Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jan Grilanc

Računalniška igra za učenje metodologije Kanban

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: prof. dr. Viljan Mahnič

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kot ena izmed možnosti za učenje metodologije Kanban je v literaturi omenjena tudi uporaba izobraževalnih iger. Na Univerzi Aalto so v ta namen uporabili igro getKanban. Za igranje te igre je potreben komplet, ki vsebuje igralno ploščo, kartice z opisom delovnih nalog, dogodkovne kartice, več raznobarvnih kock, nalepke, flomastre in različne formularje za beleženje rezultatov.

V okviru vaše naloge izdelajte program, ki bo omogočal igranje igre getKanban na računalniku. V nalogi najprej na kratko predstavite glavne značilnosti metodologije Kanban in opišite pravila za igranje. Osrednji del naloge pa namenite opisu zasnove računalniškega programa in njegovih funkcionalnosti. V zaključku predstavite še izkušnje z njegovo uporabo.

Hvala staršem za nepresahljiv vir zaupanja, Suzanne za neizmerno potrpežljivost, sodelavcem za nesamoumevno prilagodljivost ter mentorju za iskreno priložnost in vso pomoč.

Moji dragi Suzanne.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
WIP	work in progress	tekoče delo
LTDC	lead time distribution chart	diagram porazdelitve potrebnega časa
CFD	cumulative flow diagram	kumulativni diagram delovnega toka

Povzetek

Naslov: Računalniška igra za učenje metodologije Kanban

Avtor: Jan Grilanc

Diplomska naloga obravnava učenje metodološkega pristopa Kanban, natančneje učenje skozi igro. Razložili bomo pojem igrifikacije (*gamification*) ter preverili, zakaj je slednja zanimiva z vidika učenja. Pri tem se bomo oprli na obstoječo namizno igro, getKanban. Le-to bomo opisali, si pogledali primer igranja, nato pa sledi snovanje računalniške inačice. Tudi računalniško igro bomo opisali, skozi potek igranja pa bomo izpostavili njene gradnike in povedali kako smo jih z orodjem Unity ustvarili. Na koncu nas čaka krajši vpogled v uspešnost ustvarjene igre, ki ga bomo primerjali z raziskavo efektivnosti namizne inačice. Za zaključek nam ostane še opis morebitnih nadaljnjih izboljšav.

Ključne besede: Kanban, getKanban, učenje, igra, igrifikacija, Unity.

Abstract

Title: Computer game for teaching the Kanban methodology

Author: Jan Grilanc

This Bachelor's degree deals with teaching Kanban methodology, specifically through a game. We will explain the concept of gamification and look into why it is interesting within the teaching framework. For that purpose we will lean onto an existing board game, getKanban. Board game will be described along with its proceedings and mechanics, after which we will turn to creating its computer counterpart. Computer game will also be described. Through its gameplay we will present its many elements and explain how we created them using Unity. At the end we will analyze game's effectiveness which will be compared to a study of board game's effectiveness. Finally, the writing will be concluded with description of potential future improvements.

Keywords: Kanban, getKanban, learning, game, gamification, Unity.

Poglavje 1

Uvod

Med metodologijami, ki se zadnja leta uporabljajo v razvoju programske opreme, je vedno bolj razširjen Kanban - metodologija, ki združuje agilne principe z vitkimi. Letno poročilo *VersionOne* [8] tako ugotavlja, da je med anketiranci uporaba Kanbana v letu 2017 narasla iz 50% na kar 65%.

Glede na porast zanimanja za Kanban, lahko pričakujemo od delodajalcev, da bodo zahtevali poznavanje slednjega od morebitnih iskalcev zaposlitve v sektorju razvoja programske opreme. Kanban, izbrano metodologijo, je torej potrebno pravilno razumeti. Le na ta način lahko zagotovimo pravilnost uporabe in s tem povezanih koristi, obenem pa se izognemo morebitnim škodljivim aspektom, ki jih napačna uporaba lahko povzroči. V ta namen je torej pomembno izobraževanje bodočih inženirjev - skozi temu prirejen študijski program na univerzah ali pa v okviru specializiranih delavnic.

Nedavna študija na temo Kanbana v izobraževanju [5] je pokazala, da je najpogostejši način učenja skozi praktično delo na projektih, pomembno mesto pa je zasedlo tudi učenje skozi izobraževalne igre, ki pogosto simulirajo realne situacije. Poučne igre so vedno bolj razširjene, zanimivo pa je, da jih večina obravnava SCRUM [9, 4], medtem ko Kanban in ostale, podobne metodologije niso zastopane tako korenito.

Pozitivni vplivi igralnega pristopa so bili že večkrat zabeleženi [6], specifično na področju učenja Kanbana pa obstaja tudi članek [7] izpod peresa

treh sodelavcev (Villie T. Heikkilä et al), ki opisuje uporabo igre getKanban. Gre za namizno igro, plod dela Russell Healyja, ki pa ne pozna računalniške inačice. Cilj te naloge je izdelati računalniško igro, ki svojo strukturo črpa iz getKanban zasnove.

V ta namen bomo v drugem poglavju najprej povedali nekaj o metodologiji Kanban in nato opisali igro getKanban, s poudarkom na poteku igranja.

