon 12](#_Toc509342605)

[4. Attack with melee weapon 12](#_Toc509342606)

[5. Take cover 12](#_Toc509342607)

[6. Team-based decisions 12](#_Toc509342608)

[7. Etc. 12](#_Toc509342609)

[b. NPC #2 12](#_Toc509342610)

[c. Etc. 12](#_Toc509342611)

[9. SECTION VII: DETAILED LEVEL/MISSION DESCRIPTIONS 13](#_Toc509342612)

[a. Level #1 13](#_Toc509342613)

[i. Synopsis 13](#_Toc509342614)

[ii. Introductory material (Cutscene? Mission briefing?) 13](#_Toc509342615)

[iii. Mission objectives (player goals) 13](#_Toc509342616)

[iv. Physical description 13](#_Toc509342617)

[v. Map 13](#_Toc509342618)

[vi. Enemy types encountered in-level 13](#_Toc509342619)

[vii. Weapons/powerups available 13](#_Toc509342620)

[viii. Level walkthrough, including scripted sequences and non-interactive scenes. This should also include any puzzles the player must solve, as well as the solutions to those puzzles. 13](#_Toc509342621)

[ix. Closing material (Cutscene? Debriefing? Statistics menu?) 13](#_Toc509342622)

[b. Level #2 13](#_Toc509342623)

[c. Etc. 13](#_Toc509342624)

[10. SECTION VIII: CUTSCENES 13](#_Toc509342625)

[a. Cutscene #1 13](#_Toc509342626)

[i. List of actors 13](#_Toc509342627)

[ii. Description of setting 13](#_Toc509342628)

[iii. Storyboard thumbnails 13](#_Toc509342629)

[iv. Script. This should be done in screenplay format, as if you were writing a 13](#_Toc509342630)

[movie. Include the action, suggested camera angles, location descriptions, 13](#_Toc509342631)

[etc. You must also include all lines of dialogue that are to be recorded or 13](#_Toc509342632)

[displayed on the screen. Refer to any of the screenplay books in Appendix 13](#_Toc509342633)

[B for samples of this format. 13](#_Toc509342634)

[b. Cutscene #2 13](#_Toc509342635)

[c. Etc. 13](#_Toc509342636)

[11. SECTION IX: SCORING, CHEATS, EASTER EGGS, & BONUSES 14](#_Toc509342637)

[a. Score 14](#_Toc509342638)

[i. How score is tracked 14](#_Toc509342639)

[ii. How score is communicated to the player 14](#_Toc509342640)

[b. Cheats (God mode, all weapons, etc.) 14](#_Toc509342641)

[i. Cheat #1 14](#_Toc509342642)

[1. What it does 14](#_Toc509342643)

[2. How it’s activated by the developer 14](#_Toc509342644)

[3. How it’s unlocked by the player 14](#_Toc509342645)

[ii. Cheat #2 14](#_Toc509342646)

[iii. Etc. 14](#_Toc509342647)

[c. Easter Eggs/Bonus Material 14](#_Toc509342648)

[i. Easter Egg #1 14](#_Toc509342649)

[1. What it is 14](#_Toc509342650)

[2. How it’s activated/unlocked 14](#_Toc509342651)

[ii. Easter Egg #2 14](#_Toc509342652)

[iii. Etc. 14](#_Toc509342653)

[12. SECTION X: GAME MODES 15](#_Toc509342654)

[a. Single-player 15](#_Toc509342655)

[b Split-screen/co-op (if applicable) 15](#_Toc509342656)

[c. Multiplayer game types (if applicable) 15](#_Toc509342657)

[i. Gametype #1 (such as “Capture the Flag”) 15](#_Toc509342658)

[1. Description of gameplay 15](#_Toc509342659)

[2. Min/max # of Players 15](#_Toc509342660)

[3. Rules 15](#_Toc509342661)

[4. Respawning 15](#_Toc509342662)

[a. Delay 15](#_Toc509342663)

[b. Respawn locations 15](#_Toc509342664)

[c. Default weapons 15](#_Toc509342665)

