UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Andraž Gruden

Proces izdelave video iger: sistematična izgradnja procesnega modela z uporabo metode MetaME in Agilnega modeliranja

Diplomsko delo

Ljubljana, 2018

UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Andraž Gruden

Mentor: izr. prof. dr. Jaroslav Berce

Proces izdelave video iger: sistematična izgradnja procesnega modela z uporabo metode MetaME in Agilnega modeliranja

Diplomsko delo

Ljubljana, 2018

Gotovo bom naredil krivico in koga pozabil. List papirja vselej ni dovolj.

Oči, mami, vama gredo vse zasluge, da sem takšen, kot sem. Brata, hvala za ljubi mir in ljubezen. Uršika, brez tebe bi odnehal. Moja najdražja, Nežika in Izak, rad vaju imam. "Quien mucho abarca poco aprieta," ste večkrat izrekli. Hvala vam za življenjsko mentorstvo. Očitno je potrebna ena dekada, da iz osla narediš človeka. Hvala, Edo.

Proces izdelave video iger: sistematična izgradnja procesnega modela z uporabo metode MetaME in Agilnega modeliranja

Razvoj video iger ima zaradi slabo definiranega procesa visoka tveganja in otežen razvoj. Projekti pogosto končajo v neobvladovanju ali finančnih izdatkih, ki zadušijo razvoj. V delu smo predstavili vse procesne modele, ki so na voljo za razvoj video iger. Na podlagi pregleda literature smo ugotovili, da se v praksi pretežno uporabljajo XP, Scrum in njuni hibridi. Kljub temu pa raziskani modeli posedujejo pomanjkljivosti, ki jih poskušamo v tem delu zakrpati s predlaganjem procesnega modela, ki temelji na praksah in usmeritvah discipline razvoja video iger. Za predlagani model smo izvedli optimizacijo s prvim zrelostnim modelom za igre DGMM, ki je potrdil izvajanje več kot 80 % procesnih aktivnosti. Posebnost predlaganega procesa je definiranje dodatnega delovnega toka izdelave kreativnih sredstev in njegovo sinhronizacijo z razvojem. Z uporabo predlaganega modela predvidevamo zmanjšanje tveganja, natančnejšo napoved trajanja projekta in onemogočanje kriznega časa ter ostalih najpogostejših težav v razvoju. Disciplina razvoja video iger je okrnjena napredka zaradi nekonsistentne rabe terminologije in konceptov, katerih temelje za prihodnje raziskave postavljamo s tem delom.

Ključne besede: video igre, procesni model, inženiring metod.

Game development process: systematic structuring of the process model using MetaME method and Agile modeling

Game development bears high risk and difficult development due to poorly defined process. Projects often end up ungovernable or with financial expenses that asphyxiate development. In this thesis we have presented all process models that are available for game development. Based on the literature review, our findings point out that in practice, XP, Scrum and their hybrids are mostly used. Models presented are holding irregularities that we try to rectify by proposing a process model based on practices and orientations of the video game development discipline. Proposed model was optimized with first maturity model for games DGMM, that confirmed the operation of more than 80% of process activities. Proposed model uniquely defines an additional workflow of managing creative assets which is synchronized with development. With the implementation of the proposed model, we anticipate risk reduction, more precise forecast of the project duration, avoidance of crunch time and other common development issues. Video game development is truncated of the progress because of the inconsistent use of terminology and concepts. Their foundations for future research are laid within this work.

Keywords: video games, process model, method engineering.

KAZALO VSEBINE

KAZALO TABEL

KAZALO SLIK

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 se 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, str. 1). Video igre so vsestranski informacijski sistemi (Hamari in Keronen, 2017, str. 136), ki poleg zabave služijo tudi kot orodja za informiranje, oskrbo zdravja, medicine, usposabljanja, in učenja.

Igre kot učni pripomočki[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, str. 4). Igre imajo velik potencial pri podpori učenja konceptualnega razumevanja, procesov, praks, epistemologije in nenazadnje obnašanje (Clark, Tanner-Smith in Killingsworth, 2016, str. 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 (Schott in Hodgetts, 2006, str. 311). Spodbujajo se simulacije z VR[2] pripomočki z namenom praktičnega izobraževanja neizkušenih kirurgov pri katerih je ogrožanje zdravja pacientov med procesom izključeno (Ma in Zheng, 2011, str. 171). Obstajajo tehtni dokazi, da lahko igre posredujejo vplive, ki uporabnika za določena zdravstvena stanja naredijo bolj učinkovitega (prav tam, str. 311). Igralni pripomočki kot so Dance Dance Revolution, Sony Eyetoy in Nintendo Wii spodbujajo zdravje s ponujanjem alternativnih pristopov fizičnih aktivnosti (Schott in Hodgetts, 2006, str. 311). Primer je igra Rex Ronan, ki spodbuja zdrav življenjski slog s spodbujanjem protikadilskih nagnjenj (prav tam). Ulbin predstavi nevsiljiv način spremembe življenjskega sloga z izdelavo igre, ki bi pomagala otrokom s prekomerno telesno težo (Ulbin, 2017, str. 30). Številne raziskave nakazujejo tudi na terapevtske potenciale (Salmon in drugi, 2017, str. 45).

[1]Obstaja veliko nomenklatur: e-učenje, avtentični učni proces, igre z alternativnim namenom, sintetična učna okolja in zabavno izobraževanje (edutainment).

[2] Virtualna resničnost. Danes najkvalitetnejši produkti: Oculus Rift, Gear VR, HTC Vive, PlayStation VR.

Igre pozitivno vplivajo na kognitivne dejavnosti, mobilnost, rehabilitacijo bolnikov, ki trpijo za izgubo ali oslabitvijo motoričnih funkcij (kap, parkinsonizem) (Salom in drugi, 2017, str. 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 ("Breathing Labs", b. d.; "Kinestica", b. d.).

Prihodki industrije iger vztrajno rastejo. Industrija iger je globalno leta 2013 dosegla 60 milijard dolarjev, kar je zasenčilo filmsko industrijo (Fullerton, 2014, str. 459–460). Zadnje poročilo ocenjuje prihodek industrije iger na 30,4 milijarde dolarjev ("ESA", 2017). 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 (L. Rakestraw, V. Eunni in Kasuganti, 2012, str. 1). S porastom prodaje teh naprav je sunkovito poskočilo povpraševanje po igrah. Leta 2011 je bilo vsak teden izdanih 15.000 novih aplikacij (prav tam, str. 1). 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, str. 2013).

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", 2014). Povprečje za EU28, ki ga navaja Eurostat pa je 38.000 dolarjev ("Eurostat", 2017). EY za leto 2014 navaja, da je v Evropi zaposlenih 108.000 ljudi v industriji iger, ki ustvarijo preko 16 milijard evrov dobička. ("EY", 2014). 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, 2016a, str. 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, str. 182). Na izdajne roke vplivajo tudi slaba organizacija, upravljanje (Fábio Petrillo, Pimenta, Trindade in Dietrich, 2009, str. 4) in izbira metodologije (Kanode in Haddad, 2009, str. 556). Igre s slabo razvojno metodologijo bolj verjetno presežejo finančna sredstva in časovne termine (prav tam). 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, str. 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, str. 29) in z njo razvojni procesi (O'Hagan in drugi, 2014, str. 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). 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 (Aktaş in Orçun, 2016, str. 240). To je bilo zadovoljivo v času, ko so bile razvojne ekipe in količine potrebnih virov za izgradnjo iger manjše (prav tam). 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 (prav tam). Težavnost se lahko le še stopnjuje, če so ekipe geografsko razdeljene (O'Hagan, in drugi, 2014, str. 182). Igre oblikujejo ekipe 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, str. 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 in drugi, 2014, str. 182). V razvoju iger je močno vključeno ustvarjanje kreativnih vsebin, ki definirajo izgled igre (Aktaş in Orçun, 2016, str. 249). Vendar igre niso povsem le umetnost, kot tudi ne produkt popolnega inženiringa (Ramadan in Widyani, 2013, str. 95). Razvoj igre je bolj podoben ustvarjanju izdelka s prepletanjem aspektov umetnosti, glasbe, programiranja, igranja in poslovnega upravljanja (prav tam). Igre bolj stremijo k proizvajanju uporabniške izkušnje kot uporabnosti. (O'Hagan in drugi, 2014, str. 182). Poudarek je na evalvaciji uporabniške izkušnje in uporabi povratnih informacij, ki nato vodijo tok iteracij razvoja (prav tam, str. 183). Pressman zagovarja, da so igre programska oprema, ki ponujajo zabavo (Ramadan in Widyani, 2013, str. 95), vendar ni metode za določanje subjektivnega elementa zabave, na katero se osredotočajo oblikovalci iger. Zato je razvoj iger v nasprotju s tradicionalnim razvojem programske opreme toliko bolj kompleksen (Fábio Petrillo in drugi, 2009, str. 19). Potrebno je razširiti podedovane tradicionalne tehnike razvoja programske opreme, da bi lahko podprli kreativni proces razvoja video iger (prav tam).

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

(Aleem in drugi, 2016a, str. 55; Ramadan in Widyani, 2013, str. 95). V kratki zgodovini discipline usmeritve in prakse še niso bile povsem raziskane (Aleem, Capretz in Ahmed, 2016b, str. 27). Posledično ni bilo izdelanega procesnega modela, ki bi temeljil na najboljših praksah razvoja iger (O'Hagan in O'Connor, 2015, str. 15). Takšen model bi lahko zmanjšal čas razvoja, čas vstopa na trg ali celo izboljšal kvaliteto iger (prav tam). Prav tako do nedavnega še ni bilo zrelostnega modela, ki bi direktno naslovil težave ocenjevanja in izboljšanja procesov (Aleem in drugi, 2016a, str. 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] primernim zrelostnim modelom za igre[4]. Delo je prikaz sistematičnega pristopa k izgradnji domenskega procesa, ki kljub sistematizaciji in optimizaciji ohranja prostor za kreativnost.

[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.

[4] DGMM - *Digital Game Maturity Model*. Dostopen je od 16. avgusta 2016.

2 METODOLOGIJA

Metodološki pregled pretekle literature je ključnega pomena vsakega akademskega raziskovanja. Področja kot je inženiring, so zaradi podcenjevanja pomembnosti metode pregleda literature kronično utrpela pomanjkanje raziskovalnih del, kar je otežilo teoretski in konceptualni napredek (Levy in Ellis, 2006, str. 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 akademske le malo (Ruonala, 2016, str. 1). Razvoj iger je inherentno agilna aktivnost (prav tam). Večina organizacij je podedovala agilne prakse (prav tam), za katere obstaja veliko literature, vendar ta ni usmerjena v razvoj iger (Barbosa in Godoy, 2010, str. 292). S pregledom literature želimo v disciplini razvoja video iger postaviti 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 Ellis, 2006, str. 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, str. 77), ki omogočajo modifikacijo poizvedb. Če je bilo to mogoče, smo se posluževali teh. Da bi strnili zadetke, smo bili v veliko primerih 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], ki služijo podatkovnim bazam za naslavljanje predmetov ali opisov (Booth in drugi, 2012, str. 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 (prav tam, str. 76).

[5] Thesaurus. Zbirka sopomenk (sinonimov).

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 besedil. To nam je pomagalo pri spoznavanju vplivnih literatur, avtorjev in spoznanju terminologije tematike.
Kot vplivne literature smo identificirali dela:

1. Ruonala, H.-R. (2016). Agile Game Development: A Systematic Literature Review.
2. Aleem, S., Capretz, L. F., in Ahmed, F. (2016a). A Digital Game Maturity Model (DGMM).

Prvo delo je sistematičen pregled literature, ki vsebuje analize člankov in njihove povezave. Slednje delo je prvi primer zrelostnega pristopa za igre, ki smo ga uporabili v našem delu. Iz terminologije smo razbrali gesla, ki so nam pomagala pri kreiranju poizvedb. Gesla si 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, str. 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 njene 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 ekvivalentnih izrazov procesnih modelov (življenjskih ciklov, metod):

1. Code and fix (Tsui, Karam, in Bernal, 2016, str. 58) ~ Build and fix (Sabharwal, 2009, str. 8).
2. Stage gate (Peters, 2008, str. 113) ~ Phased-release (Lethbridge in Laganiere, 2005, str. 430).
3. Incremental Software Development Life Cycle (Sabharwal, 2009, str. 16) ~ Incremental delivery (Sommerville, 2010, str. 47) ~ Incremental Process (Pressman in Maxim, 2014, str. 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]. Besede, ki jih nismo našli. smo prevedli sami in jih podprli z opombo v izvorni obliki.

[6] Gams, M. in Kaluža B. (2013). *Računalniški slovarček*. Kamnik: Amebis.

3 PROCESNI MODELI

Razvoji video iger so v prvi vrsti projekti razvoja programske opreme (Bates, 2004, str. 217). Zato se za njihovo upravljanje 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 (Pressman in Maxim 2014, 15). Temelj programskega inženiringa predstavlja proces (prav tam). Proces je zbirka aktivnosti, del in nalog, ki se izvajajo ob ustvarjanju produkta (Pressman in Maxim, 2014, str. 16) in lahko vsebujejo tudi sebi podrejene procese (Sommerville, 2010, str. 16). Predstavlja primarni pristop, ki projekt organizira v aktivnosti (Lethbridge in Laganiere, 2005, str. 10). Te aktivnosti bivajo v okvirjih ali modelih, ki definirajo njihove povezave s procesom in drugimi aktivnostmi (Pressman in Maxim, 2014, str. 31). Procesi iste narave so razdeljeni v procesne modele (Rolland, 1998, str. 8), ki so poenostavljena predstavitev procesa razvoja (Sommerville, 2010, str. 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 (Muffatto, 2006, str. 73). Vsakršna izguba člana razvoja ima negativen učinek na proces razvoja (prav tam). Med ad hoc modele spadajo Slam dunk, Big bang in priložnostni model (Jalote, Palit, Kurien in Peethamber, 2004, str. 2; Lethbridge in Laganiere, 2005, str. 428; Peters, 2008, str. 109). Ker se pri slednjem kodiranje smatra za osrednjo aktivnost tega procesa, ga v praksi imenujejo tudi (code-and-fix) kodiraj in popravi (Tsui in drugi, 2016, str. 74) ali (build-and-fix) gradi in popravi (Sabharwal, 2009, str. 8).