V tretjem poglavju bomo pogledali, kakšne so zahteve za prenos v digitalno obliko in kako nam je pri realizaciji pomagalo razvojno orodje Unity. V istem poglavju bomo pod drobnogled vzeli posamezne gradnike digitalne igre in razložili, kako funkcionirajo. Četrto poglavje bo predstavilo potek igre skozi primer simulacije enega dne, kjer bomo opisane gradnike prikazali v praktični luči.

V petem poglavju nas čakajo uporabniški vtisi in želje, ki jih bomo primerjali z vtisi študentov iz zgoraj omenjene študije [7]. V zaključku se bomo ozrli na prehojeno pot, povzeli najpomembnejše rezultate in začrtali možnosti za nadaljevanje dela.

Poglavje 2

Metodologija

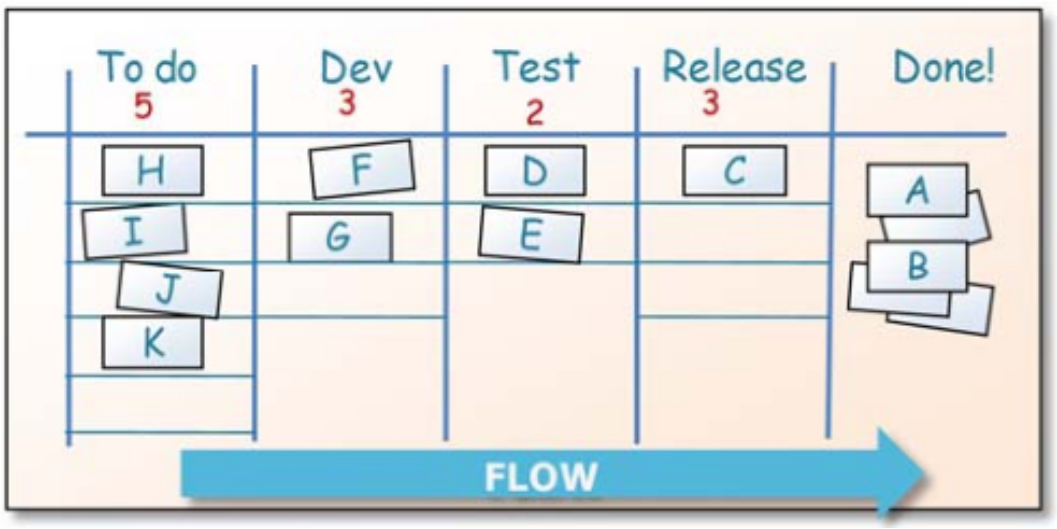
2.1 Kanban

Ko govorimo o Kanbanu, govorimo o načinu obvladovanja dela. Svoje korenine ima v štiridesetih letih prejšnjega stoletja, kjer so njegove zametke vzpostavili v Toyotinih tovarnah. Zavoljo konkurenčnosti se je Toyota takrat usmerila v drugačno obliko ponudbe in začela proizvajati avtomobile le glede na obstoječa naročila. S takim pristopom je prišlo drugačno obvladovanje produkcije.

Kanban se že dlje časa iz avtomobilske industrije uspešno seli na druga področja, posebej pa izstopa v programerskih vodah. Danes je še vedno presenetljivo podoben njegovim začetkom. V osnovi ga lahko opišemo s tremi ključnimi točkami [3]:

- vizualizacija delovnega toka,
- omejevanje tekočega dela (*work in progress limit*, v nadaljevanju *omejitev WIP*),
- spremljanje potrebnega časa (*lead time*).

Razložimo zgornje pojme. Delovni tok najlažje vizualiziramo tako, da delo, ki ga je treba opraviti, razdelimo na manjše, zaključene enote, ki jih zapišemo na



Slika 2.1: Kanban tabla povzeta iz knjige Kanban and Scrum [3].

fizične kartice. Kartice nato obesimo na tablo. Da bi poleg dela prikazali še njegov potek (delovni tok), si pri tem pomagamo s poimenskimi stolpci. Vsak stolpec na tabli ponazarja nek korak v razvoju (toku), preko položaja kartic v stolpcih pa hitro vidimo, kako daleč v razvojnem procesu je posamezno opravilo (kartica).

Primer table s karticami lahko vidimo na sliki 2.1, kjer je delovni tok razdeljen na pet stolpcev. V prvem so še nezačete kartice, v drugem kartice, ki so v razvoju, v tretjem pa tiste, ki so razvite, a se jih še testira za morebitne napake. V četrtem stolpcu govorimo o karticah, ki čakajo na izdajo (*release*), zadnji stolpec pa vsebuje kartice, ki so zaključene in so torej na koncu delovnega toka. Na tem mestu poudarimo, da sam Kanban ne predpisuje oblike table ali števila stolpcev, ampak lahko vsaka razvojna skupina prilagodi strukturo table svojemu razvojnemu procesu.