[5. Victory conditions 15](#_Toc509342666)

[6. Scoring 15](#_Toc509342667)

[7. Maps 15](#_Toc509342668)

[ii. Gametype #2 15](#_Toc509342669)

[iii. Etc. 15](#_Toc509342670)

[13. SECTION XI: ASSET LIST 16](#_Toc509342671)

[a. Art 16](#_Toc509342672)

[i. Model & Texture List 16](#_Toc509342673)

[1. Characters 16](#_Toc509342674)

[a. Player character 16](#_Toc509342675)

[i. Undamaged 16](#_Toc509342676)

[ii. Damaged 16](#_Toc509342677)

[b. Allies 16](#_Toc509342678)

[c. Bad guys 16](#_Toc509342679)

[d. Neutrals 16](#_Toc509342680)

[2. Weapons 16](#_Toc509342681)

[a. Weapon #1 16](#_Toc509342682)

[b. Weapon #2 16](#_Toc509342683)

[c. Etc. 16](#_Toc509342684)

[3. Equipment/Gadgets 16](#_Toc509342685)

[a. Item #1 16](#_Toc509342686)

[b. Item #2 16](#_Toc509342687)

[c. Etc. 16](#_Toc509342688)

[4. Environmental Objects 16](#_Toc509342689)

[a. Object #1 16](#_Toc509342690)

[b. Object #2 16](#_Toc509342691)

[c. Etc. 16](#_Toc509342692)

[ii. Animation list 16](#_Toc509342693)

[1. Characters 16](#_Toc509342694)

[a. Character #1 16](#_Toc509342695)

[i. Move #1 16](#_Toc509342696)

[ii. Move #2 16](#_Toc509342697)

[iii. Etc. 16](#_Toc509342698)

[b. Character #2 17](#_Toc509342699)

[c. Etc. 17](#_Toc509342700)

[2. Weapons 17](#_Toc509342701)

[a. Weapon #1 17](#_Toc509342702)

[i. Firing animation 17](#_Toc509342703)

[ii. Reload animation 17](#_Toc509342704)

[iii. Projectile in flight animation (if appropriate) 17](#_Toc509342705)

[3. Destructible or animated objects in the world 17](#_Toc509342706)

[a. Object #1 17](#_Toc509342707)

[b. Object #2 17](#_Toc509342708)

[c. Etc. 17](#_Toc509342709)

[iii. Effects list 17](#_Toc509342710)

[1. Weapon effects list 17](#_Toc509342711)

[a. Firing effects 17](#_Toc509342712)

[b. Hit effects 17](#_Toc509342713)

[c. Etc. 17](#_Toc509342714)

[2. Environmental effects 17](#_Toc509342715)

[a. Decals 17](#_Toc509342716)

[b. Smoke 17](#_Toc509342717)

[c. Sparks 17](#_Toc509342718)

[d. Fire 17](#_Toc509342719)

[e. Explosions 17](#_Toc509342720)

[f. Etc. 17](#_Toc509342721)

[iv. Interface Art List 17](#_Toc509342722)

[1. Icons 17](#_Toc509342723)

[2. Buttons 17](#_Toc509342724)

[3. Menus 17](#_Toc509342725)

[4. Windows 17](#_Toc509342726)

[5. Etc. 18](#_Toc509342727)

[b. Sound 18](#_Toc509342728)

[i. Environmental Sounds 18](#_Toc509342729)

[1. Walking/running sounds on different surfaces 18](#_Toc509342730)

[2. Foley sounds of character actions within the game 18](#_Toc509342731)

[3. Explosions 18](#_Toc509342732)

[4. Doors opening and closing 18](#_Toc509342733)

[5. Etc. 18](#_Toc509342734)

[ii. Weapon Sounds 18](#_Toc509342735)

[1. Weapon #1 18](#_Toc509342736)

[a. Firing sound 18](#_Toc509342737)

[b. Hit sound 18](#_Toc509342738)

[c. Reload sound 18](#_Toc509342739)

[2. Weapon #2 18](#_Toc509342740)