3.2 SEKVENČNI PROCESNI MODELI

Procesni modeli iz te družine stremijo k vzpostavitvi strukture in ureditve razvoja programske opreme (Pressman in Maxim, 2014, str. 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] , ki odraža medsebojno povezanost procesnih elementov (prav tam).

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

[7] Workflow.

3.3 EVOLUCIJSKI PROCESNI MODELI

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 (Pressman in Maxim, 2014, str. 45). Ti modeli so ponavljajoči, iterativni. Sem spadajo prototipiranje, spiralni in vzporedni procesni model (prav tam).

Prototipiranje poteka po iteracijah. Načrt in izvedba prve iteracije sta hitra (prav tam). Investitor po iteraciji prototip pregleda in poda povratno informacijo (prav tam). Takšen potek omogoča natančnejše definiranje zahtev (prav tam). Iteracija se zaključi ob rektifikaciji prototipa in izpolnitvi želja investitorjev, sočasno pa pripomore k boljšemu razumevanju prihodnjih. Navadno ta model služi kot idealen mehanizem za prepoznavo programskih zahtev (prav tam, str. 46). Spiralni model združuje iterativno naravo prototipiranja in kontroliran sistematični pogled kaskadnega modela (prav tam, str. 48).

Omogoča hiter razvoj z izdajanjem vse bolj popolne različice programa (prav tam). Zgodnje izdaje predstavljajo modele ali prototipe, kasnejše iteracije pa vedno bolj popolne različice (prav tam). Vzporedni model omogoča uporabo iterativnih in vzporednih elementov v kombinaciji z vsemi spoznanimi modeli v tem poglavju (prav tam). Vse razvojne aktivnosti se odvijajo vzporedno, vendar je lahko vsaka aktivnost na svoji stopnji razvoja. (prav tam, str. 50). Model prestavlja abstrakcijo dejanskega procesnega cikla (prav tam). V njem se definira serijo sekvenčnih dogodkov vseh aktivnosti, dejanj in opravil (prav tam). Vsaka aktivnost, dejanje ali opravilo se odvija vzporedno z drugo aktivnostjo, dejanjem in opravilom (prav tam, str. 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, str. 52). Sem spadajo komponentni, formalni, unificiran, sinhronizacijsko in stabilizacijski model ter rapidni razvoj (prav tam, str. 52–54).

Komponentni model nastavlja proces razvoja programske opreme z že prevedeno programsko opremo (prav tam, str. 53). Komercialne komponente (COTS[8]) zagotavljajo namenske funkcionalnosti z dobro definiranimi vmesniki, ki omogočajo integracijo v programsko opremo v razvoju (prav tam).

[8] *Commercial off-the-shelf.*

Po naravi je model evolucijski, ta pa zahteva iterativni pristop k razvoju (Pressman in Maxim, 2014, str. 53). Formalni model zajema zbirko aktivnosti, ki vodijo v formalno, matematično specifikacijo programske opreme (prav tam). Ta model omogoča specifikacijo, razvoj in verifikacijo sistemov z apliciranjem strogih matematičnih notacij (prav tam). Primer formalnega razvojnega procesa je model čiste sobe (prav tam). Vsak inkrement v razvoju ima formalno specifikacijo, na podlagi katere se izvede implementacija (Sommerville, 2010, str. 32). Unificiran model je poskus združitve najboljših značilnosti procesnih modelov z umestitvijo njihovih najboljših praks v agilne procese (Sommerville, 2010, str. 50; Pressman in Maxim, 2014, str. 57). Je primer modernega procesnega modela, ki je bil izpeljan iz unificiranega jezika modeliranja[9] in povezanega unificiranega razvojnega procesa programske opreme (Sommerville, 2010, str. 50). Sinhronizacijsko-stabilizacijski model je podoben inkrementalnemu (Peters, 2008, str. 120). Razvojna ekipa izdela specifikacijo, določi prioritete in razdeli razvoj na štiri večje izdaje (prav tam). Izdaje[10] so kandidat za javnost (RC) 1 do 3 in izdaja v proizvodnjo (prav tam). Kandidat za javnost je prvi večji izid in predstavlja mejnik v razvojnem ciklu z možnostjo širše modifikacije kode in vsebine (prav tam). Rapidni razvoj predvideva fragmentacijo programskih zahtev v module (Sabharwal, 2009, str. 19). Ti se nato razvijajo in neodvisno integrirajo v produkt (prav tam). Najpomembnejši atribut modela je hitrost poteka razvoja od analize zahtev do končnega sistema (prav tam). Čas izdaje enega modula navadno traja od 60 do 90 dni in ga imenujemo časovni okvir (prav tam). Modularna fragmentacija omogoča uporabo komponent, na podlagi katerih se skrajša časovni okvir razvojnega cikla in stroške razvoja (prav tam, str. 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 in drugi, 2016, str. 84). Agilni razvoj je pravzaprav nabor najboljših praks, zbranih iz drugih življenjskih ciklov in uspešnih praks kodiranja (Peters, 2008, str. 116). Ne obstaja končna definicija, kaj sestavlja Agilno metodo, vendar obstaja kar nekaj karakteristik, ki so metodam sorodne (Tsui, Karam in Bernal, 2016, str. 84).

[9] UML – *Unified modeling language*. V delu ga uporabljamo za grafično definiranje procesnih modelov.
[10] *Release Candidate* (RC), *Release to Manufacturing* (RTM).

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, str. 59). Tabela 3.1 spodaj jih podrobno opisuje.

Tabela 3.1: Podrobno opisana načela agilnih modelov

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Vir: Sommerville (2010, str. 60).

Med agilne modele spadajo XP[11], Scrum, Crystal, Lean, Kanban, DSDM[12], FDD[13], ASD[14], TDD[15], DAD[16], odprtokodni model in Scrumban (Pressman in Maxim, 2014, str. 72–83; Sommerville, 2010, str. 58–74; Peters, 2008, str. 112–124; Lethbridge in Laganiere, 2005, str. 433–434). Ekstremno programiranje uporablja paradigmo objektno orientiranega razvoja (Pressman in Maxim, 2014, str. 72). Posebnost modela je izdelava zgodb (prav tam, str. 73) ali scenarijev na podlagi zbiranja zahtev (Sommerville, 2010, str. 65). Te opisujejo potrebne funkcionalnosti in lastnosti programske opreme (Pressman in Maxim, 2014, str. 73). Scrum model je agilna metoda, ki se fokusira na upravljanje inkrementalnega razvoja. Inovativna lastnost modela so sprint cikli (Sommerville, 2010, str. 73). Sprint je enota načrtovanja, kjer se oceni potrebno delo, izberejo funkcionalnosti za razvoj in opravijo implementacije programske opreme (Sommerville, 2010, str. 73).

[11] *Extreme programming*, ekstremno programiranje.
[12] *Dynamic System Development Method*, razvoj dinamičnih sistemov. DSDM konzorcij je skrbnik metode.
[13] *Feature driven development*, funkcionalno usmerjen razvoj.
[14] *Adaptive software development*, prilagodljiv razvojni model.
[15] *Test driven development*, testno usmerjen razvoj.
[16] *Discipline agile delivery*, disciplinirana agilna dostava.

Kristalno družino modelov je razvil Alistair Cockburn z namenon prilagoditve metodologij projektom. Primeri modelov so: Crystal Clear, Crystal Orange, Crystal Orange Web (Tsui, Karam, in Bernal 2013, 91; Abrahamsson, Salo, Ronkainen in Warsta, 2017, str. 39).

Metodologije so definirane z barvami (Tsui, Karam, in Bernal 2013, str. 91). Temnejša kot je barva, primernejša je metodologija za zahtevnejše projekte (prav tam). Lean ali vitka metoda je bila ustvarjena za potrebe avtomobilske industrije (Janes in Succi 2014, 131). Povzema sedem principov; znižanje odpadkov, kvaliteto produkta, ustvarjanje znanja, proizvodnjo v pravem trenutku[17], spoštovanje ljudi in konstantno optimizacijo (prav tam).

Medtem ko se ostale agilne metode nagibajo k agilnosti, se ta metoda k efektivnosti (prav tam, str. 144). Kanban metoda se izvaja z uporabo kanban sistema, ki omejuje delo v teku z uporabo vizualnih signalov (Anderson in Carmichael, 2016, str. 1). Vizualizacija se prične s Kanban tablo, ki mora imeti vidno označene in definirane točke zaveze in jasno opredeljene meje dela v teku (prav tam, str. 13). Zaveza je eksplicitni dogovor med naročnikom in razvojem, ki narekuje razvoju delo na tistih postavkah, ki jih naročnik želi (prav tam, str. 15). Zahteve ali postavke se nahajajo v bazenu idej, ki so lahko izbrane (prav tam). Čas od izbrane postavke do izdaje definira naročnikov dobavni rok (prav tam). Dobavni rok razvoja pa je čas od izbranih postavk do njihove dostave naročniku (prav tam). Ta čas definira vse postavke in ga označuje kot delo v teku[18] (prav tam). DSDM je metoda za razvoj sistemov z zahtevnimi časovnimi omejitvami s pomočjo inkrementalnega prototipiranja v kontroliranem okolju (Pressman in Maxim, 2014, str. 80). Filozofija modela sloni a modificiranem Paretovem principu, da je mogoče narediti 80 % aplikacije v 20 % časa, ki bi bil potreben za izdajo celotne aplikacije (prav tam).

Funkcionalno usmerjen razvoj je zgrajen okoli jedra najboljših praks. Zbrane zahteve naročnika predstavljajo razredi s svojimi metodami (funkcijami), ki skupaj tvorijo funkcionalnosti (Palmer in Felsing, 2002, str. 39). Funkcionalnosti se ovrednotijo glede na uporabno vrednost naročnika. Ovrednotene služijo kot vodilo in sledenje napredku razvoja (prav tam). Prilagodljiv razvojni model se fokusira na razvoj velikih in kompleksnih sistemov (Abrahamsson in drugi, str. 73). Metoda spodbuja uporabo inkrementalnih in iterativnih tehnik s konstantnim prototipiranjem (prav tam). Testno usmerjen razvoj je evolucijski proces, pri katerem se izdela testni protokol in test pred produkcijsko kodo (Duka in Hribar, 2010, str. 1). To pa zato, ker testni scenariji vodijo načrtovanje, saj ti določajo potrebe (prav tam, str. 1).

[17] *Just in time* – JIT.
[18] *Work in Progress* – WiP.

Disciplinirana agilna dostava je metoda, katere osrednji del je izbira dostavnih (izdajnih) pristopov na podlagi problematike (Lethbridge in Laganiere, 2005, str. 434). DAD pozna štiri različice razvojnega dela procesa. Prva je agilna/bazična različica, ki razširja cikel izgradnje po Scrum metodi (prav tam). Druga je naprednejša/Lean, ki ima za osnovo Kanban. Tretja je neskončni cikel izdajanja in zadnja je raziskovalna, ki ima za osnovo Lean Start-up[19] pristop (prav tam). Odprtokodni model brezplačno distribuira programsko opremo z izvorno kodo (prav tam). Model pričakuje, da bodo člani skupnosti zato imeli razlog za izboljšanje programa. Zagotavljanje kakovosti izvaja skupnost sama (prav tam). Različice se izdajajo na podlagi specifikacij, pridobljenih preko e-pošte, oglasnih desk in drugih neformalnih medijev (prav tam).

Integracije se dogajajo pogosto, preko spleta. Razvijalci imajo skupne vizije, saj opremo potrebujejo in jo razvijajo za lastne potrebe (Tsui in drugi, 2016, str. 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, str. 238). Od metode Scrum je povzel tehnike dnevnih sestankov, definiranja zgodb in sam 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 s 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 doseganje optimalne pretočnosti. WiP pomaga razvijalcem omejiti večopravilnost, da bi povečal produktivnost (prav tam).

[19] 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.

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 podrobneje definirata faze življenjskega cikla igre (Babu in Maruthi, 2013, str. 1491). 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 (prav tam). Z definicijo življenjskega cikla podata razširjen vpogled v faze razvoja iger (prav tam, str. 1502).

Ramadan in Widyani na podlagi ključnih aktivnosti različnih organizacij in raziskovalcev predlagata GDLC[20] (PRILOGA Č) (Ramadan in Widyani, 2013, str. 98). Pristop predlaga faze: iniciacijo, predprodukcijo, produkcijo, testiranje, beto in izdajo (prav tam). Iniciacija služi za stvaritev koncepta igre in enostaven opis specifikacije (prav tam). Predprodukcija zajema revidiranje oblikovanja in izdelavo prototipa. Po tej fazi je končan GDD[21], na podlagi katerega je izdelan prototip (prav tam). V prvi iteraciji produkcije služi prototip za temelje in strukturo, v nadaljnjih pa se ga izboljšuje (prav tam). Produkcija se konča z izdelanim prototipom, ki predstavlja celoto (prav tam). Sledi testiranje, po katerem se izda poročilo defektov in seznam izboljšav. Rezultat testiranja odloča, ali se razvoj nadaljuje v beta fazo (prav tam). Slednja predstavlja identifikacijo defektov in povratnih informacij uporabnikov (prav tam). Iz bete lahko ponovno sledi faza produkcije z namenom izboljšanja igre ali faza izdaje, če je rezultat beta testiranja zadovoljiv (Ramadan in Widyani, 2013, str. 99).

Aslan in Balci predstavita metodologijo za kompleksni razvoj iger GAMED[22]. PRILOGA B prikazuje življenjski cikel DEG[23], ki je osnova metodologije (Aslan in Balci, 2015, str. 309) . DEG cikel je sestavljen iz štirih faz: oblikovanja igre, oblikovanja programske opreme, implementacije (izdaje) in učenja na podlagi iger (povratne informacije) (prav tam, str. 309. Metodo označujeta za iterativno, saj pričakuje povratne tranzicije. Če 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 (prav tam, str. 310). Za podporni proces metodologija uporablja spiralni model, ki ga aplicira v fazo oblikovanja igre (PRILOGA C).

[20] *Game development lifecycle.*
[21] *Game design document.*
[22] *diGital educAtional gaMe dEvelopment*.
[23] *Digital education game.*

Oblikovanje poteka po spiralnem vzorcu, pri čemer vsaka iteracija pomeni izpopolnjen dokument specifikacije ideje. Iteracija vsebuje aktivnosti: prototipiranja, testiranja igranja, evalvacije in analize tveganj (Aslan in Balci, 2015, str. 313).