Razvojna skupina prav tako oceni, koliko kartic je lahko naenkrat v delu, običajno pa ima kar vsak stolpec svojo omejitev tekočega dela, ki jo dosežemo z nastavljanjem omejitve WIP. Na ta način Kanban poskuša poskrbeti, da delo v posameznem koraku nikoli ne pade v eno od skrajnosti: preveč kartic, kot smo jih zmožni naenkrat rokovati (nekatere kartice stojijo, posledično

imamo slab potrebni čas), oz. premalo kartic (nekateri člani razvojne ekipe nimajo dela).

Z omejevanjem WIP poskušamo optimizirati potek dela, velja pa omeniti, da Kanban ne predpisuje nobenih specifičnih vrednosti ali napotkov za določanje takih omejitev. To je smiselno, saj Kanban uporabljajo različno velike ekipe, z različnim številom posameznih specialistov. Namesto konstant nam Kanban ponuja ogrodje za prilagoditev poteka dela, obenem pa dopušča sprotno spreminjanje omejitev WIP glede na naše potrebe.

Vse kartice začnejo pot na levi strani, v prvem stolpcu, nato pa se postopno premikajo proti desni, kot to definira delovni tok. To počnejo s pomočjo mehanizma *pull*, namenjenega nadzoru delovnega toka.

Recimo, da smo postavljeni v vlogo testerja. Vse kartice, na katerih trenutno delamo, so vedno v istem stolpcu; v primeru slike 2.1 gre za stolpec *Test* ter kartici *D* in *E*. Mehanizem *pull* nam omogoča, da katerokoli izmed kartic, ki so zaključile delo v predhodni fazi – razvoj v stolpcu *Dev* –, pomaknemo proti desni in začnemo delo na njej – testiranje v *Test* stolpcu. Takemu pomikanju kartice od leve proti desni rečemo tudi potovanje po plavalni stezi (*swimming lane*), paziti pa je treba na omejitev WIP!

Ostane nam še tretji princip, spremljanje potrebnega časa. Ideja tu je računanje časa, ki ga posamezna kartica preživi v delovnem toku. Pri tem poskušamo slednjega zmanjšati in obenem zagotoviti optimalen razvoj. Naš cilj je najti zdravo ravnotežje med omejitvijo WIP in izkoriščenostjo virov. Običajno si pomagamo s temu namenjenim diagramom porazdelitve potrebnega časa – več o diagramih v kasnejšem poglavju.

Iz zgoraj napisanega je razvidno, da je cilj Kanbana predvsem izboljšanje delovnega procesa. Zaradi večje prilagodljivosti Kanban ne predpisuje bolj specifičnih navodil ali omejitev, ki bi delo ekipe lahko nehote upočasnili, namesto da ga spodbujajo. Poleg tega Kanban ne prepoveduje dodatnih pravil, ki bi jih razvojna ekipa želela vključiti v delovno okolje! Tako imamo npr. proste roke, da definiramo vloge posameznikov v ekipi in z njimi povezane obveznosti, podobno kot to počne metodologija Scrum, s to razliko, da jih

slednji zelo natančno predpiše in pričakuje doslednost pri njihovi izvedbi.

Prav njegova ohlapnost je glavni razlog, da je učenje Kanbana relativno težko. V uvodu omenjena študija [5] je ugotovila, da imamo v znanstveni literaturi zelo malo virov, ki obravnavajo učenje Kanbana v univerzitetnem okolju.

Eden izmed načinov poučevanja Kanbana je tudi učenje skozi igro. Princip ni nov, zaradi pozitivnega vpliva na osredotočenost pa je (vsaj kratkoročno) učinkovit, kot je to ugotovila študija iz leta 2014 [2].

Poleg nagrajevanja igralca in ostalih psiholoških tehnik, nam igre lahko ponudijo simulacijo realnih življenjskih situacij – v nekaterih primerih je tak pristop celo bolj primeren od dejanskih izkušenj iz vidika porabljenega časa, saj omenjene situacije v igrah lahko pohitrimo in tako zaobjamemo daljša obdobja simuliranega dela v zelo kratkem intervalu, npr. v poteku ene igrine seje. S tem lahko potencialno pokrijemo mesece realnega dela. Seveda pa moramo upoštevati, da igra ne more povsem nadomestiti izkušenj, pridobljenih skozi pravo delo.

2.2 Opis igre getKanban

V tem poglavju si bomo pogledali potek simuliranega dneva v igri getKanban ter opisali posamezne igralne elemente, ko bo predvidena njihova uporaba. GetKanban je namenjen več igralcem, ki tvorijo ekipo, zadolžene za razvoj spletne aplikacije z naročniškim poslovnim modelom – več naročnikov pomeni več naročnin, cilj igre pa je maksimirati zaslužek. To ekipa doseže z razvojem novih funkcionalnosti, ki pripeljejo nove uporabnike.