[3. Etc. 18](#_Toc509342741)

[iii. Interface Sounds 18](#_Toc509342742)

[1. Various clicks, beeps, etc., as the player maneuvers through the menus 18](#_Toc509342743)

[2. Alert/acknowledgment sounds as the player picks up objects or his game state changes 18](#_Toc509342744)

[c. Music 18](#_Toc509342745)

[i. Ambient 18](#_Toc509342746)

[1. Loop #1 + duration 18](#_Toc509342747)

[2. Loop #2 18](#_Toc509342748)

[3. Etc. 18](#_Toc509342749)

[ii. “Action” 18](#_Toc509342750)

[1. Loop #1 + duration 18](#_Toc509342751)

[2. Loop #2 18](#_Toc509342752)

[3. Etc. 18](#_Toc509342753)

[iii. “Victory” loops 18](#_Toc509342754)

[iv. “Defeat” loops 19](#_Toc509342755)

[v. Cutscene music 19](#_Toc509342756)

[1. Piece #1 19](#_Toc509342757)

[a. General description of mood and accompanying action 19](#_Toc509342758)

[b. Duration 19](#_Toc509342759)

[2. Piece #2 19](#_Toc509342760)

[3. Etc. 19](#_Toc509342761)

[d. Voice 19](#_Toc509342762)

[i. Actor #1 lines 19](#_Toc509342763)

[1. Line #1. Each line in the game must have a unique identifying file 19](#_Toc509342764)

[name. This will help both the recording process and localization. Don’t forget to include various screams, yells, grunts, laughs, and other “non-word” lines. 19](#_Toc509342765)

[2. Line #2 19](#_Toc509342766)

[3. Etc. 19](#_Toc509342767)

[ii. Actor #2 lines 19](#_Toc509342768)

[iii. Etc. 19](#_Toc509342769)

[14. SECTION XII: LOCALIZATION PLAN 19](#_Toc509342770)

[a. Languages with full text and voice localization 19](#_Toc509342771)

[b. Languages with text localization only 19](#_Toc509342772)

[c. Text to be localized 19](#_Toc509342773)

[i. In-game text 19](#_Toc509342774)

[ii. Game interface text 19](#_Toc509342775)

[d. Voice to be localized 19](#_Toc509342776)

[i. (See “Voice” section of asset list above) 19](#_Toc509342777)

[15. SECTION XIII: MAJOR EVENT PLANNING 20](#_Toc509342778)

[a. Trade Shows 20](#_Toc509342779)

[i. Trade Show #1 20](#_Toc509342780)

[1. Date 20](#_Toc509342781)

[2. Materials needed for event 20](#_Toc509342782)

[3. Demo description and specifications 20](#_Toc509342783)

[ii. Trade Show #2 20](#_Toc509342784)

[iii. Etc. 20](#_Toc509342785)

[b. Special Publicity Events 20](#_Toc509342786)

[i. Event #1 (such as “Editors Day” to show off game) 20](#_Toc509342787)

[1. Date 20](#_Toc509342788)

[2. Description of event 20](#_Toc509342789)

[3. Materials needed for event 20](#_Toc509342790)

[4. Demo description and specifications 20](#_Toc509342791)

[ii. Event #2 20](#_Toc509342792)

[iii. Etc. 20](#_Toc509342793)

[c. PR/Marketing Support 20](#_Toc509342794)

[i. Date when concept art will be available 20](#_Toc509342795)

[ii. Date when first screenshots will be available 20](#_Toc509342796)

[iii. Plan for creating additional screenshots throughout project 20](#_Toc509342797)

[iv. Plan for making team available for interviews 20](#_Toc509342798)

[v. Etc. 20](#_Toc509342799)

[d. Sales Team Support 20](#_Toc509342800)

[i. Projected date of first “sell-sheet” 20](#_Toc509342801)

[ii. Demo loop for retail outlets 20](#_Toc509342802)

[iii. Other materials 20](#_Toc509342803)

[iv. Etc. 20](#_Toc509342804)

[e. Prerelease Demo 20](#_Toc509342805)