Vzorec se nadaljuje, dokler kvaliteta oblikovanja ni potrjena v aktivnosti evalvacije. Končni dokument te aktivnosti predstavlja specifikacija oblikovanja igre (prav tam). 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 (prav tam, str. 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 informacij. Po potrebi posodobitve se življenjski cikel ponovno izvede (prav tam, str. 317).

Barbosa in Godoy predstavita Game-Scrum, ki je hibrid Scrum in XP procesnih modelov (Barbosa in Godoy, 2017, str. 293). Scrum skrbi za upravljanje projekta, medtem ko XP zagotavlja inženiring. Procesni hibrid je primeren za razvojne ekipe z malo ali nič izkušnjami (prav tam, str. 293). Proces se deli na faze: predprodukcija, faza izdelave GDD, produkcija in postprodukcija (prav tam). V predprodukciji poteka iskanje dejavnika zabave, izdelava idealnega koncepta in oblikovanja (prav tam). Delo poteka po principu metode poskušanja in popravljanja napak (prav tam).

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[24] (PRILOGA H, pod Scrum). Produkcija poteka iterativno, priporoča se Kanban metodo za kreacijo umetnin in podajanje časovnih okvirjev ter mejnikov njihove izdelave (prav tam, str. 293). Če 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[25] razvoja, ki služi za identifikacijo pomanjkljivosti v preteklem razvojem procesu (prav tam, str. 294).

[24]Backlog. Konstantno posodobljen seznam funkcionalnosti, ki jih je potrebno razviti.
[25] Postmortem. Dokumentiran potek razvoja.

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 (Musil, Schweda, Winkler in Biffl, 2010, str. 5). Verjetno je 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 (prav tam).

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

Koutonen in Leppänen sta 2013 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, str. 12). Prav slednja povzemata tudi delo avtorjev Tran in Biddle (2008)[26], ki označujeta Scrum, XP in Kanban za najpogosteje uporabljene modele (prav tam, 2013, str. 4).

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

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

Prav tako agilne pristope podpira Fullerton, saj meni, da je Scrum primeren za reševanje zapletenih problemov oblikovanja iger (Fullerton, 2014, str. 369). Svoje mnenje podaja John Comes[27], ki meni, naj opustimo kaskadne (sekvenčne) procese in sprejmemo agilne (Novak, 2012, str. 366).

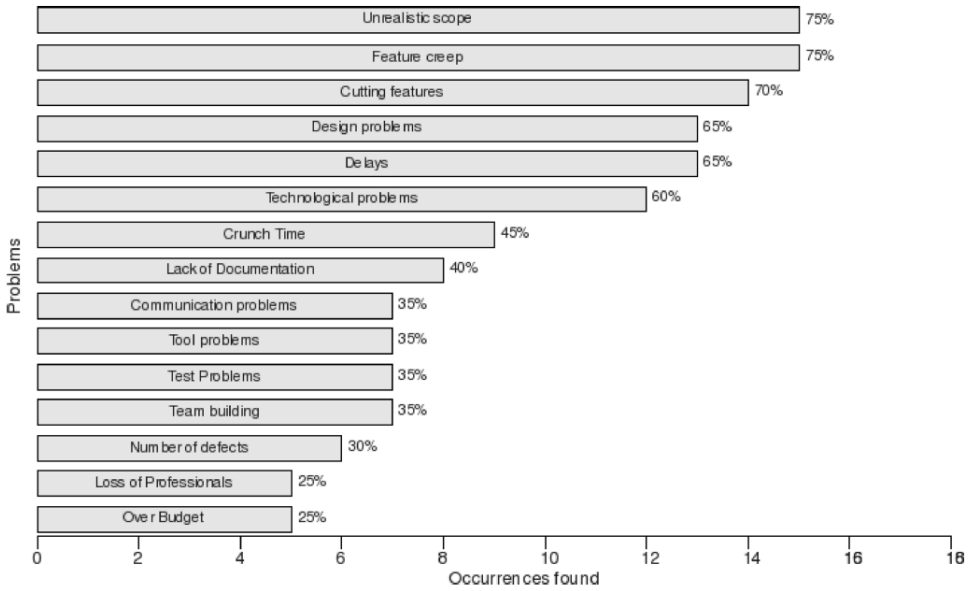
[26] Tran, M., Biddle, R. (2008). Collaboration in serious game development: a case study. In: Proc. of the 2008 Conf. on Future Play, 49–56.

[27] Avtor igre Wolfshade MUD, pri Electronic Arts izdal: Command & Conquer: Generals – Zero Hour in Lord of the Rings: The Battle for Middle-Earth, pri Powered Games izda: Superme Commander in Demigod.

4.1 POMANKLJIVOSTI AGILNIH PRISTOPOV

Agilni pristopi niso usmerjeni v izdelavo dokumentacije, ampak na zgodaj delujočo verzijo igre (O'Hagan in O'Connor, 2015, str. 5). Slabo definirana specifikacija vodi v defekte, vpliva na funkcionalnost, vire in časovnice (Kanode in Haddad, 2009, str. 263). Med produkcijo ti pristopi nadaljujejo s prototipiranjem in izdelavo inkrementov igre, kar privede do sprememb, ki drastično vplivajo tudi na GDD (prav tam). Posledično se spremeni okvir projekta, kar negativno vpliva na produkcijo (prav tam). Poleg tega so bili agilni modeli namenjeni za razvoj programske opreme in ne iger (O'Hagan in O'Connor, 2015, str. 5). Več kot polovica študij primerov omenja kreativnost kot glavni izziv uporabe agilnih pristopov (Ruonala, 2016, str. 29). Multidisciplinarne ekipe se pogosto delijo med umetnike in programerje, kar privede do težav v komunikaciji (Kanode in Haddad, 2009, str. 263). Umetniki imajo večji odpor do sprejemanja agilnih metod in ostalih uveljavljenih procesnih kontrol kot tehnično orientirani zaposleni (Ruonala, 2016, str. 29). Brazilska raziskava je pokazala, da agilne prakse tudi niso najučinkovitejše (Politowski, Vargas, Fontura in Foletto, 2016, 155). V raziskavi je sodelovalo 58 podjetij, ki razvijajo video igre (prav tam). 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. Tabela 4.1 spodaj prikazuje rezultate te raziskave.

Tabela 4.1: Najpogostejše težave, ki se pojavljajo v razvoju iger



Vir: Petrillo in drugi (2009, str. 18).

Kategorije procesov so bile definirane glede na naravo procesov in si sledijo: agilni, predpisujoči, ad-hoc in brez procesa (Politowski, Vargas, Fontura in Foletto, 2016, 155). Agilne predstavljajo XP, Kanban ter Scrum in veljajo za iterativne pristope konstantnega posodabljanja procesa. Predpisujoči so derivati kaskadnih in sekvenčnih procesnih modelov (prav tam, 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 definira pristope brez metodološke podlage (prav tam). Pogoste težave so bile predstavljene na podlagi raziskave, ki so jo o pogostih težavah med razvojem igre naredili Petrillo in kolegi (prav tam). Rezultati so pokazali, da so zakasnitve izdajnih rokov, nerealističen okvir in pomanjkanje dokumentacije najpogostejši problemi (prav tam, 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 kot tisti z največjo stopnjo uspešnosti projektov (prav tam, 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, saj se lahko iz faze beta testiranja vrne v predprodukcijo, medtem ko med produkcijo poteka iterativna izboljšava prototipa. Čeprav pristop podpira postprodukcijsko 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, str. 216). Glavni razlog je ujemanje dobavnih rokov in najbolj priljubljenih časov izdaj, kot je recimo božič (Ruonala, 2016, 8). Zato je bistveno, 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, str. 286). Posledično GDLC 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 PRILOGI D je prikaz več artefaktov dokumentacije, ki jih avtorji identificirajo med razvojem igre. Zato sklepamo, da en sam dokument ne zadostuje za razvoj video iger.

Čeprav ni določenih standardov za kreiranje dokumentacije, veljajo določeni dokumenti za osnovne komponente trdnih temeljev. In te predstavljajo predlog igre, GDD, TDD[28], vodnik umetniškega sloga[29] (UB), projektni načrt (Novak, 2012, str. 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, str. 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 D) kot pri definiranju faz razvoja (PRILOGA E), ki je pa pristop ni podedoval. Največjo težavo pristopa predstavlja obnašanje cikla. Faza oblikovanja je iterativen proces (Schell, 2008, str. 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, str. 310) in se v toku produkcije dnevno spreminja (Novak, 2012, str. 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, str. 367), kar je v nasprotju z linearno realizacijo zahtev iz dokumentacije. Ta lahko privede do pozlačevanja[30], ki pomeni sprejemanje nemogočih zahtev (Rucker, 2002, str. 27) in je nasprotje iterativnemu razvoju (Novak, 2012, str. 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[31] 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, str. 22). Flood pri definiranju GUP predstavi in komentira GWP[32] ("Gamedev", 2003). Kljub številnim pomanjkljivostim ta proces prvič predstavi artefakte, ki so specifični za igre (Wilson Brotto Furtado, 2012, str. 21) (specifikacija igre, umetniška biblija, tehnična specifikacija) ("Ecured", b. d.). GWP, ki je bil uporabljen pri igri Ankh omenjata tudi (Flynt in Salem, 2004, 151).

[28] *Tehnical Design Document*.
[29] Pogosti izraz je umetnikova biblija - UB. *Art-Bible* - AB.
[30] Gold-plating.
[31] *Game Unified Process*.
[32] *Game Waterfall Process*.

GWP ima težavo, da predstavlja linearen proces ("Ecured", b. d.). Pri tem procesu ocenjevalci, testerji in ravnatelji ne vidijo produkta, dokler je ta v razvoju (prav tam). V realnosti je tako, da se po evalvaciji navadno zahtevajo spremembe (prav tam). To zahteva, da razvoj ponovi faze, ki so bile predvidoma že končane (prav tam). Razvoj iger ni linearen proces (Flynt in Salem, 2004, str. 151). V praksi je tendenca po izogibanju uporabi kaskadnih pristopov, ker je nemogoče definirati in načrtovati program vnaprej (Rucker, 2002, str. 38). V praksi je GWP (razvoj igre Ankh) predstavljal napor ("Ecured", b.d.). Razvoj igre potrebuje fleksibilnost zato, da lahko ekipa vključi spoznanja, ki so se identificirala med razvojem z namenom konstantne izboljšave kvalitete (Flynt in Salem, 2004, str. 18). GWP in prav tako GUP ne podajata priporočila uporabe ali referenčnega modela za aplikacijo teh procesov, na kar najverjetneje vpliva tudi tajna narava industrije iger (Wilson Brotto Furtado, 2012, str. 22). Med neuspešne spada tudi model, ki ga objavi Demachy na Gamasutra ("Gamasutra", 2003). Ta interpretira XP metodologijo in predlaga XGD[33] za video igre (prav tam). Vendar pristop ne zagotavlja strukturiranih usmeritev, deloma zato, ker gre za agilno metodologijo (Wilson Brotto Furtado, 2012, str. 22). Poleg naštetih se v akademski literaturi pojavita tudi dve pobudi procesov AGP[34] in Predpisujoča metodologija za razvoj video iger, ki sta nastali v okviru zaključnih nalog. Obema predlogoma primanjkuje evalvacijski proces za določanje njune ustreznosti (prav tam). Slednji pa je neustrezen, ker igre niso predpisujoče (Koster, 2013, str. 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, str. 10). Zato kakršnikoli predpisujoči (sekvenčni) pristopi niso priporočljivi.

[33] *eXtreme Game Development Process*.
[34] *Agile Game Process*.

4.3 UGOTOVITVE IN USMERITVE MODELIRANJU

V času predprodukcije se priporoča prototipiranje, saj služi za lažje definiranje igre (Kanode in Haddad, 2009, str. 261). Definiranje igre je mišljeno kot odkrivanje zabave, ki je osrednja zahteva pri razvoju iger, za katero pa ni merila (Koutonen in Leppänen, 2013, str. 3). Zabavo je potrebno validirati na vsakem koraku razvojnega procesa, kar vodi v apliciranje visoko iterativnega upravljanja procesa (prav tam). To potrjuje dejstvo, da razvoj iger sloni na iteracijah (Novak, 2012, str. 366). Manjuka in kolegi potrjujejo, da agilne prakse dobro delujejo v fazi predprodukcije zaradi hitrih iteracij in prototipiranja (Manjuka, Chakradhar Raju in Sai Chand, 2016, 558). Kanode in Haddad poleg tega predlagata uporabo enostavnih agilnih metod v fazi predprodukcije, ki omogočajo eksploracijo igralnosti in uporabniških interakcij (Kanodein Haddad, 2009, str. 261). Za izboljšanje procesa se priporoča združevanje razvojnih procesov (Manjuka in drugi, 2016, 558). Podobno meni tudi (Keith, 2010, str. 298), ki navaja, da je Scrum primeren za predprodukcijo, medtem ko priporoča Lean in Kanban pristope v produkcijski fazi (prav tam). V produkciji je mogoče uporabiti tudi bolj formalen proces, ker je bila večina eksperimentiranja v predprodukcijski fazi že zaključenega (Kanode in Haddad, 2009, str. 264). Zanimivost razvoja iger je tudi v življenjskem ciklu razvoja. Življenjski cikel se nikoli ne konča (Bartle, 2003, str. 93). Virtualni svet se po koncu produkcije posodablja (prav tam). V tem času se izdajajo popravki, posodobitve in razširitve v glavnem z namenom daljšanja življenjske dobe igre (Novak, 2012, str. 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, str. 54). Prav tako sodelujejo z modificiranjem[35] igre, s čimer se vključijo v primitiven razvojni proces (Cooper in Scacchi, 2015, str. 5). Glavna razlika med razvojem programske opreme in razvojem igre je v razširjeni uporabi in integraciji multimedijskih sredstev (Kanode in Haddad, 2009, str. 261). Kanode in Haddad jasno navajata, da razvoj video iger potrebuje inženirsko prakso, ki bo znala upoštevati specifične karakteristike upravljanja multimedijskih sredstev in privlačnega igranja (prav tam). Za 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 (prav tam).

[35] Modding. Razvijalci izdajo orodja za modificiranje igre, s katerimi uporabniki manipulirajo za ustvarjanje novih vsebin v igri. Za začetnika moddinga se smatrajo razvijalci igre Doom.