2.2.1 Elementi, vloge in začetna postavitev

Za začetek naštejmo vse potrebščine, ki jih potrebujemo za igranje:

- Igralna plošča
- Kartice

- Kocke v treh barvah (rdeča, modra, zelena)
- Barvni markerji
- Prazni diagrami za beleženje izbranih podatkov

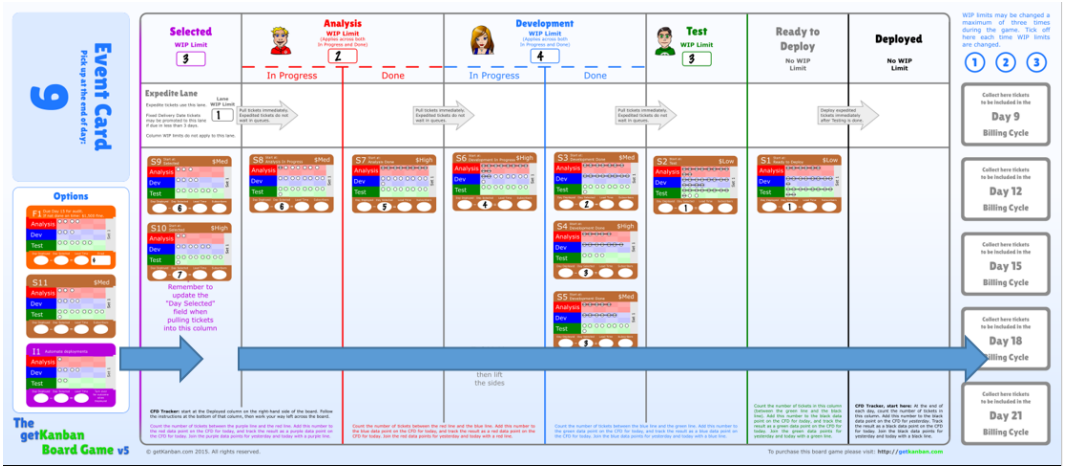
Poleg zgoraj naštetih pripomočkov igra definira naslednje vloge:

- Projektni vodja – njegova naloga je koordinacija ekipe s poudarkom na čim hitrejšem zaključku igre.
- Sledilec kumulativnega diagrama toka – najtežja vloga, glede na igrina navodila, obsega pa risanje istoimenskega diagrama, ki bo razložen kasneje.
- Sledilec porazdelitve delovnega časa in sekvenčnega diagrama dela – podobno kot zgornja vloga, več o grafih samih pa v nadaljevanju.
- Finančni analitik – snovalec finančnega poročila.
- Ostali igralci (analitiki, razvijalci, testerji) – njihov doprinos je v veliki meri predstavljen s kockami, katerih uporabo določajo našteti specialisti.

Pred pričetkom igre je treba pripraviti igralno ploščo. Na temu namenjeno mesto na igralni plošči najprej postavimo dogodkovne kartice, zraven njih pa kupček še neuporabljenih kartic.

Navodila predvidevajo tridnevni finančni cikel, na koncu katerega določimo in zabeležimo rezultate zaključenih kartic. Da je začetek bolj dinamičen, začnemo z igranjem na deveti dan, ki je zadnji dan takega cikla. S takim pristopom se izognemo počasnemu začetku, spoznamo vpliv zaključenih kartic in dosežemo bolj zanimivo dinamiko igranja že prvi dan. Stanje po osmih dneh simuliramo s predpisano začetno postavitvijo prvih desetih kartic.

Kot je to razvidno iz slike 2.2, plošča definira naše stolpce. Ločimo stolpec *Selected* (vsebuje novo izbrane kartice, na katerih še nismo pričeli z delom),



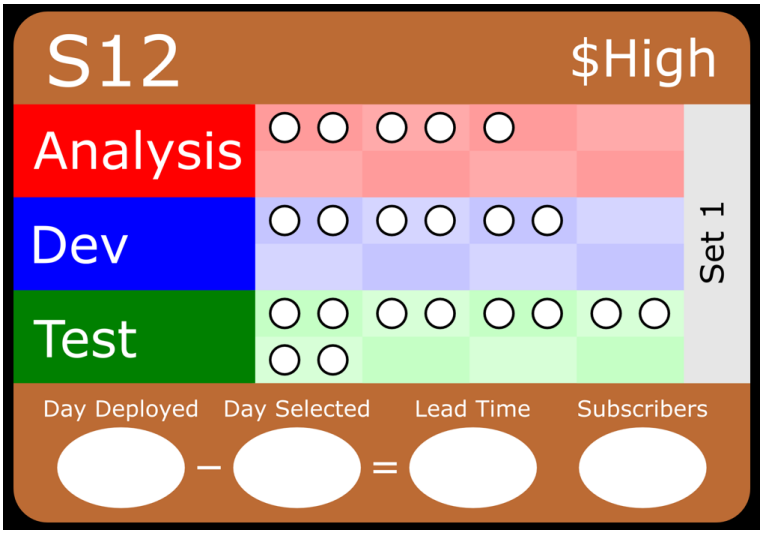
Slika 2.2: GetKanban igralna plošča, kot je prikazana v navodilih [1].

dva stolpca *Analysis* (analiza in zaključena analiza), dva stolpca *Development* (razvoj in zaključen razvoj) in stolpec *Test* (testiranje razvitih funkcionalnosti). Sledita še stolpec *Ready to Deploy* (zaključene kartice, čakajoče na postavitev) in stolpec *Deploy* (postavitve ob koncu finančnega cikla). Posebno omembo potrebuje pospeševalna plavalna steza (*expedite lane*) na vrhu stolpcev, ki služi posebnim primerom kartic.