[i. Date 20](#_Toc509342806)

[ii. Scope 21](#_Toc509342807)

[iii. Content 21](#_Toc509342808)

[16. SECTION XIV: TECHNICAL SUMMARY 21](#_Toc509342809)

[a. Single-Player 21](#_Toc509342810)

[i. PC 21](#_Toc509342811)

[1. Minimum system requirements 21](#_Toc509342812)

[2. Recommended system requirements 21](#_Toc509342813)

[3. Number of characters viewable at once 21](#_Toc509342814)

[4. Max # polys per character 21](#_Toc509342815)

[5. Max # polys per level 21](#_Toc509342816)

[ii. Console Platform #1 21](#_Toc509342817)

[iii. Etc. 21](#_Toc509342818)

[b. Multiplayer 21](#_Toc509342819)

[i. Type of connectivity (Splitscreen? LAN? Online?) 21](#_Toc509342820)

[ii. Max # simultaneous players 21](#_Toc509342821)

[iii. Client-server? Peer-to-peer? 21](#_Toc509342822)

[iv. Etc. 21](#_Toc509342823)

[17. SECTION XV: MISCELLANEOUS 21](#_Toc509342824)

[a. Acronyms used in this document 21](#_Toc509342825)

[b. Definition of terms 21](#_Toc509342826)

[18. SECTION XVI: REFERENCES 22](#_Toc509342827)

[a. Games 22](#_Toc509342828)

[b. Movies 22](#_Toc509342829)

[c. Books 22](#_Toc509342830)

[d. Art 22](#_Toc509342831)

Vir: (Bates, 2004, 276–91)

PRILOGA I: PRAKSE RAZLIČNIH AGILNIH PROCESNIH MODELOV IN NJIHOVE RAZLAGE

|  |  |  |
| --- | --- | --- |
| MODEL | PRAKSA | NAMEN |
| XP | 40 urni teden | Teden je sestavljen iz 40 delavnih ur. Niti dva delavna tedna, ki presežeta te ure nista dovoljena. Če se to zgodi se ta pojav rešuje kot problem. |
| Scrum | Sprint | Je procedura produkcije novega inkrementa produkta v času imenovanem Sprint, ki navadno traja 30 koledarskih dni. |
| Crystal | Tehnika optimizacije metodologije | Namen prakse je izdelati specifično Crystal metodologijo z uporabo projektnih intervjujev in delavnic. Po vsakem inkrementu se lahko uporabi spoznanje in uporabi v naslednjem za izboljšanje procesa. |
| FDD | Razvoj po funkcionalnostih | Razvoj in spremljanje napredka na podlagi seznama razdeljenih in esencialnih naročnikovih funkcionalnosti. |
| RUP | Vizualno modeliranje sistema | Zgrajeni so modeli sistema, saj so ti kompleksni za razumevanje. Pogosto se uporablja UML. |
| DSDM | Integracija testiranja skozi celoten življenjski cikel | Vsaka komponenta sistema se testira, ko se izvede njegov razvoj. Testiranje se izvaja inkrementalno. Zaradi evolucijskega razvoja je značilno regresijsko testiranje. |

Vir:(Abrahamsson in drugi, 2017, 24–68)

PRILOGA J: KRITERIJI ZA IZBIRO ORODIJ UPRAVLJANJA PROJEKTOV RAZVOJA PROGRAMSKE OPREME



Vir: (Ahmad in Laplante, 2006, 78)

PRILOGA K: IDENTIFICIRANIH 5 DIMENZIJ Z 18 FAKTORJI VPLIVA NA DELOVANJE IGRE IN RAZVOJNI PROCES



Vir: (Aleem, Capretz, in Ahmed, 2016a, 62)

PRILOGA L: OCENE PRAGA ZA DOSEGANJE POSAMEZNE STOPNJE ZRELOSTI

|  |  |  |
| --- | --- | --- |
| Game maturity level | Total questions | Passing threshold (80%) |
| Ad-Hoc | 31 | 25 |
| Opportunistic | 51 | 41 |
| Consistent | 54 | 43 |
| Organized | 54 | 43 |
| Optimized | 43 | 36 |

Vir: (Aleem, Capretz, in Ahmed, 2016a, 68)

PRILOGA M: VPRAŠALNIK ZA OCENITEV OPTIMIZIRANE ZRELOSTI RAZVOJNEGA PROCESA VIDEO IGER

**GDPA 5.1 GDD Management**

S.5.1.1 Defined game design guidelines and concepts are followed

for all new game development projects.