Potek dela umetnikov in programerjev je različen in potrebuje natančno sinhronizacijo (Ruonala, 2016, str. 29). Zato Manjuka in kolegi priporočajo kaskadni model za izdelavo sredstev, ker so za njih zahteve jasno definirane, kar pa pokrije samo upravljanje kreativnega procesa in ne sinhronizacije z inženiringom (Manjuka in drugi, 2016, str. 558). 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 podobni hibridi, ki slonijo na iteracijah. Iterativen in inkrementalen razvoj predstavljata temeljne principe agilnega modeliranja, ki je metoda za izgradnjo agilnih procesnih modelov (Ambler, 2002, str. 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, str. 32). To pomeni, da mora biti proces dokumentiran in frekvenčno revidiran (prav tam). AM pa je bolj produkcijska filozofija kot trdna zbirka pravil (Unger in Novak, 2011, str. 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, str. 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, str. 419).

5.1 AGILNO MODELIRANJE

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

Iterativno in inkrementalno kategorijo definirajo prakse:

- apliciranja pravih artefaktov,
- izdelave več vzporednih modelov,
- iteracije do naslednjega artefakta,
- izvajanja manjših inkrementov (prav tam, str. 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 učinkovito uporabiti diagram kot pa napisati 1024 vrstic kode (prav tam, 46).

Vzporedno modeliranje omogoča simultani zajem informacij več različnih artefaktov (Ambler, 2002, str. 48). V primeru, da informacije za določen artefakt postanejo neprimerne, se izvede iteracija do naslednjega artefakta (prav tam).

S tem se informacije prenesejo na drug artefakt in se posledično omogoči 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, str. 180).

Skupinsko delo definirajo prakse:

- skupinsko modeliranje,
- aktivna participacija vlagateljev,
- kolektivno lastništvo,
- javni prikaz modelov (Ambler, 2002, str. 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 (prav tam, str. 52). Aktivna participacija spodbuja prisotnost uporabnikov ali vlagateljev na lokaciji z namenom izmenjave informacij glede zahtev in pričakovanj projekta (prav tam, str. 53). Za kolektivno lastništvo velja, da lahko vsakdo sodeluje pri izgradnji določenega modela, kar predstavlja priložnost večje identifikacije napak (prav tam, str. 54). V ta namen se uporabljajo tudi table za javni prikaz modelov, kar spodbuja odprto komunikacijo med zaposlenimi (prav tam, str. 55).

Enostavnost definirajo prakse:

- ustvarjanja preproste vsebine,
- enostavnega prikaza modela,
- uporabe enostavnih orodij (prav tam, str. 45).

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

Validacijo definirajo prakse:

- upoštevanja preverljivosti,
- dokazovanja s kodo (prav tam, str. 45).

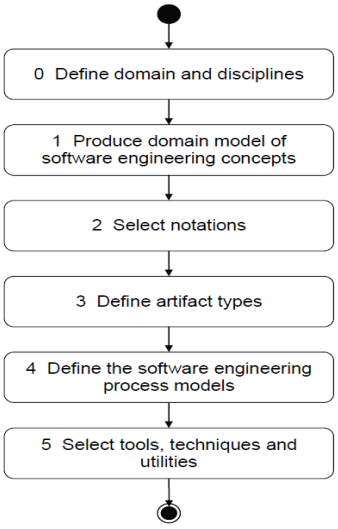
Moderni procesi vključujejo aktivnosti testiranja in zagotavljanje kvalitete čez celoten življenjski cikel. Nekateri modeli celo priporočajo definiranje testnih konceptov pred izgradnjo samega sistema (prav tam, str. 58). K temu stremi praksa upoštevanja preverljivosti. Ob zgrajenem modelu pa je potrebno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

AM podaja tudi dopolnilne prakse, ki jih lahko poljubno apliciramo in se delijo v tri kategorije: produktivnost, dokumentacija in motivacija (prav tam, 60). Vključujejo apliciranje standardov modeliranja, med katerimi je najbolj poznan UML (prav tam, 61). Spodbujajo postopno vpeljevanje in uporabo že obstoječih modelov (prav tam, 64) in modeliranje z namenom razumevanja problema in izboljšanja komunikacije (prav tam, 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, str. 429). Discipline lahko predstavljajo korake ali faze procesa (zahteve, analiza, razvoj, itd.) (Engels in Sauer, 2010, str. 429–430). Nato je zgrajen model iz konceptov ali temeljnih aktivnosti domene, ki so organizirane na podlagi disciplin (Engels in Sauer, 2010, str. 429). 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, str. 430). Naslednji korak predstavlja definiranje artefaktov (prav tam). V tem koraku se jeziki in njegovi elementi pripišejo konceptom z namenom izražanja njihovih lastnosti (prav tam). Medtem ko domenski model predstavlja semantično domeno (pomen) konceptov, jeziki predstavljajo sintaktično domeno (oblika). Hierarhiji domenskega modela konceptov in artefaktov morata biti kompatibilni (prav tam, 431). Temu sledi definiranje procesnega modela, pri katerem je potrebno definirati aktivnosti, ki so potrebne za izpolnitev nalog (prav tam). Procesni model je strukturiran iz aktivnosti, mejnikov in elementov nadzora toka (prav tam). Zadnji korak predstavlja izbira orodij, tehnik in pripomočkov skupaj s koncepti uporabe, ki so potrebni za usmerjanje in poenostavitev izgradnje artefaktov (prav tam).

Slika 5.1: Temeljni proces metode MetaME za izgradnjo procesnega modela

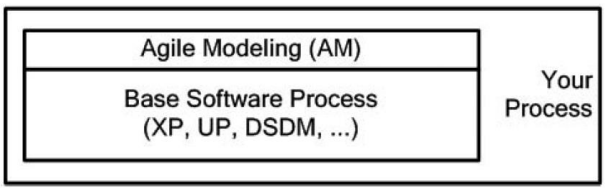


Vir: Engels in Sauer (2010, str. 430).

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 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, str. 10).

5.4 OPTIMIZACIJA PROCESA

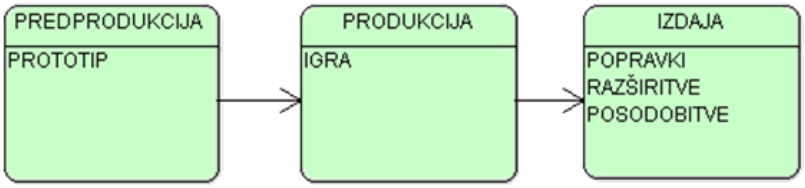
Ker želimo v sklopu RV2 ugotoviti, ali predlagani model ustreza optimizaciji z DGMM, bomo po izvedbi izgradnje predloga procesnega modela izvedli simulacijo optimizacije s prvim zrelostnim pristopom za igre DGMM. Slednji upošteva faktorje, ki temeljijo na razvojni, potrošniški in poslovni perspektivi (Aleem, Capretz in Ahmed, 2016a, str. 57–58). Na podlagi teh perspektiv so avtorji definirali 18 različnih faktorjev (PRILOGA J), ki jih imenujejo procesne aktivnosti razvoja video iger (GDPA)[36] (prav tam, str. 59). Zrelost procesa razdelijo na pet stopenj (naraščajoče): ad-hoc, priložnosten, konsistenten, organiziran in optimiziran (prav tam, str. 61). Vsako stopnjo zrelosti se preverja z namenskim vprašalnikom, katerega izjave se nanašajo na izvajanje GDPA (prav tam). Ocenjevanje zrelosti poteka tako, da posamezno izjavo (PRILOGA L) potrdimo, če model zagotavlja njeno izvajanje in zavrnemo, če model izvajanja ne podpira (prav tam). Na koncu seštejemo število potrjenih izjav in s tem dobimo število[37] GDPA, ki se izvajajo (prav tam). Če je število izvajanih aktivnosti večje od določenega praga[38], pomeni, da je procesni model dosegel zrelosti na stopnji, ki smo jo preverjali (prav tam). Za doseganje praga zrelosti velja, da se v procesu izvaja 80 % aktivnosti glede na izjave, ki so podane za določeno stopnjo zrelosti (PRILOGA K) (prav tam, str. 68).

[36] *Game Development Process Activities*.
[37] NA – *number of applicable statements*.
[38] PT – *Passing treshold*.

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, str. 95). Faze, ki jih posamezni avtorji navajajo, so prikazane v PRILOGI E. Na podlagi literature smo konsolidirali faze razvoja iger in jih kronološko razvrstili: formalizacija problema, razvoj koncepta, predprodukcija, produkcija, testiranje, postprodukcija.

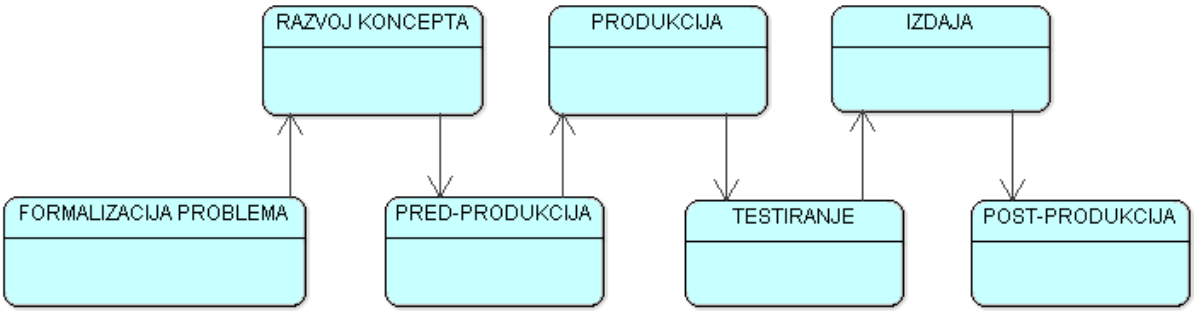
6.2 IZGRADNJA DOMENSKEGA MODELA

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 jo imenujeta formulacija problema (Aslan in Balci, 2015, str. 310). Ta faza definira probleme različnih domen in načine reševanja, ki privedejo do efektivnega učenja na podlagi iger (prav tam, str. 311). 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 (prav tam), ki pa predstavlja prvo fazo po Bates in Novak (Bates, 2004, str. 203; Novak, 2012, str. 352).

Widyani to fazo imenuje iniciacija (Widyani, 2013, str. 98). V tej fazi je predstavljena ideja igre v pisni obliki (Novak, 2012, str. 352). Čeprav Unger in Novak označujeta predprodukcijo za prvo fazo, pa navajata, da se začne po uspešno sprejetem konceptu (Unger in Novak, 2011, str. 176). Zato bomo predprodukcijo kronološko postavili za fazo razvoja koncepta.

Predprodukcija je faza, kjer se sprejme odločitev, kakšna bo igra in kako bo izdelana (Unger Novak, 2011, str. 176). Predstavlja fazo načrtovanja (prav tam, str. 353). Vrhunec predprodukcije predstavlja dokaz koncepta, ki ga predstavlja prototip. Namen je dokazati zmožnost premagovanja tehničnih zahtevnosti (Bates, 2004, str. 88). Po potrditvi prototipa se lahko začne produkcija, ki predstavlja fazo razvoja igre (Novak, 2012, str. 358). Produkcija je kompleksna in časovno potratna (Bates, 2004, str. 212). Za uspešno izvajanje faze je priporočena fragmentacija opravil v manjše naloge, ki se jim se strogo sledi (prav tam). 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, str. 259). Testiranje se deli na alfa in beta faze (prav tam). V razvoju mobilnih iger sta fazi navadno združeni (Unger in Novak, 2011, str. 176). Alfa faza predstavlja točko, v kateri je mogoče testirati celotno igranje (prav tam). Testna ekipa v tej fazi testira vse module igre, zapiše defekte v bazo in ustvari testni načrt (prav tam). Beta faza je namenjena popravilu defektov in implementaciji vseh sredstev (zvok, umetnine) v igro (prav tam). 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, str. 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, str. 362). Po fazi izdaje sledi postprodukcijska faza. Faza predstavlja nadaljevalno produkcijo, ki jo Fullerton imenuje tudi vzdrževanje (Fullerton, 2014, str. 414). 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, str. 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	IME STANJA
Vilice ali združitev	Deli in združuje dejavnosti	▬

Vir: Rumbaugh in drugi (2004, str. 40–90).

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, str. 310).
Nato sledi faza razvoja koncepta, ki ima namen definiranja osrednje funkcionalnosti igre, predstavitve grafičnega izgleda in zgodbe (Bates, 2004, str. 203). Dokumenti, ki jih razvoj koncepta proizvede, so: višji koncept, predlog igre ("pitch doc") in koncept (prav tam). Vsak tip dokumenta, ki predstavlja koncept, ima svoj namen, zato ni potrebno, da uporabimo vse. V našem primeru bomo uporabili poimenovanja po Novak, ki končni dokument koncepta imenuje predlog igre (Novak, 2012, str. 387). Predlog igre definira: zgodbo, like, koncepte umetnin, žanr, igralni pogon, igranje, trg, sestavo ekipe in analizo tveganj (Bates, 2004, str. 204–205; Novak, 2012, str. 387). Če 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, str. 390).

Na podlagi predloga igre se izdela prvi prototip[39] igre. Predstavlja najpomembnejši material, ki ga lahko proizvedemo v tej fazi (Fullerton, 2014, str. 486). Velja za dokaz delovanja in zmožnost premagovanja tehničnih zahtevnosti (Bartle, 2003, str. 88). Če je dokaz koncepta uspešen, se začne faza oblikovanja ali predprodukcija (Novak, 2012, str. 353). Cilj je izdelava GDD, TDD, UB, definiranje produkcijske poti in stvaritev projektnega načrta (Novak, 2012, str. 353). GDD pa je najobsežnejši dokument v razvoju igre (prav tam, str. 391). Novak navaja, da se ta dokument konstantno posodablja (prav tam). Schell dodaja, da se ga občasno posodablja in je ponavadi neurejen (Schell, 2008, str. 382). Na polovici projekta se ga opusti, saj vsebuje igra sama vse pomembne podrobnosti (prav tam). Na podlagi GDD tehnična ekipa izdela TDD (Novak, 2012, str. 393). Ta dokument predstavlja produkcijsko pot, ki vzpostavi načrt, kako se bo razvoj premaknil od koncepta do programske opreme (prav tam). 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) (Bates, 2004, str. 209; Novak, 2012, str. 394). Zadnje dejanje predprodukcijske faze predstavlja izdelava prototipa (Bates, 2004, str. 207–208). Ta artefakt je lahko analogne ali digitalne oblike. Pred izdelavo digitalnega je dobro izdelati enostaven analognegi prototip, ki ima namen zagotoviti zabavno in prepričljivo igralno mehaniko.