2.2.2 Kartice: osnovne lastnosti in tipi

Primer kartice vidimo na sliki 2.3. Vse kartice imajo nekaj osnovnih lastnosti, kot sta naziv in tržna vrednost (nizka, srednja, visoka). Posamezna kartica ima tudi določeno število pik, ki nakazujejo potrebni obseg dela – vloženo delo bomo tekom igre simulirali z metanjem kock. Obseg dela je razdeljen na tri skupine: analiza, razvoj in testiranje. Predpisani obseg se upošteva pri potovanju kartice po plavalni stezi. Dokler npr. obstaja v analizi še nekaj nevloženega dela, kartica ne sme v razvoj.

Kartice imajo tudi prostor za zapis zaporedne številke dneva, ko so bile izbrane (z umestitvijo na ploščo) ter prostor za številko dneva postavitve (*deploy*). Zadnja dva prostora sta namenjena izračunu potrebnega časa in pridobljenih naročnikov. Izračun je odvisen od tipa kartice, nakazanega z



Slika 2.3: Osnovna kartica, kot je prikazana v navodilih [1].

barvo ozadja.

Tipi se delijo na štiri skupine: standardne zahteve, nujne zahteve, zahteve z določenim datumom in zahteve z neotipljivo vrednostjo.

Standardne zahteve so osnovne kartice rjave barve. Njihova izpolnitev pripelje nove naročnike.

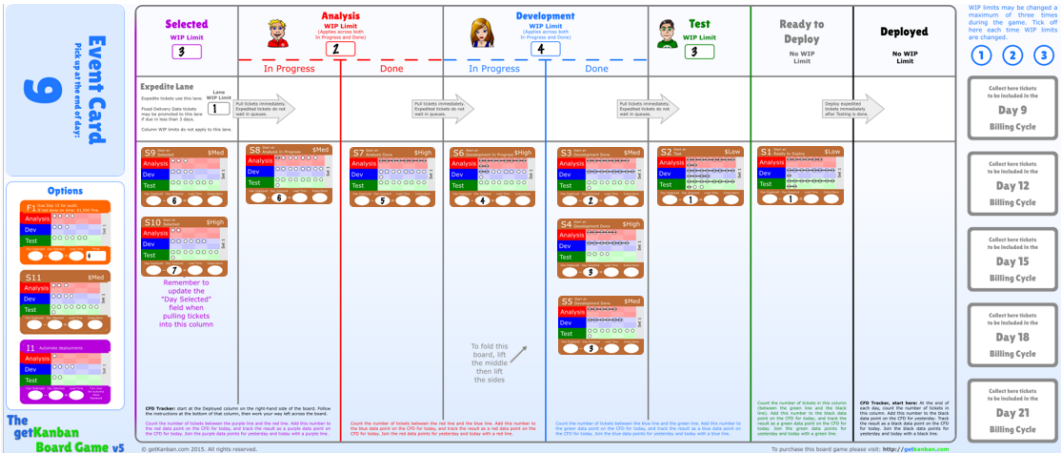
Nujne zahteve so predstavljene z belimi karticami, ki uporabljajo pospeševalno plavalno stezo. Predstavljajo pomembno delo z visokimi nagradami.

Zahteve z določenim datumom so oranžne barve. Imajo določen datum, do katerega jih je potrebno zaključiti. S tem se običajno izognemo denarni kazni, oz. zaslužimo nagrado.

Zahteve z neotipljivo vrednostjo predstavljajo kartice vijolične barve, ki ob uspešnem zaključku ne definirajo takojšnjega, otipljivega rezultata.

2.2.3 Pričetek igranja s pripisovanjem kock

Igra začne vsak dan s sestankom (*standup meeting*, v nadaljevanju *standup*). Na njem lahko določimo nove omejitve WIP (neobvezen korak). Nato moramo obvezno izbrati nove kartice – dovolj, da povsem napolnimo prvi, izbirni



Slika 2.4: GetKanban igralna plošča, začetna postavitev [1].

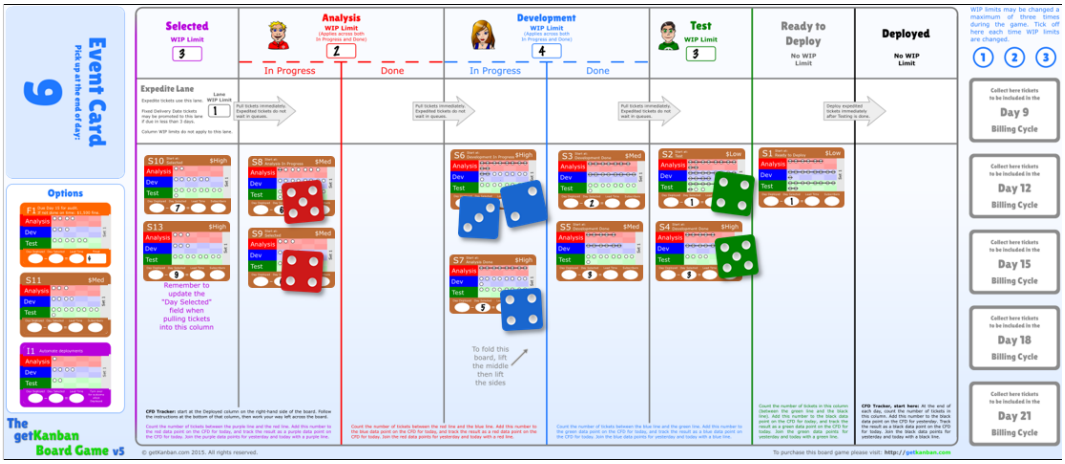
stolpec! V primeru začetne postavitve na sliki 2.4 predvidevamo nespremenjeno omejitev WIP. Izberemo torej le eno kartico in izbirni stolpec je poln, na voljo pa so nam vse kartice v kupčku. Na izbrani kartici označimo dan izbire – 9.