S.5.1.2 The GDD is well understandable by all stakeholders.

S.5.1.3 The GDD is available to all development team members

at the beginning of the production phase.

S.5.1.4 A log is maintained to record development team

members’ complaints regarding GDD transformation issues.

**GDPA 5.2 Team Configuration & Management**

S.5.2.1 Team configuration and management demonstrate a

positive impact on game development activities.

S.5.2.2 Team members are satisfied with the communication

and collaboration protocol.

**GDPA 5.3 Requirement Modelling and Management**

S.5.3.1 The target market segment is fully captured by the

identified requirements of a particular game.

S.5.3.2 Game requirements are reviewed and revised on a

regular basis when required.

S.5.3.3 The quality attribute of games is accommodated by

identified requirements.

**GDPA5.4 Game Prototyping**

S.5.4.1 Prototyping helps in improving and developing the

final game efficiently.

S.5.4.2 Prototyping helps in identifying game mechanics,

rules, and algorithms.

S.5.4.3 The developed prototype refines the created content

of the game and also balances the gameplay.

**GDPA 5.5 Risk Management**

S.5.5.1 Risk assessment is helpful in reducing associated

development risks.

S.5.5.2 There is a backup plan to handle identified risks and

explore other solutions that would reduce or eliminate risk.

S.5.5.3 The development team always has a functional and

technical design specification with a complete risk assessment

document before the start of the production phase

for all projects.

**GDPA 5.6 Quality of Architecture**

S.5.6.1 The management team is continuously improving the

evaluation process for game architecture quality.

S.5.6.2 Game architecture documents are reviewed and

updated regularly to avoid future bottlenecks.

S.5.6.3 Game architecture includes robustness features that

enable the game to be functional in unexpected circumstances.

**GDPA 5.7 Asset Management**

S.5.7.1 The asset management system can reduce duplication

of assets and remove outdated assets from the asset

library.

S.5.7.2 Assets created for a game fit into the game concept

and have a positive effect on game appearance.

**GDPA 5.8 Game Engine Development & Management**

S.5.8.1 The development team has adequate resources and

skills to develop its own game engines for game development

or to enhance the capabilities of existing ones by adding

middleware.

S.5.8.2 Game engines are reused for different game projects.

**GDPA 5.9 Test Management**

S.5.9.1 The selected testing approach ensures game performance

and quality.

S.5.9.2 The testing team experiments with innovative techniques

on a regular basis to improve the game testing

process.

S.5.9.3 A developed test plan keeps track of functional and

non-functional requirements test outcomes and uses the

results to improve game quality and playability.

**GDPA 5.10 Maintenance Support**

S.5.10.1 The maintenance support system team regularly

examines, maintains, and improves the support system for

effective and easy reporting service.

S.5.10.2 The project team is continuously improving the

maintenance support system for developed games.

**GDPA 5.11 Fun Factor Analysis**

S.5.11.1 A blend of playability and usability methods in addition

to innovative ideas are used to enhance the consumer

playability experience in term of challenges, storyline, game

level curiosity, full control, and feeling of independence.

S.5.11.2 The fun factor analysis strategic plan is monitored

on a regular basis, and improving it is a continuous strategic

effort of the project team.

**GDPA 5.12 Ease of Use**

S.5.12.1 Consumer feedback indicates satisfaction and ability

to navigate conveniently between menu.

S.5.12.2 The defined strategy to enhance consumer experience

related to ease of use metrics is regularly reviewed

and updated.

**GDPA 5.13 Market Orientation**

S.5.13.1 The organization is able to gain competitive advantage

by using its market orientation strategy.