[39] *Proof of concept.*

Končan prototip je lahko dokaz delovanja produkcijske poti od ideje do realizacije (Novak, 2012, str. 354). Za kreiranje prototipa se navadno uporablja tehnika vertikalnega reza[40], ki pomeni, da vsebuje ravno toliko vsebine, da poda jasen občutek igranja in takšni ponavadi vsebujejo veliko začasnih sredstev (Mitchell, 2012, str. 64).

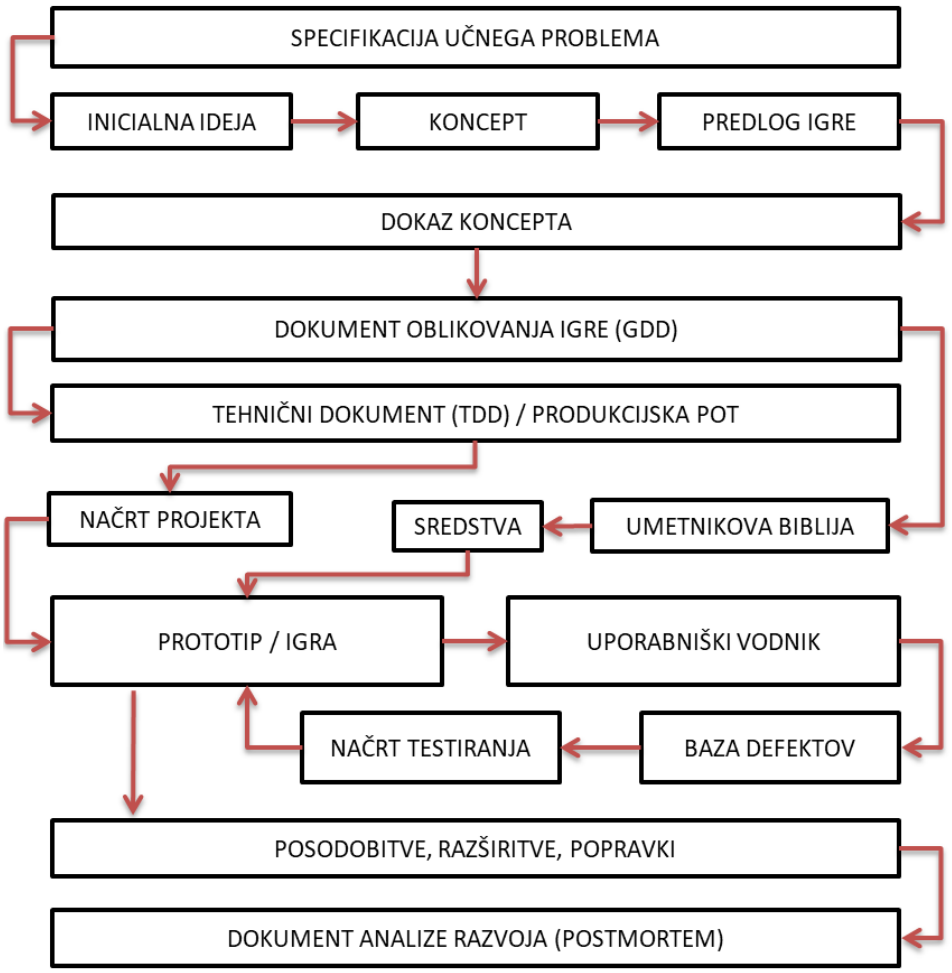
Nato sledi faza produkcije. V produkciji se kreirajo: različice igre (prototipi), uporabniški vodnik, UB in sredstva (Mitchell, 2012, str. 218). Za razvoj igre se priporoča pristop, ki je fokusiran na igralca (Adams, 2013, str. 49; Fullerton, 2014, str. 10). To pomeni, da s pomočjo introspekcije (Schell, 2008, str. 14) pred kodiranjem definiramo izkušnjo, ki jo želimo podoživeti ob igranju. To izvajamo iterativno z izdajanjem različic. Različica zajema vsa sredstva, ki so na razpolago, in predstavlja verzijo igre, ki jo lahko pregledamo in testiramo (Mitchell, 2012, str. 218). Izdelava UB se lahko začne že v predprodukciji, vendar po definiranju žanra v GDD (Novak, 2012, str. 393). Navadno se UB gradi iterativno v času produkcije, ker se jo v fazi predprodukcije ne potrebuje (Bartle, 2003, str. 88). UB služi za definiranje umetniškega sloga igre (Bates, 2004, str. 208) na podlagi katerega se v tej fazi prične razvoj končnih sredstev (Fullerton, 2014, str. 414).

Dopol njevanje priročnika je priporočljivo vzporedno z razvojem (Rucker, 2002, str. 6). Vsebuje razlago igre, vodnika po namestitvi, hiter začetek in detajlno razlago funkcionalnosti uporabniškega vmesnika (prav tam).

Produkciji sledi faza testiranja. Ta faza predstavlja zagotavljanje kakovosti[41] in skrbi, da igra zadovoljuje vse potrebe, preden se izda na trg (Levy in Novak, 2009, str. 57). V tem koraku se izdelata podatkovna baza defektov in načrt testiranja (prav tam). Po testiranju sledi faza postprodukcije. Artefakte v tej fazi predstavljajo: popravki, posodobitve in razširitve (Novak, 2012, str. 365). Popravki so brezplačne verzije in so ustvarjene z apliciranjem popravkov na originalno različico izdaje (prav tam). Popravki so lahko aplicirani tudi z namenom reševanja ostalih produkcijskih defektov (prav tam). Posodobitve izboljšujejo originalno različico igre (prav tam). Te so večinoma ustvarjene z namenom podaljšanja življenjske dobe igre (prav tam). Razširitve lahko delujejo kot samostojne igre ali pa potrebujejo originalno igro za delovanje (prav tam). Po izdaji igre se izdela dokument analize razvoja, ki pripomore k izboljšanju procesa v prihodnjih projektih. McAllister in White zaradi težav analiziranja podatkov priporočata standardizacijo tega dokumenta (McAllister in White, 2015, str. 20). Slika 6.3 prikazuje model artefaktov, ki je ustrezno kronološko razvrščen.

[40] Vertical slice.
[41] QA- *Quality assurance*.

Slika 6.3: Model artefaktov pri razvoju video iger, zgrajen na podlagi besedil 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, str. 49). Uporaben je tudi za dokazovanje izvedljivosti idej, saj predstavlja delujoči model ideje. Z njim si omogočimo formalizacijo ideje in izolacijo problemov (Fullerton, 2014, str. 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, str. 49).

Proces se nadaljuje v želeno stanje, ki predstavlja predprodukcijo. Ta se navadno konča, ko je investitor že videl igralno različico igre in je zadovoljen z ekipo in delom (Adams, 2013, str. 46). Če investitor projektu da zeleno luč, se začne produkcijska faza (prav tam).

Produkcijska faza se odvija med aktivnostmi načrtovanja, kodiranja in evalvacije (Novak, 2012, str. 367). Fullerton jo opisuje kot iterativni proces med testiranjem igranja, evalvacije in razvoja (Fullerton, 2014, str. 272). 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, str. 367). Vsaka iteracija mora imeti svoj cikel razvoja skupaj z načrtovanjem potreb, artefaktov in urnika (prav tam), 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, postprodukcija).

Posebnost predlaganega procesnega modela je implementacija toka razvoja sredstev, kar priporoča tudi Bates (Bates, 2004, str. 171). Adams potrjuje, da se to prične izvajati v fazi elaboracije (produkcija) (Adams, 2013, str. 13), to potrjuje tudi (Mitchell, 2012, str. 85). Sredstva se proizvajajo do faze testiranja. Ponavadi v alfa fazi še niso povsem izdelana (Bates, 2004, str. 214; Novak, 2012, str. 359), vendar morajo biti do začetka beta faze permanentna, drugače se beta ne izvede (Levy in Novak, 2009, str. 53). Mesta neizdelanih sredstev zaradi razvoja zapolnjujejo začasna sredstva[42] (Levy in Novak, 2009, str. 52; McAllister in White, 2015, str. 16). Začasna sredstva se med razvojem pogosto uporablja za testiranje uporabnosti brez distrakcije estetike (Novak, 2012, str. 261). Takšna praksa se pogosto uporablja in je priporočljiva, saj onemogoča ozka grla produkcije (Zagal in Altizer, 2015, str. 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, str. 79). Še težje pa je odkriti manjkajočo zvočno podlago (prav tam).

Za izdelavo sredstev se priporoča linearni proces z izboljševanjem serije prototipov. To vodi v ponavljajoč kaskadni način razvoja sredstev (Bates, 2004, str. 227), kar priporočajo tudi Manjuka in kolegi (Manjuka in drugi, 2016, str. 558).

[42] Placeholders.

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, str. 85). Po sprejetju koncepta se proces nadaljuje v aktivnost izvedbe, kjer se sredstvo izdela (prav tam, str. 103). Izdelano sredstvo se preveri s testiranjem v različici igre (Levy in Novak, 2009, str. 57). 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 G – 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, str. 5). Slabo nadzorovane spremembe v dokumentaciji lahko prerastejo v večje težave, ki lahko negativno vplivajo na funkcionalnost igre, sredstva ali časovne mejnike (prav tam, str. 5). Priporoča se 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, str. 5), predvideva naš model sprotno posodobitev dokumentacije po vsakem inkrementu igre ali izdelavi 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 (III, 2004, str. 319). Ne glede na izkušnje ekipe dobra dokumentacija pripomore k boljšem razvoju (prav tam).

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, str. 264; Manjuka in drugi, 2016, str. 557–558; Ruonala, 2016, str. 9). Predlagani procesni model to omogoča, saj izbor agilnih praks v fazi načrtovanja iteracije ni nujen. Za lažje definiranje dokumentacije priporočamo uporabo osnutkov. V PRILOGI G je primer osnutka GDD. Na sliki 6.4 modra stanja predstavljajo faze testiranja.

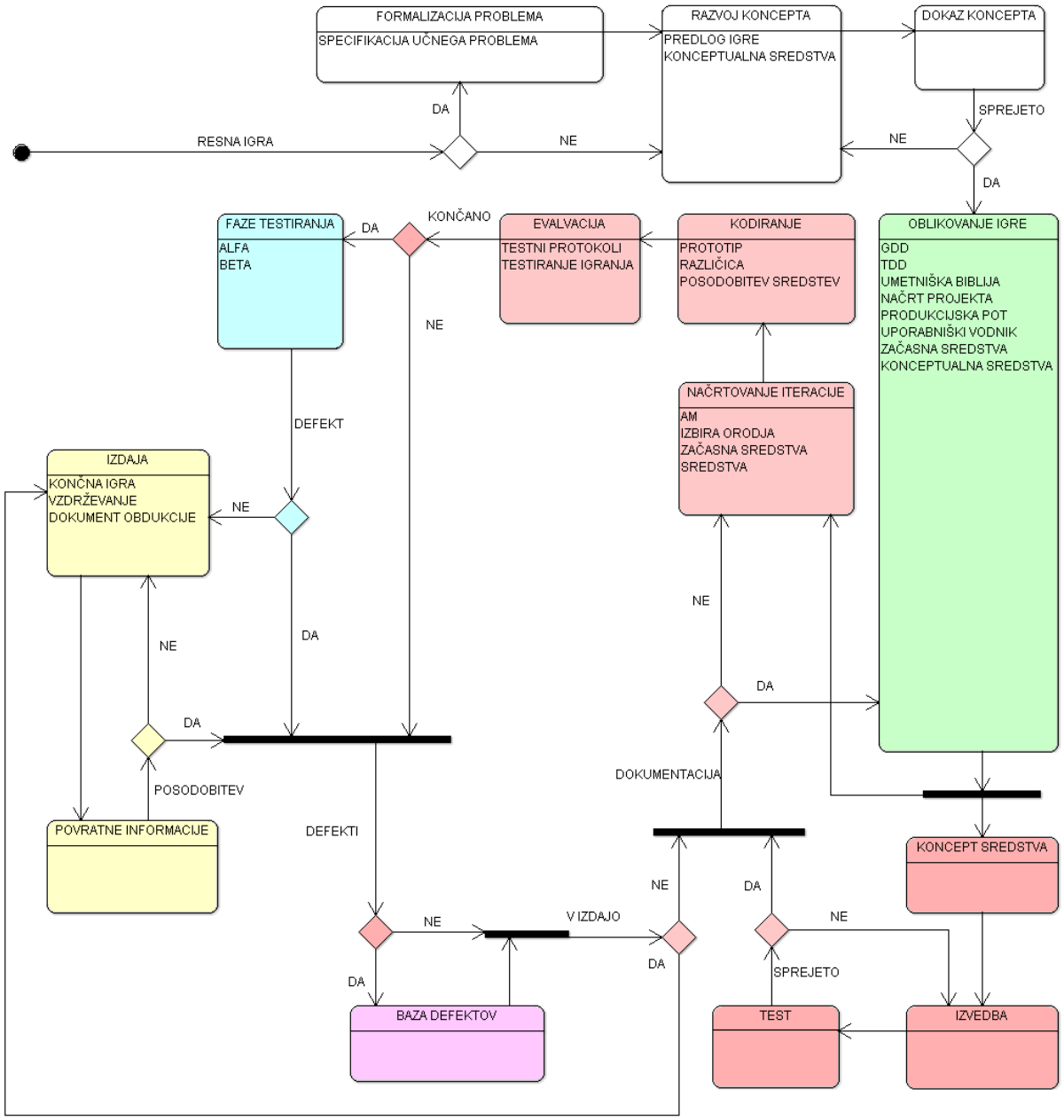
V fazo testiranja preidemo, ko igra predstavlja končni izdelek ali če nas obvezujejo kakršnekoli časovne ali pogodbene obveznosti. Ta faza se deli na alfa in beta fazo testiranja. Alfa stremi k

zaklepanju funkcionalnosti, medtem ko se beta faza osredotoča na poliranje igre in reševanje defektov (Levy in Novak, 2009, str. 52–53).

V obeh fazah se proces nadaljuje iterativno, dokler obstajajo defekti ali dokler ti niso označeni za nepomembne (WNF, NAB)[43] (prav tam, str. 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 neizogibna. Vzrok ne leži nujno v rani izdaji, temveč v tisoče različnih strojnih konfiguracijah, ki jih je nemogoče v celoti predvideti in testirati (Bates, 2004, str. 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, str. 421).