Sledi dodeljevanje kock posameznim karticam. Ta mehanika služi simulaciji vloženega dela. Količina pik, dobljenih z metom kock, se odšteje od preostalega potrebnega dela na kartici, ki so ji bile pripisane kocke.

Pri tem velja nekaj pravil. Kocke se delijo na tri barve, vsaka barva pa se nanaša na eno izmed faz razvojnega procesa: rdeča je namenjena analizi, modra razvoju, zelena pa testiranju. Na voljo imamo dve rdeči, tri mode in dve zeleni kocki. Na naslednji korak lahko nadaljujemo šele, ko smo karticam pripisali vse kocke.

Lahko bi rekli, da posamezna kocka simulira delo enega člana razvojne ekipe. Kocke lahko uporabimo tudi izven njihove barvne skupine, a se takrat njihov met šteje le polovično (z zaokroževanjem na najbližje zgornje celo število), s čimer prikažemo prehajanje sodelavcev na področje izven poznanih okvirov njihove specializacije in posledično manj opravljenega dela.

Navodila, ki nas vodijo skozi deveti dan, pripišejo kocke karticam, kot je to prikazano na sliki 2.5. Vidimo, da smo v testni stolpec premaknili kartico, ki je predhodno zaključila razvoj. Obema karticama v testiranju smo nato



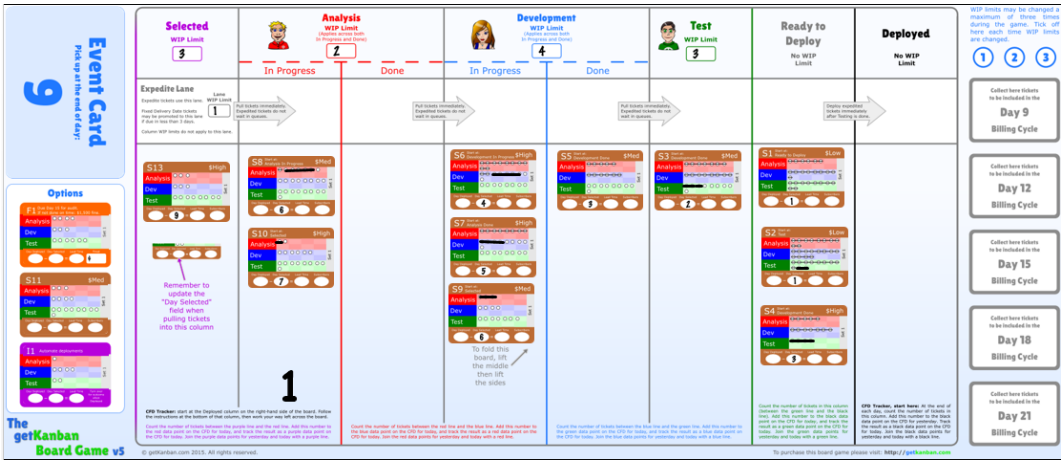
Slika 2.5: Igralna plošča z dodeljenimi kockami [1].

pripisali po eno zeleno kocko. S premikom kartice v testiranje smo sprostili razvoj, v katerega smo posledično lahko sprejeli edino kartico z zaključeno analizo. V razvoju smo imeli v tem trenutku dve kartici. Prvi smo dodelili dve, drugi pa eno modro kocko. Analiza ima omejitev WIP nastavljeno na dve kartici. Ker smo jo ravnokar sprostili, je lahko sprejela novo kartico iz izbirnega stolpca. Obema karticama v analizi smo nato pripisali po eno rdečo kocko. S tem smo uporabili vse kocke, kot to igra od nas zahteva.

2.2.4 Simulacija vloženega dela

Prišli smo do koraka, ko dodeljene kocke vržemo. Z metom simuliramo vloženo delo. Skupno število pik vseh kock, dodeljenih posamezni kartici nato odštejemo od potrebnega dela. Pazimo, da upoštevamo vloženo delo, odvisno od tega v katerem stolpcu se kartica nahaja. V primeru večjega števila pik (presežek oz. *overflow*), kot je potrebno za dokončanje dela na neki kartici, lahko preostanek prenesemo na katerokoli kartico v istem stolpcu, kot se nahaja prvotno izbrana kartica. Presežek lahko upoštevamo le pod pogojem, da izvira iz kartice, ki ni imela dodeljenih več kot dveh kock.