S.5.13.2 Developed game concepts are aligned with the

requirements of the target market.

S.5.13.3 Developed games are able to maximize their consumers’

playing time.

**GDPA 5.14 Time to Market**

S.5.14.1 Games are published before competitors’ games.

S.5.14.2 Being first to market helps to retain existing consumers

and attract new ones.

**GDPA 5.15 Relationship Management**

S.5.15.1 Developed games are able to retain their consumers

for a long time.

S.5.15.2 The development team follows a balanced playerand

game-centred strategy.

**GDPA 5.16 Monetization Strategy**

S.5.16.1 The revenue model contributes to strengthening the

financial position of the organization.

S.5.16.2 The organization successfully achieves its financial objectives.

S.5.16.3 Return on investment increases over a period of

time.

**GDPA 5.17 Innovation**

S.5.17.1 Past innovative measures taken by the development

team have resulted in improved game development and

management processes.

**GDPA 5.18 Stakeholder Collaboration**

S.5.18.1 All stakeholders are involved in game-related decisions.

(Aleem, Capretz, in Ahmed, 2016a, 67–68)

PRILOGA N: ODGOVORI NA IZJAVE V PRILOGA L ZA DOLOČITEV OCENE ZRELOSTI ZA OPTIMIZIRAN PROCESNI MODEL PO ZRELOSNEMU MODELU DGMM

|  |  |
| --- | --- |
| Šifra izjave | Odgovor |
| S.5.1.1 | DA |
| S.5.1.2 | DA |
| S.5.1.3 | DA |
| S.5.1.4 | NE |
| S.5.2.1 | DA |
| S.5.2.2 | DA |
| S.5.3.1 | DA |
| S.5.3.2 | DA |
| S.5.3.3 | DA |
| S.5.4.1 | DA |
| S.5.4.2 | DA |
| S.5.4.3 | DA |
| S.5.5.1 | DA |
| S.5.5.2 | DA |
| S.5.5.3 | DA |
| S.5.6.1 | DA |
| S.5.6.2 | DA |
| S.5.6.3 | DA |
| S.5.7.1 | DA |
| S.5.7.2 | DA |
| S.5.8.1 | NE |
| S.5.8.2 | NE |
| S.5.9.1 | DA |
| S.5.9.2 | DA |
| S.5.9.3 | DA |
| S.5.10.1 | DA |
| S.5.10.2 | DA |
| S.5.11.1 | DA |
| S.5.11.2 | DA |
| S.5.12.1 | DA |
| S.5.12.2 | DA |
| S.5.13.1 | DA |
| S.5.13.2 | DA |
| S.5.13.3 | DA |
| S.5.14.1 | DA |
| S.5.14.2 | DA |
| S.5.15.1 | DA |
| S.5.15.2 | NE |
| S.5.16.1 | DA |
| S.5.16.2 | DA |
| S.5.16.3 | DA |
| S.5.17.1 | DA |
| S.5.18.1 | DA |