[43] WNF - *Will Not Fix*, pomeni, da defekt ni dovolj pomemben. NAB - *Not a bug*, pomeni, da ni defekt.

Slika 6.4: Predlagani procesni model za video igre, zgrajen na podlagi združevanja procesnega in artefaktnega modela po meta-metodi MetaME

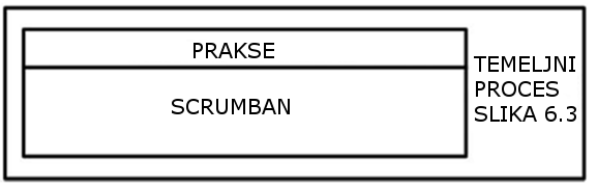


6.6 IZBIRA ORODIJ

Izbira se odvija v času produkcije ali postprodukcije (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, str. 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 H. 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 (prav tam, str. 10).

Slika 6.5: Abstrakcija aplikacije predlaganega procesnega modela



V razvoju video iger obstajajo tehnične in netehnične usmeritve, ki omogočajo uspeh projektov. Tehnične so zagotoviti: tehnični, časovni in projektni načrt, razumevanje tveganj in znanje ponastavitve projekta v težavah (Bates, 2004, str. 219). Netehnične predstavljajo: vzdrževanje komunikacije, sledenje stroškom, vzdrževanje ekipnega duha in identitete (prav tam, str. 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 (prav tam, str. 212). Za takšno opravilo je priporočljiva uporaba Scrum prakse, fragmentacije produkta na obvladujoče entitete (Sommerville, 2010, str. 74). Vendar Scrum ne definira tehničnih aspektov agilnega pristopa, ampak se osredotoča le na upravljanje inkrementalnega razvoja (prav tam, str. 72), zato so najverjetneje podjetja Scrum metodologijo posvojila v različnih oblikah.

Dr. Lennart E. Nacke za Novak 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" (Novak, 2012, str. 373). To nakazuje na uporabo hibrida Scrum in Kanban metodologije, imenovano Scrumban, ki jo predstavita Yilmaz in O'Connor. S Kanban dimenzijo metodologije vsak trenutek vemo, koliko dela je v obtoku (Yilmaz in O'Connor, 2016, str. 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, ki jih nadgradi v polstrukturirane intervjuje (prav tam). V realnosti veliko razvojnih ekip zgreši pri napovedi trajanja projekta (Novak, 2012, str. 358). Zavoljo tega se proti koncu produkcijskega cikla velikokrat pojavi čas krize[44] (prav tam). Z uporabo praks tega procesnega modela pridobimo na reševanju težav slabih časovnih napovedi razvoja. Kot podporo Scrumban predlagamo agilni praksi XP pristopa: konstantno prisotnega naročnika (vlagatelja) in fiksen 40-urni tedenski delavnik, 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 od enega do treh razvijalcev pri majhnih neodvisnih razvojnih timih, medtem ko lahko ekipa šteje več sto zaposlenih pri velikih organizacijah (Mitchell, 2012, str. 56). Čeprav je v teoriji mogoče izdelati igro le z umetnikom in programerjem, je potrebno veliko več za sestavo temeljev uspešne igre (Novak, 2012, str. 319). Vsako igro je potrebno upravljati, oblikovati, programirati, testirati. Prav tako igra potrebuje zvok, umetnine in glasbo (Bates, 2004, str. 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, (Bates, 2004, str. 153–179; Novak, 2012, str. 319–339), pri čemer lahko glede na velikost ekipe ena oseba pokriva več vlog (Bates, 2004, str. 151; Novak, 2012, str. 319).

[44] Crunch time.

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, str. 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 (prav tam, str. 376–379). Proces je priporočeno podpreti tudi z uporabo digitalnih programskih paketov. Ravnateljstvo že dolgo uporablja takšna orodja za administracijo individualnih ali skupine projektov (Ahmad in Laplante, 2006, str. 76). Orodja kot so: Blosson, Breeze, Kanbanize, Scrumwise, Kanbanery, Yodiz, ZenHub, Leankit, Jira so idealna za spremljanje nalog, saj omogočajo kreiranje Kanban tabel ("Zapier", 2017; "LifeWire", 2018; "Pcmag", 2017; "Technologyadvice", 2017; "Agilescout", 2010; "Medium", 2016).

Tabela 6.2: Orodja za kreiranje kanban tabel, ki se najpogosteje priporočajo

zapier	lifewire	pcmag	technologyadvice	Agile scout	medium
Trello	GreenHopper	Asana	Scrumwise	AgileZen	ZenHub
Meister task	Kanbanize	Wrike	Kanbanery	Axosoft	Trello
LeanKit	KanbanFlow	LeanKit	Volerro	Blossom	Asana
Kanban Tool	Kanban Tool	Volerro	LeanKit	Flow	Swift
KanbanFlow	LeanKit	Trello	Zenhub	FogBugz	Kanban Tool
Blossom	Trello	KanbanFlow	Trello	Hansoft	Virtual Kanban
Breeze	Volerro	ZenKit	Jira	Jam Circle	
Monday				Kanban Tool	
Pipefy				LeanKit	

Vir: "Zapier" (2017); "LifeWire" (2018); "Pcmag" (2017); "Technologyadvice" (2017); "Agilescout" (2010); "Medium" (2016).

Tabela 6.2 prikazuje kanban orodja, ki se najpogosteje priporočajo. Z zeleno so označena orodja, ki se priporočajo pogosteje. Zgornja vrstica predstavlja vire (spletne strani končnice .com). Poleg teh orodij se v veliki meri uporabljajo HacknPlan, Hubot, Trello, Asana in Slack (Lin in drugi, 2006, str. 333, str. 2; "Gamasutra", 2017) . Orodja lahko omogočajo tudi definiranje t.i. Gantt[45] ali PERT[46] grafov, ki so primerni za definiranje časovnih načrtov (Baars, 2006, str. 15). Novi paketi orodij prinašajo prednosti, saj omogočajo upravljanje s tveganji, upravljanje z najboljšimi praksami, e-poštne notifikacije in kolaboracije (Lin in drugi, 2016, str. 333–336).

Za izbiro ustreznega programskega paketa predlagamo analitični hierarhični proces[47], ki omogoča izbiro paketa, na podlagi teorije izbire po Saatyu. Primarna naloga postopka je kvantifikacija relativnih prioritet za podano zbirko alternativ (PRILOGA I) na podlagi sodb, ki jih poda odločevalec (Ahmad in Laplante, 2006, str. 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 (prav tam, str. 81). Ker lahko projekti zaradi količine defektov postanejo neobvladljivi, se priporočajo rešitve za spremljanje izvorne kode (McShaffry in Graham, 2012, str. 111). Komercialne rešitve predstavljajo SourceSafe, Perforce in AlienBrain, odprtokodne pa Subversion, TortoiseSVN in Git (McShaffry in Graham, 2012, str. 15–114).

[45] Je grafikon, ki predstavlja časovni razpored projekta.
[46] Program evaluation and review technique, je statistično orodje za analizo nalog, ki so vključene v razvoj.
[47] AHP – *Analytical Hierarchy Process*.

6.7. 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 L) za ocenjevanje optimalne zrelosti po DGMM, podali odgovore. Če predlagani model zagotavlja izvajanje posamezne GDPA, smo v tabelo zapisali DA, če pa ne zagotavlja izvajanja, smo zapisali NE. Rezultati so v PRILOGI M.

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, ki smo jo želeli doseči. Izjavi S.5.8.1 in S.5.8.2 se nanašata na aktivnosti, povezane z razvojem in vzdrževanjem igralnih pogonov. Za to nalogo je odgovoren programer (Novak, 2012, str. 334). V literaturi je razbrati, da se je pri številčno manjših ekipah bolje usmeriti na oblikovanje in izdelavo igre, kot na izgradnjo igralnega pogona (prav tam). Sama izdelava pogona, vzdrževanje in nudenje podpore več platformam je časovno in stroškovno potratno opravilo za manjše ekipe. (McShaffry in Graham, 2012, str. 22). Ti dve izjavi sta predvsem odvisni od številčnosti ekipe in njene 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, ki jo želimo doseči z iterativnim razvojem prototipov (Fullerton, 2014, str. 14).

Rezultati (PRILOGA M) 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 K). 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 za igre: ad hoc, sekvenčni, evolucijski, specializirani, agilni in procesni. 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, saj apliciranje delovnega toka razvoja sredstev v razvojni proces iger ustvari dva vzporedna procesa, ki morata biti usklajena. Takšne implementacije in sinhronizacije ne predvideva noben pregledan proces. Predvidevamo, da to povzroča ozka grla, ki so nato vzrok za nastanek najpogostejših zapletov v razvoju (pog. 4.1).

Zato smo v skopu raziskovalnega vprašanja – 2 (RV2) predlagali procesni model za razvoj video iger, ki bi služil kot prikaz sistematične izgradnje procesnega modela, ki bi upošteval lastnosti, zahteve in potrebe discipline razvoja video iger. Predlagani procesni model smo sestavili z uporabo metode MetaME, ki ima to 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 predlagani procesni model služi kot rigidni delovi tok, 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, onemogočanje kriznega časa in ostale najpogostejše težave v razvoju. Posebnost predlaganega procesa je definiranje dodatnega delovnega toka izdelave kreativnih sredstev in njegovo sinhronizacijo z razvojem. Razvoj poteka nemoteno z uporabo začasnih sredstev, ki jih posodablja glede na razpoložljivost in dostavo izdelanih s strani delovnega toka sredstev. Da bi pregledali ustreznost, 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 njegove 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 napredek discipline. Disciplina razvoja video iger potrebuje več tovrstnih študij, s katerimi bo zgradila trdnejše temelje in omogočila teoretski in konceptualni napredek. S tem bo disciplina video iger pridobila terminološke temelje, ki bodo služili za opis lastnega inženiringa metodologij.

8 VIRI

1. Abrahamsson, P., Salo, O., Ronkainen, J. in Warsta, J. (2017). *Agile Software Development Methods: Review and Analysis*. Dostopno prek http://arxiv.org/abs/1709.08439

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

3. Agilescout. (2010). *Top Agile Tools - Best Kanban Tools*. Dostopno prek https://agilescout.com/best-kanban-tools

4. Ahmad, N., & Laplante, P. A. (2006). Software Project Management Tools: Making a Practical Decision Using AHP. V E. Hossain, G. Fortino, D. Heirman, X. Li, A. Molisch, S. Nahavandi, R. Perez, L. Shafer, M. Shahidehpour, D. A. Grier, J. Reed, S. Spuregon in A. M. Tekalp (ur.), *2006 30th Annual IEEE/NASA Software Engineering Workshop* (str. 76–84). https://doi.org/10.1109/SEW.2006.30

5. Aktaş, A. in Orçun, E. (2016). A survey of computer game development. V E. Hossain, G. Fortino, D. Heirman, X. Li, A. Molisch, S. Nahavandi, R. Perez, L. Shafer, M. Shahidehpour, D. A. Grier, J. Reed, S. Spuregon in A. M. Tekalp (ur.), *The Journal of Defense Modeling and Simulation*, *13*(2), (str. 239–251). Dostopno prek https://doi.org/10.1177/1548512914554405

6. Aleem, S., Capretz, L. F. in Ahmed, F. (2016a). A Digital Game Maturity Model (DGMM). V R. Nakatsu in M. Rauterberg (ur.), *Entertainment Computing*, *17*(Supplement C), (str. 55–73). https://doi.org/10.1016/j.entcom.2016.08.004

7. Aleem, S., Capretz, L. F. in Ahmed, F. (2016b). Game development software engineering process life cycle: a systematic review. V G. Alessandro, L. Murta in Hoek A. (ur.), *Journal of Software Engineering Research and Development*, *4*(1), 6. https://doi.org/10.1186/s40411-016-0032-7

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

9. Anderson, D. J. in Carmichael, A. (2016). *Essential Kanban Condensed*. Blue Hole Press.

10. Aslan, S. in Balci, O. (2015). GAMED: digital educational game development methodology. V M. Petty in G. Wainer (ur.), *SIMULATION*, *91*(4), (str. 307–319). https://doi.org/10.1177/0037549715572673

11. Babu, K. S. in Maruthi, R. (2013). Lifecycle for Game Development to Ensure Enhanced Productivity. V M. Balakrishnan,V. Balaji, I. Shanin, P. Muralidhar, P. K. D. Pramanik, R. M. F. Allwihan, C. Bhuvaneswari (ur.), *International Journal of Innovative Research in Computer and Communication Engineering*, *1*(8), (str. 1490–1503). Dostopno prek http://www.ijircce.com/upload/2013/october/36_Lifecycle.pdf

12. Barbosa, E. in Godoy A. (2010). Game-Scrum: An Approach to Agile Game Development. V R. L. S. Dazzi in A. Sena (ur.), *Proceedings of SBGames 2010*, (str. 292–295). Dostopno prek https://www.academia.edu/15250630/GameScrum_An_Approach_to_Agile_Game_Development

13. Bartle, R. (2003). *Designing Virtual Worlds*. United States: New Riders Games.

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

15. Baars, W. (2006). Project Management Handbook. Hague: Data Archiving and Networked Services.

16. Blow, J. (2004). Game Development: Harder Than You Think. V S. L. Bahra, E. Allan, P. Bailis, S. Bourne, T. Coatta, M. Compton, S. Feldman, N. Forsgren, C. Fournier, B. Fried, C. Grier, T. Killalea, T. Limoncelli, K. Matsudaira, M. K. McKusick, E. Meijer, G. Neville-Neil, T. Schlossnagle, J. Waldo, M. Whittaker. (ur.), *Queue*, *1*(10), (str. 28–37). https://doi.org/10.1145/971564.971590

17. Booth, A., Papaioannou, D. in Sutton, A. (2012). *Systematic Approaches to a Successful Literature Review*. London, Sage.

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

19. Clark, D. B., Tanner-Smith, E. E. in Killingsworth, S. S. (2016). Digital Games, Design, and Learning: A Systematic Review and Meta-Analysis. V P. K. Murphy, A. C. Dowd, G. M. Lloyd (ur.), *Review of Educational Research*, *86*(1), (str. 79–122). https://doi.org/10.3102/0034654315582065

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

21. Cooper, K. M. L. in Scacchi, W. (Ur.). (2015). *Computer Games and Software Engineering* (1 edition). Boca Raton: Chapman and Hall/CRC.