Kartice, ki so izpolnile vloženo delo, premaknemo v naslednji stolpec. Tam bodo počakale vsaj do naslednjega dne. Igra predvideva mete, ki služijo



Slika 2.6: Igralna plošča po metu kock [1].

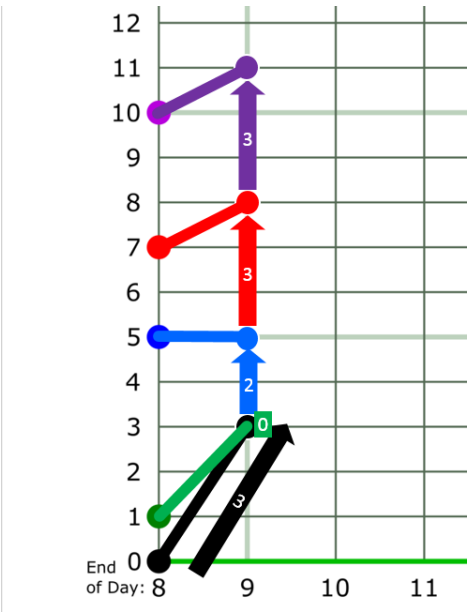
razpletu prikazanem na sliki 2.6. Na eni izmed testnih kartic smo imeli manj potrebnega dela, kot je bil vreden met kocke. Ker testni stolpec še ni izpolnil omejitve WIP, smo iz razvoja lahko vzeli eno zaključeno kartico in ji pripisali presežek.

Podobno situacijo smo imeli z rdečimi kockami v analizi: eden izmed metov je imel presežek, zato smo dotično kartico prestavili v razvoj (in pri tem spet upoštevali omejitev WIP), neizkoriščene pike pa pripisali kartici iz izbirnega stolpca, a le po premiku v analizo!

2.2.5 Zaključek finančnega cikla, diagrami

Ker smo na koncu tridnevnega finančnega cikla, sledi poseben korak: postavitev kartic z izračunom finančnega stanja. Iz stolpca zaključenih kartic prestavimo slednje v postavitveni stolpec. Na vsaki izmed premaknjenih kartic označimo zaporedno številko dneva postavitve, v tem primeru 9. Izračunamo potrebni čas, tako da postavitvenemu dnevu odštejemo izbirni dan. Na hrbtu kartice nato na podlagi rezultata preverimo, koliko naročnikov smo pridobili: manjši potrebni čas pomeni večje število naročnikov; kartica, ki preživi predolgo na igralni plošči, lahko vodi v izgubo uporabnikov.

Na koncu vsakega dneva posodobimo kumulativni diagram toka (*cumu-*

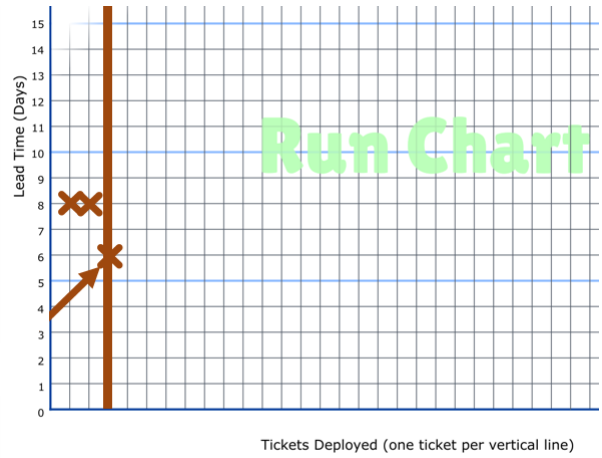


Slika 2.7: Kumulativni diagram toka na koncu devetega dne [1].

lative flow diagram, v nadaljevanju tudi *CFD*). Namen tega diagrama je beleženje dnevnega stanja na posameznih delih plošče. To počnemo z ločevanjem števila kartic v pet, barvno ločenih skupin – stolpec postavitev (črna), stolpec zaključenega testiranja (zelena), stolpca zaključenega razvoja in testiranja v teku (modra), stolpca zaključene analize in razvoja v teku (rdeča), ter izbirni stolpec s stolpcem analize v teku (vijolična).

Igra nam poda zabeleženo stanje na koncu osmega dne: 0 postavitev, 1 kartica z zaključenim testiranjem, 4 kartice z zaključenim razvojem oz. v testiranju, 2 kartic z zaključeno analizo oz. v razvoju in 3 kartice v izbirnem stolpcu oz. v analizi. Risanje diagrama nadaljujemo od tu naprej. Os x predstavlja zaporedno številko dneva, os y število kartic, risanje pa sledi določenim navodilom. Poglejmo si jih na primeru. Ker je opis lahko nekoliko abstrakten, imamo za referenco pripravljen odsek diagrama po že opravljenem delu – slika 2.7.