1. Obstaja veliko nomenklatur: e-učenje, avtentični učni proces, igre z alternativnim namenom, sintetična učna okolja in zabavno izobraževanje (edutainment). [↑](#footnote-ref-1)
2. Virtualna resničnost. Danes najkvalitetnejši produkti: Oculus Rift, Gear VR, HTC Vive, PlayStation VR. [↑](#footnote-ref-2)
3. Zrelostni model za učne igre poda tudi. Aslan, Serdar. 2016. Digital Educational Games: Methodologies for Development and Software Quality. Imenuje ga IDEALLY - dIgital eDucational gamE softAre quaLity evaLuation methodologY. Dostopen je od 30. septembra 2016. [↑](#footnote-ref-3)
4. DGMM - Digital Game Maturity Model. Dostopen je od 16. avgusta 2016. [↑](#footnote-ref-4)
5. Thesaurus. Zbirka sopomenk (sinonimov). [↑](#footnote-ref-5)
6. Gams, Matjaž, in Boštjan Kaluža. 2013. *Računalniški slovarček*. Kamnik: Amebis. [↑](#footnote-ref-6)
7. Workflow. [↑](#footnote-ref-7)
8. Commercial off-the-shelf. [↑](#footnote-ref-8)
9. Procesni model je vpeljalo in ga uporablja podjetje IBM. [↑](#footnote-ref-9)
10. UML – Unified modeling language. V delu ga uporabljamo za grafično definiranje procesnih modelov. [↑](#footnote-ref-10)
11. Release Candidate-RC, Release to Manufacturing (RTM). [↑](#footnote-ref-11)
12. Extreme programming, ekstremno programiranje. [↑](#footnote-ref-12)
13. Dynamic System Development Method, razvoj dinamičnih sistemov. DSDM konzorcij je skrbnik metode. [↑](#footnote-ref-13)
14. Feature driven development, funkcionalno usmerjen razvoj. [↑](#footnote-ref-14)
15. Adaptive software development, prilagodljiv razvojni model. [↑](#footnote-ref-15)
16. Test driven development, testno usmerjen razvoj. [↑](#footnote-ref-16)
17. Discipline agile delivery, disciplinirana agilna dostava. [↑](#footnote-ref-17)
18. Just in time – JIT. [↑](#footnote-ref-18)
19. Work in Progress – WiP. [↑](#footnote-ref-19)
20. Je metodologija za proizvajanje produktov in ustvarjanje organizacij, ki se usmerja na krajše življenjske cikle v kombinaciji z eksperimentiranjem poslovnih hipotez, ponavljajočim izdajanjem produktov in empiričnim učenjem. [↑](#footnote-ref-20)
21. Game development lifecycle. [↑](#footnote-ref-21)
22. Game design document. [↑](#footnote-ref-22)
23. diGital educAtional gaMe dEvelopment. [↑](#footnote-ref-23)
24. Digital education game. [↑](#footnote-ref-24)
25. Backlog. Konstantno posodobljen seznam funkcionalnosti, ki jih je potrebno razviti. [↑](#footnote-ref-25)
26. Postmortem. Dokumentiran potek razvoja. [↑](#footnote-ref-26)
27. Tran, M., Biddle, R.: Collaboration in serious game development: a case study. In: Proc.

    of the 2008 Conf. on Future Play, 49–56 (2008). [↑](#footnote-ref-27)
28. Avtor igre Wolfshade MUD, pri Electronic Arts 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-28)
29. Tehnical Design Document. [↑](#footnote-ref-29)
30. Pogosti izraz je umetnikova biblija - UB. Art-Bible. [↑](#footnote-ref-30)
31. Gold-plating. [↑](#footnote-ref-31)
32. Game Unified Process. [↑](#footnote-ref-32)
33. Game Waterfall Process. [↑](#footnote-ref-33)
34. eXtreme Game Development Process. [↑](#footnote-ref-34)
35. Agile Game Process. [↑](#footnote-ref-35)
36. 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-36)
37. Multimedia asset ali game asset. [↑](#footnote-ref-37)
38. Game Development Process Activities. [↑](#footnote-ref-38)
39. NA – number of applicable statements. [↑](#footnote-ref-39)
40. PT – Passing treshold. [↑](#footnote-ref-40)
41. Proof of concept. [↑](#footnote-ref-41)
42. Vertical slice. [↑](#footnote-ref-42)
43. QA- Quality assurance. [↑](#footnote-ref-43)
44. Placeholders. [↑](#footnote-ref-44)
45. WNF: Will Not Fix, pomeni, da defekt ni dovolj pomemben. NAB: Not a bug, pomeni, da ni fefekt. [↑](#footnote-ref-45)
46. Crunch time. [↑](#footnote-ref-46)
47. Je grafikon, ki predstavlja časovni razpored projekta. [↑](#footnote-ref-47)
48. Program evaluation and review technique, je statistično orodje za analizo nalog, ki so vključene v razvoj. [↑](#footnote-ref-48)
49. AHP – Analytical Hierarchy Process. [↑](#footnote-ref-49)