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

23. EcuRed. (b.d.). *Game unified Process*. Dostopno prek https://www.ecured.cu/Game_Unified_Process

24. Engels, G. in Sauer, S. (2010). A Meta-Method for Defining Software Engineering Methods. V G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, B. Westfechtel (ur.), *Graph Transformations and Model-Driven Engineering* (str. 411–440). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17322-6_18

25. Entertainment Software Association. (2017). *Essential Facts about the Computer and Video Game Industry*. Dostopno prek http://www.theesa.com/wp-content/uploads/2017/04/EF2017_FinalDigital.pdf

26. Eurostat. (2017). *Labour market*. Dostopno prek http://ec.europa.eu/eurostat/web/labour-market/labour-costs/database

27. Ey. (2014). *Creating Growth, Measurning curtural and creative markets in the EU*. Dostopno prek http://www.ey.com/Publication/vwLUAssets/Measuring_cultural_and_creative_markets_in_the_EU/$FILE/Creating-Growth.pdf

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

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

30. Gamasutra. (2014). Game Developer Salary Survey 2014: The results are in!. Dostopno prek https://www.gamasutra.com/view/news/221533/Game_Developer_Salary_Survey_2014_The_results_are_in.php

31. Gamasutra. (2017). Extreme Game Development: Right on Time, Every Time. Dostopno prek https://www.gamasutra.com/view/feature/131236/extreme_game_development_right_on_.php

32. Gamasutra. (2017). Finally! A project amanaging tool for Game Development!. Dostopno prek https://www.gamasutra.com/blogs/GustavoMonforte/20170213/291283/Finally_A_project_managing_tool_for_Game_Development.php

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

34. Schott G. in D. Hodgetts. (2006). Health and Digital Gaming: The Benefits of a Community of Practice. V D. F. Marks (ur.), *Journal of Health Psychology*, *11*(2), (str. 309–316). https://doi.org/10.1177/1359105306061189

35. Hamari J. in Keronen L. (2017). Why do people play games? A meta-analysis. V P. Hills (ur.), *International Journal of Information Management* 37 (3): (str. 125–41). https://doi.org/10.1016/j.ijinfomgt.2017.01.006

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

37. Jalote, P., Palit, A., Kurien, P. in Peethamber, V. T. (2004). Timeboxing: a process model for iterative software development. V P. Avgeriou in D. Shepherd (ur.), *Journal of Systems and Software*, *70*(1), (str. 117–127). https://doi.org/10.1016/S0164-1212(03)00010-4

38. Janes, A. in Succi, G. (2014). *Lean Software Development in Action* (2014 edition). New York: Springer.

39. Kanode, C. M. in Haddad, H. M. (2009). Software Engineering Challenges in Game Development. V S. A. Madani (ur.), *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations* (str. 260–265). Washington, DC, USA: IEEE Computer Society. https://doi.org/10.1109/ITNG.2009.74

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

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

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

43. Koutonen, J. in Leppänen, M. (2013). How Are Agile Methods and Practices Deployed in Video Game Development? A Survey into Finnish Game Studios. V H. Baumeister in B. Weber (ur.), *Agile Processes in Software Engineering and Extreme Programming* https://doi.org/10.1007/978-3-642-38314-4_10

44. L. Rakestraw, T., V. Eunni, R. in Kasuganti, R. (2013). The mobile apps industry: A case study. V D. Haytko, C. Sharp (ur.), *Journal of Business Cases and Applications*, *The mobile apps industry*, (str. 74–98). Dostopno prek https://www.researchgate.net/publication/257735171_The_mobile_apps_industry_A_case_study

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

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

47. Levy, Y. in J. Ellis, T. (2006). A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research. V R. G. Saadé in E. Cohen (ur.), *Informing Science: The International Journal of an Emerging Transdiscipline*, *9*, (str. 181–212). https://doi.org/10.28945/479

48. Lifewire. (2018). 7 Kanban Board Tools for Project Collaboration. Dostopno prek https://www.lifewire.com/kanban-board-tools-for-project-collaboration-771630

49. Lin, Bin, Alexey Zagalsky, Margaret-Anne Storey in Alexander Serebrenik. (2016). Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. V E. Tardos in H. McCormick (ur.), *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, (str. 333–336). CSCW '16 Companion. New York, NY, USA: ACM. https://doi.org/10.1145/2818052.2869117

50. Ma, M. in Zheng, H. (2011). Virtual Reality and Serious Games in Healthcare. V S. Brahnam in L. C. Jain (ur.), *Advanced Computational Intelligence Paradigms in Healthcare 6. Virtual Reality in Psychotherapy, Rehabilitation, and Assessment* (str. 169–192). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17824-5_9

51. Madani, K., Pierce, T. W. in Mirchi, A. (2017). Serious games on environmental management. V G. Haghighat (ur.), *Sustainable Cities and Society*, *29*, (str. 1–11). https://doi.org/10.1016/j.scs.2016.11.007

52. Manjuka, R, M Chakradhar Raju in M Sai Chand. (2016). *Software Engineering Challenges in Game Development.* Dostopno prek https://www.ijraset.com/fileserve.php ?FID=5768

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

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

55. Medium. (2016). Free Kanban board software. Dostopno prek https://medium.com/project-management-learnings/free-kanban-board-softwares-478faabf91b0

56. Mitchell, B. L. (2012). *Game Design Essentials*. John Wiley & Sons.

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

58. Musil, J., Schweda, A., Winkler, D. in Biffl, S. (2010). Improving Video Game Development: Facilitating Heterogeneous Team Collaboration through Flexible Software Processes. V A. Riel, R. O'Connor, S. Tichkiewitch, R. Messnarz (ur.), *Systems, Software and Services Process Improvement* (str. 83–94). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15666-3_8

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

60. O'Hagan, A. O., Coleman, G. in O'Connor, R. V. (2014). Software Development Processes for Games: A Systematic Literature Review. V B. Barafort, R. V. O'Connor, A. Poth in R. Messnarz (ur.), *Systems, Software and Services Process Improvement* (str. 182–193). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-43896-1_16

61. O'Hagan, A. O. in O'Connor, R. V. (2015). Towards an Understanding of Game Software Development Processes: A Case Study. V R. V. O'Connor, M. U. Akkaya, K. Kemaneci, M. Yilmaz, A. Poth in R. Messnarz (ur.), *Systems, Software and Services Process Improvement* (str. 3–16). Springer, Cham. https://doi.org/10.1007/978-3-319-24647-5_1

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

63. Pcmag. (2017). The Best Kanban Apps of 2018. Dostopno prek https://www.pcmag.com/roundup/356732/the-best-kanban-apps

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

65. Petrillo F. in Pimenta M. (2010). Is agility out there? Agile practices in game development. V E. Tardos in H. McCormick (ur.), *Proceedings of the 28th ACM International Conference on Design of Communication*, (str. 9–15). https://doi.org/10.1145/1878450.1878453

66. Petrillo F., Pimenta M., Trindade, F. in Dietrich, C. (2008). Houston, we have a problem...: a survey of actual problems in computer games development. V E. Tardos in H. McCormick (ur.), *Proceedings of the 2008 ACM symposium on Applied computing* (str. 707–711). https://doi.org/10.1145/1363686.1363854

67. Petrillo F., Pimenta M., Trindade, F. in Dietrich, C. (2009). What Went Wrong? A Survey of Problems in Game Development. V E. Tardos in H. McCormick (ur.), *Computers in Entertainment (CIE) - SPECIAL ISSUE: Media Arts and Games*, *7*(1), 13:1–13:22. https://doi.org/10.1145/1486508.1486521

68. Politowski C., Daniel V., Fontura L. in Antonio Augutso Foletto A. (2016). Software Engineering Processes in Game Development: a Survey about Brazilian Developers' Experiences. V M. S. Hounsell in F. S. Correa da Silva (ur.), *Proceedings od SBGames 2016.* Dostopno prek http://www.sbgames.org/sbgames2016/downloads/anais/157812.pdf

69. Pressman, R. S. in Maxim, B. (2014). *Software Engineering: A Practitioner's Approach* (8 edition). New York, NY: McGraw-Hill Education.

70. Ramadan, R. in Widyani, Y. (2013). Game development life cycle guidelines (str. 95–100). https://doi.org/10.1109/ICACSIS.2013.6761558

71. Rolland, C. (1998). A comprehensive view of process engineering. V B. Pernici in C. Thanos (ur.), *Advanced Information Systems Engineering* (str. 1–24). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0054216

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

73. Rumbaugh, J., Jacobson, I. in Booch, G. (2004). *The Unified Modeling Language Reference Manual*, (2 edition). Boston: Addison-Wesley Professional.

74. Ruonala, H.-R. (2016). Agile Game Development: A Systematic Literature Review.

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

76. Salmon, J. P., Dolan, S. M., Drake, R. S., Wilson, G. C., Klein, R. M. in Eskes, G. A. (2017). A survey of video game preferences in adults: Building better games for older adults. V R. Nakatsu in M. Rauterberg (ur.), *Entertainment Computing*, *21*, (str. 45–64). https://doi.org/10.1016/j.entcom.2017.04.006

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

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

79. Soomro, S., Ahmad, W. F. W. in Sulaiman, S. (2013). Evaluation of Mobile Games Using Playability Heuristics. V H. B. Zaman, P. Robinson, P. Oliver, T. K. Shih in S. Velastin (ur.) *Advances in Visual Informatics* (str. 264–274). Springer, Cham. https://doi.org/10.1007/978-3-319-02958-0_25

80. Stacey, P. in Nandhakumar, J. (2008). Opening up to agile games development. V E. Tardos in H. McCormick (ur.), *Communications of the ACM*, *51*(12), 143. https://doi.org/10.1145/1409360.1409387

81. Sylvester, T. (2013). *Designing Games: A Guide to Engineering Experiences* (1 edition). Sebastopol, CA: O'Reilly Media.

82. Technologyadvice. (2017). 8 Kanban Tools for Project Managers and Developers. Dostopno prek https://technologyadvice.com/blog/information-technology/kanban-tools-project-managers-developers

83. Tsui, F., Karam, O. in Bernal, B. (2016). *Essentials Of Software Engineering* (4 edition). Burlington, Massachusetts: Jones & Bartlett Learning.

84. Tutorialspoint. (b. d.). *SDLC - V - Model*. Dostopno prek: https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm

85. Ulbin, M. (2017). *Razvoj igre za pomoč otrokom s prekomerno težo* (diplomska naloga). Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko. Dostopno prek https://dk.um.si/IzpisGradiva.php?id=66054&lang=slv

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

87. Wilson Brotto Furtado, A. (2012.). *Domain-Specific Game Development* (doktorska disertacija). Dostopno prek https://repositorio.ufpe.br/bitstream/123456789/2165/1/arquivo9604_1.pdf

88. Yilmaz, M. in O'Connor, R. (2016). *A Scrumban Integrated Gamification Approach To Guide Software Process Improvement: A Turkish Case Study*. https://doi.org/10.17559/TV-20140922220409

89. Zagal, J. P. in Altizer, R. (2015). Placeholder Content in Game Development: Benefits and Challenges. V E. Tardos in H. McCormick (ur.), *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play* (str. 745–750). New York, NY, USA: ACM. https://doi.org/10.1145/2793107.2810319

90. Zapier. (2017). *The 11 Best Kanban Apps to Build Your Own Productivity Workflow*. Dostopno prek https://zapier.com/blog/best-kanban-apps

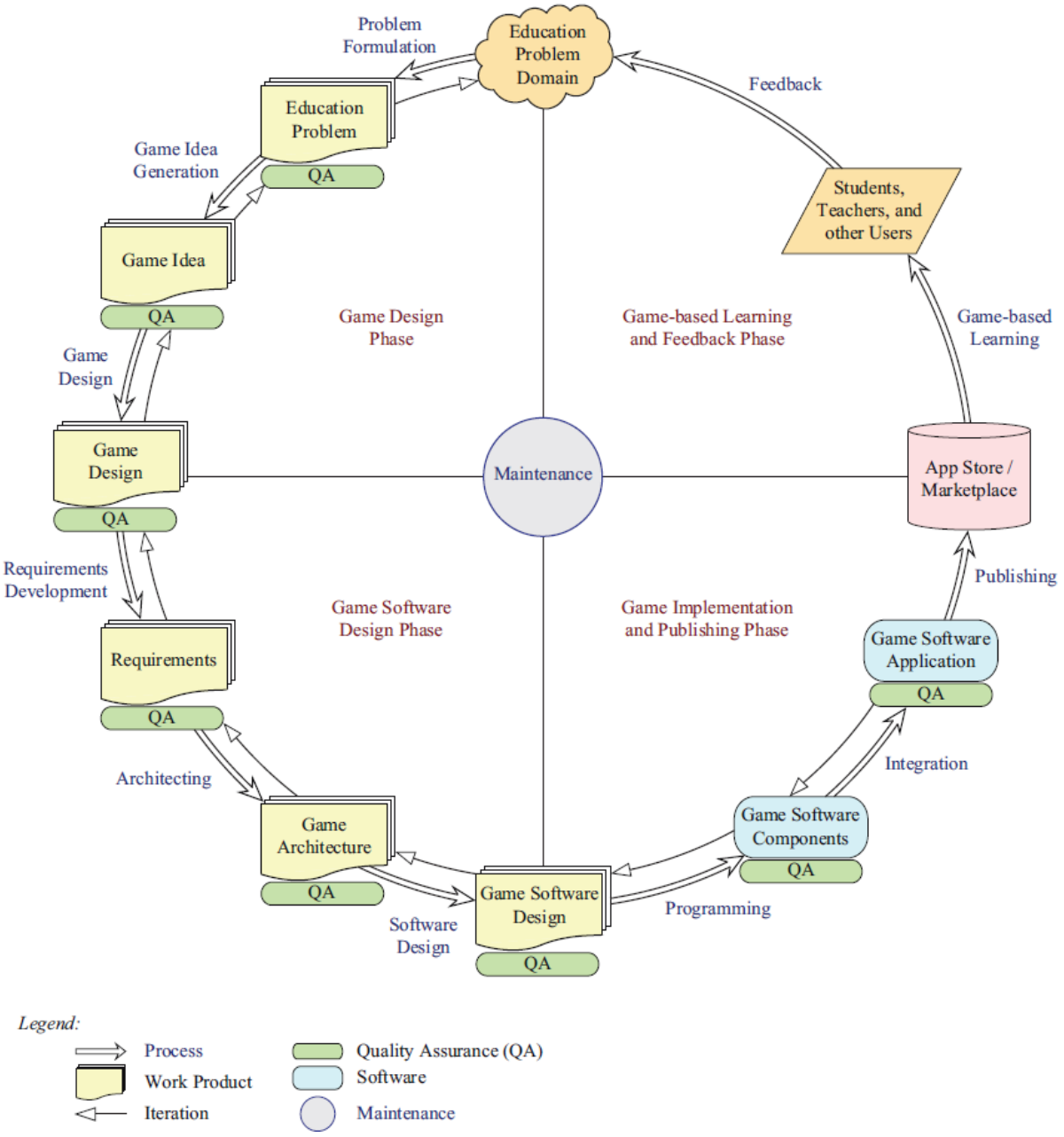
PRILOGE

PRILOGA A: ŠTEVILO TIPOV CITATOV NA LETO IZDAJE RAZISKAVE

Years														
Citation type	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Book	0	0	0	0	0	0	0	0	0	0	0	0	3	2
Journal	1	2	2	2	3	4	1	4	2	5	2	2	4	0
Conference	1	1	4	1	1	5	7	14	15	15	17	10	4	10
Workshop	0	0	0	0	0	0	0	1	1	0	0	1	0	1
Total	2	3	6	3	4	9	8	19	18	20	19	13	11	13

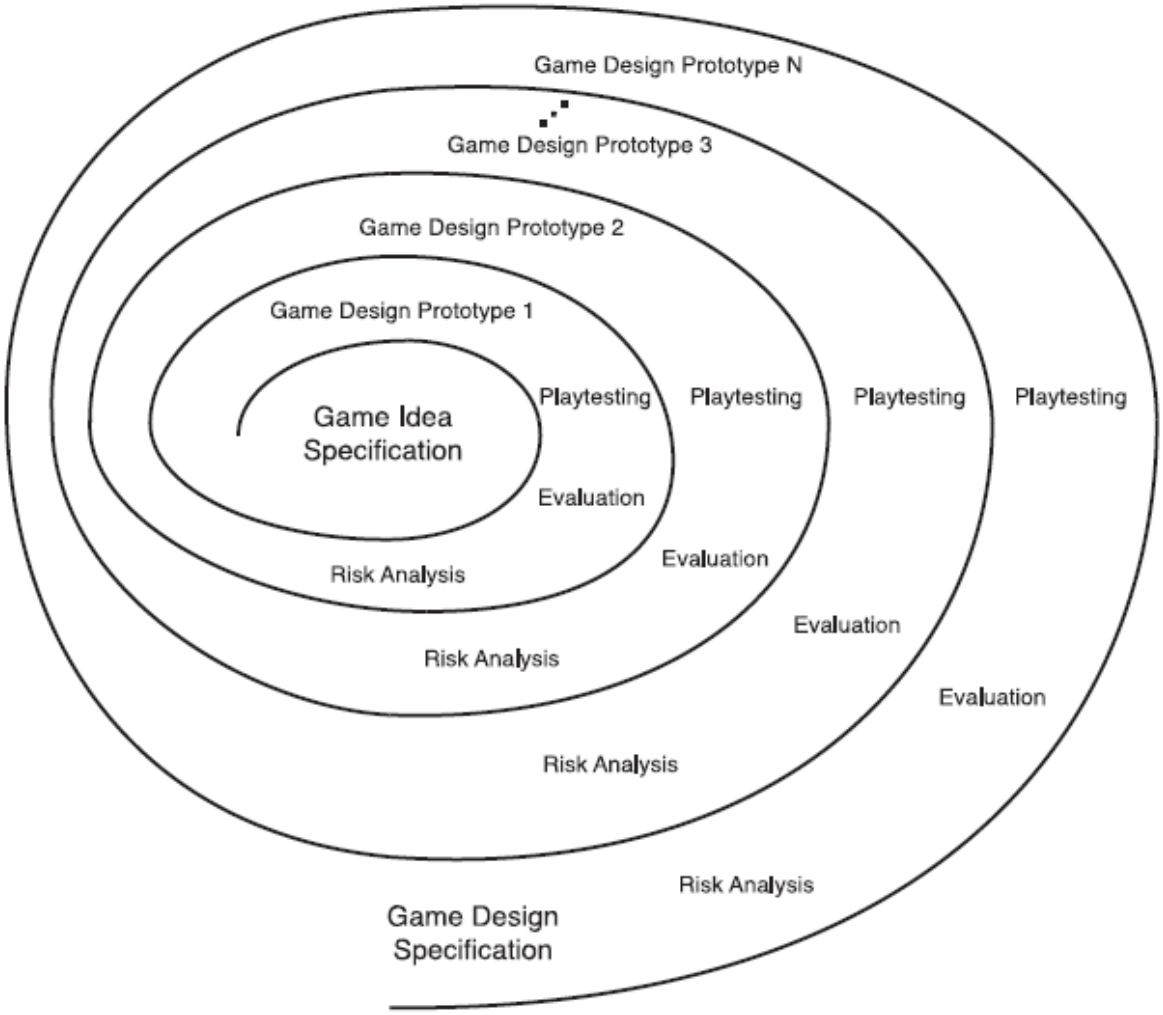
Vir: (Aleem in drugi, 2016b, str. 14).

PRILOGA B: ŽIVLJENJSKI CIKEL METODOLOGIJE ZA RESNE IGRE GAMED



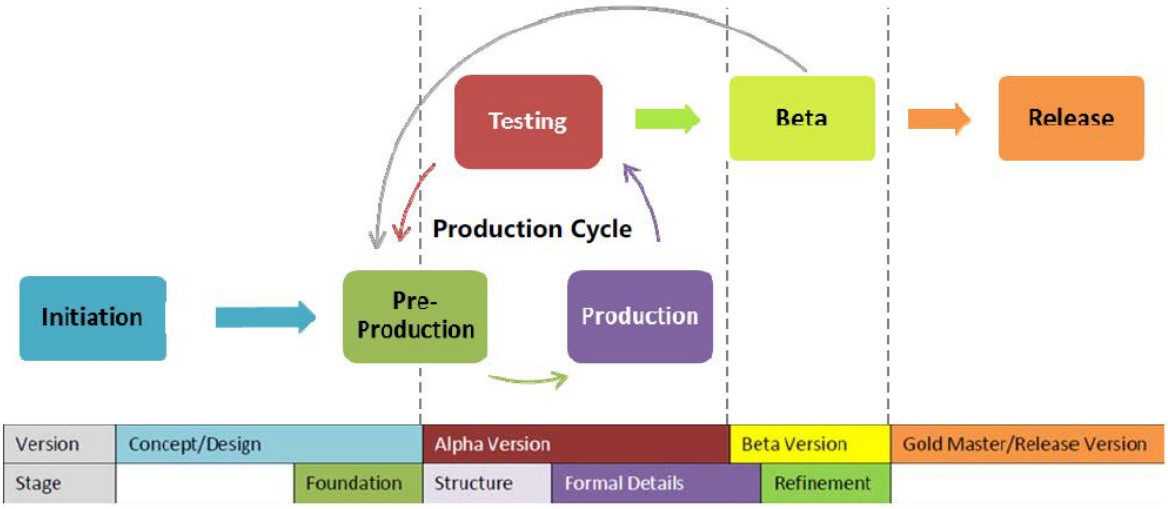
Vir: (Aslan in Balci, 2015, str. 309).

PRILOGA C: SPIRALNI MODEL ZA PODPORO OBLIKOVANJU IGRE



Vir: (Aslan in Balci, 2015, str. 313).

PRILOGA Č: PREDLAGANI ŽIVLJENJSKI CIKEL RAZVOJA ZA VIDEO IGRE



Vir: (Ramadan in Widyani, 2013, str. 98).

PRILOGA D: 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 E: PRIKAZ FAZ RAZVOJA PO AVTORJIH

AVTOR	FAZE										
Novak (2012)	KONCEPT	PREDPRODUKCIJA	PROTOTIP	PRODUKCIJA	ALFA	BETA	GOLD	POSTPRODUKCIJA			
Unger in Novak (2011)	PREDPRODUKCIJA	PRODUKCIJA	ALFA & BETA	GOLD	POSTPRODUKCIJA						
Ramadan in Widyani (2013) lasten podroben	INICIACIJA	PREDPRODUKCIJA	PRODUKCIJA	TESTIRANJE	BETA	IZDAJA					
Bates (2004)	RAZVOJ KONCEPTA	PREDPRODUKCIJA	RAZVOJ	ALFA	BETA	PREKINITEV KODIRANJA	IZDAJA	POPRAVKI	POSODOBITVE		
Aslan in Balci (2015)	FORMULACIJA PROBLEMA	IZDELAVA IDEJE	NAČRTOVANJE IGRE	RAZVOJ ZAHTEV	ARHITEKTURA	NAČRTOVANJE PROGRAMSKE OPREME	RAZVOJ	INTEGRACIJA	IZDAJA	UČENJE	POVRATNE INFORMACIJE
Bartle (2004)	PREDPRODUKCIJA	PRODUKCIJA	IZDAJA	PODPORA							
Fulleton (2014)	KONCEPT	PREDPRODUKCIJA	PRODUKCIJA	ZAGOTAVLJANJE KVALITETE	VZDRŽEVANJE						
Adams (2013)	FAZA KONCEPTA	FAZA ELABORACIJE	FAZA TUNINGA								
Kanode in Hadadd (2009)	PREDPRODUKCIJA	PRODUKCIJA	TESTIRANJE								

PRILOGA F: PRIMERJAVA TEMELJINIH KONCEPTOV METOD INŽENIRINGA PROGRAMSKE OPREME

SPEM	ISO24744	RUP	MetaME
	Stage	Discipline	Domain Discipline
			Concept
Work Product Definition, Work Product Use	Work Product (Model, SoftwareItem, HardwareItem, Document) ModelUnit	Artifact (Model, ModelElement, Document, SourceCode, Executable)	Artifact
Work Definition	WorkUnit	Work	Work
Activity	Process	Workflow Activity	Process Activity
Task Definition, Task Use	Task	Task	Task
Step	Action	ActivityStep	ActionStep
			Transformation
(Phase, Iteration) as Kind	TimeCycle, Phase, Build	Lifecycle, Phase, Iteration	Phase
Milestone	Milestone	Milestone	Milestone
Role Definition, Role Use	Role	Worker	Role
Tool Definition	Tool		Tool Utility
	Language, Notation		Notation
Guidance	Guildeline		Guidance
	Technique		Technique
	Constraint		Constraint

Vir: (Engels in Sauer, 2010, str. 418).

PRILOGA G: OSNUTEK DOKUMENTA OBLIKOVANJA IGRE PO BATES (2004)

Vir: (Bates, 2004, str. 276–91).

PRILOGA H: PRAKSE RAZLIČNIH AGILNIH PROCESNIH MODELOV

MODEL	PRAKSA	NAMEN
XP	40-urni teden	Teden je sestavljen iz 40 delovnih ur. Niti dva delovna 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, str. 24–68).

PRILOGA I: KRITERIJI ZA IZBIRO ORODIJ UPRAVLJANJA PROJEKTOV

	Criteria	Description
1	Task Scheduling	Task scheduling refers to the assignment of start and end times to a set of tasks. This feature lets software project manager track important project milestones and note who is responsible for each task.
2	Resource Management	This feature lets the software project manager organize and trace requirement details to ensure that proper resources are committed to the project. The software project manager can establish information relationships between multiple documents, assign attributes to the information, such as task assignment, priority and status, and change these over time to reflect changes in the project.
3	Collaboration	Collaboration enables both structured and free-flow sharing of knowledge and best practice. It includes project status reports that are accessible via a Web page, integrated e-mail or discussion boards.
4	Time Tracking	Time tracking allows recording, analyzing and reporting associated with project working routine. Software project manager can use time tracking feature to manage employee timesheets and expenses, calculate salaries, prepare project estimates, and get invoices based on personal or client work rates
5	Estimating	The estimate feature allows the project manager to generate, manage, and validate estimates of effort for a wide variety of projects. It evaluates the project plan, project requirements, information about working environment, and even different aspects of company's culture.
6	Risk Assessment	Risk assessment helps the software project manager in identifying and planning for potential project risks. It could also help software project manager to describe the various risk factors and how to score them.
7	Change Management	This feature lets software project manager control schedules, resources, and deliverables of project. It can manage the impact that changes have on project objectives, and it lets software project manager trace changes to see how each requirement's changes affect multiple other requirements.
8	Reporting/ Charts	In addition to Gantt or PERT charts, some tools provide hundreds of charts and reports. In addition some tools allow users to develop a custom report format that suits the organization.
9	File Attachment	File attachment let users customize project tasks, like file sharing and document management systems, and Web page authoring. Some tools help users control document version and checkout as well.
10	E-mail notification	Most software project management tools provide integrated e-mail notification to keep team members informed of the current status such as defects, change in documentation issues and requests, and other related issues.
11	Process/ Methodology	Process/Methodology features allow the software project manager to develop and implement a consistent and standardized process workflow.
12	Portfolio Management	Portfolio Management feature helps the software project manager manage multiple projects that are related, such as infrastructure technologies, desktop applications and so on, and allocate resources accordingly.

Vir: (Ahmad in Laplante, 2006).

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

Dimensions	Activity ID (AID)	Game Development Process Activities (GDPA)
Game Design Strategy	1	Game design document management (GDD)
	2	Team configuration and management (TCM)
	3	Requirements management and modelling (RMM)
	4	Game prototyping (GP)
	5	Risk management (Risk_Mgmt.)
Game Development Methodology	6	Quality architecture (QA)
	7	Assets management (AM)
	8	Game engine and development (GED)
Game Playability & Usability	9	Test management (TM)
	10	Maintenance support (Mt._S)
	11	Fun factor analysis (FFA)
	12	Ease of use (EU)
Business Performance	13	Market orientation (MO)
	14	Time to market (TTM)
	15	Relationship management (RM)
	16	Monetization strategy (MS)
	17	Innovation (I)
	18	Stakeholder collaboration (SC)

Vir: (Aleem in drugi, 2016a, str. 62).

PRILOGA K: OCENE PRAGA ZA DOSEGANJE 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 in drugi, 2016a, str. 68).

PRILOGA L: VPRAŠALNIK ZA OCENITEV OPTIMIZIRANE ZRELOSTI

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 in drugi, 2016a, str. 67–68).

PRILOGA M: 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