V postavitveni stolpec smo torej prestavili tri kartice. Postavimo se na deveti dan (os x) in štejemo tri mesta navzgor od števila postavitev



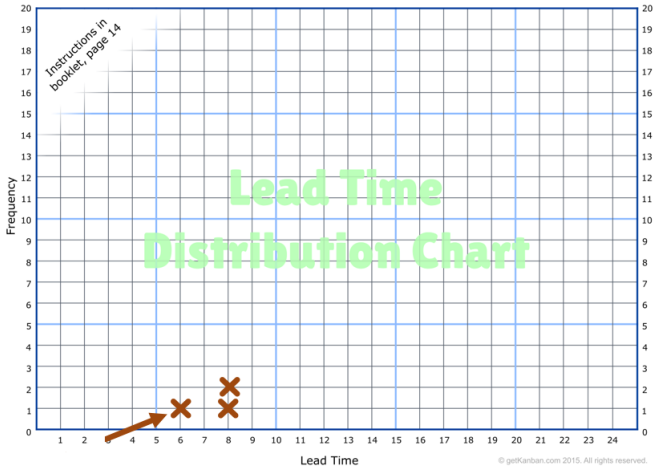
Slika 2.8: Diagram potrebnega časa [1].

prejšnjega dne (0). Na tem mestu naredimo črno točko in jo s črnim markerjem povežemo s črno točko prejšnjega dne. Postavitve so edina vrednost, ki jo štejemo od rezultata predhodnega dne navzgor, vse ostale količine kartic pa nato dodajamo eno nad drugo. Ker v stolpcu zaključenega testiranja nimamo nobene kartice, se to prevede v točko na istem mestu, kot je pravkar označeno število postavitev. To točko z zelenim markerjem povežemo s točko, ki predstavlja število kartic z zaključenim testiranjem prejšnjega dne.

Enako ponovimo za vse nadaljnje točke - nad zeleno točko dve mesti višje narišemo modro (imamo eno kartico v testiranju in eno v stolpcu zaključenega razvoja), tri mesta nad modro še rdečo (tri kartice v razvoju), za zaključek pa še vijolično točko tri mesta višje (dve kartic v analizi in ena v izbirnem stolpcu). Vsako izmed naštetih točk prav tako povežemo z barvnim parom iz prejšnjega dne. Dobimo stanje slike 2.7.

Ostaneta nam še dva diagrama, ki ju dopolnimo le vsake tri dni: sekvenčni diagram dela (*run chart*) in diagram porazdelitve delovnega časa (*lead time distribution chart*, v nadaljevanju tudi *LTDC*). Oba sta relativno preprosta.

Sekvenčni diagram dela predstavlja os y, ki označuje potrebni čas, na osi x pa vodimo postavitve kartic. Tako na koncu devetega dne za tri postavljene kartice z zaporednimi potrebnimi časi 8, 8 in 6 dni dobimo sliko 2.8.



Slika 2.9: Porazdelitev delovnega časa [1].

Malenkost več dela zahteva diagram porazdelitve delovnega časa. Tu na osi x vodimo potrebni čas, na osi y pa frekvenco pojavitev slednjega. Frekvenca narašča z vsako kartico, ki ponovi potrebni čas njenih predhodnikov, izračunan ob postavitvi kartice; vsaka ponovitev predstavlja novo, za eno mesto višje postavljeno točko. Primer stanja na koncu devetega dne lahko vidimo na sliki 2.9.

Oba diagrama uporabljata barvne markerje za označevanje kartic. Barva je odvisna od tipa kartice, uporabljamo pa rjave, bele, oranžne in vijolične markerje.

2.2.6 Zaključek dneva

Po posodobljenih diagramih vse kartice umaknemo iz postavitvenega stolpca na posebno mesto ob desnem robu igralne plošče. Teh mest je več in se ponovijo za vsak finančni cikel (deveti dan, dvanajsti dan itd.). Na ta način ločimo postavljene kartice po tridnevnih fazah za morebitni podrobni vpogled v potek igre, saj nam diagrami ne podajo informacij o specifičnih karticah, kot so npr. njihova vrednost ali potrebno delo.

Na vrsti je finančno poročilo. Preštejemo nove naročnike, jih dodamo k

obstoječim, ter izračunamo skupno vrednost naročnin. Nato odštejemo globe in prištejemo denarne nagrade, v kolikor so se le-te pojavile med igranjem.

Na koncu povlečemo še dogodkovno kartico z vrha kupčka. Preberemo in upoštevamo njene posledice. S tem smo zaključili deveti dan. Naslednji dan ponovimo vse korake, razen tistih, ki se tičejo finančnega cikla.

Poglavje 3

Zasnova in tehnične rešitve

3.1 Analiza in načrt

V okviru analize je bilo potrebno ugotoviti, kakšne so naše zahteve in kaj mora digitalna inačica igre pravzaprav podpirati. Obenem so postale očitne tudi nekatere razlike, ki jih s seboj pripelje prenos fizičnega v svet ničel in enic. Teoretični del mora kasneje služiti praktični realizaciji.

Predstaviti moramo vizualne elemente igre. Potrebovali bomo ploščo, kartice in kocke. V primeru plošče lahko prezentacijo nekoliko strnemo, saj je namizna oblika precej široka, mi pa imamo opravka z omejeno površino (računalniški zaslon). Tu pride v poštev npr. manjše število stolpcev, poleg tega pa ne potrebujemo posebnih mest za odlaganje postavljenih kartic in za oba kupčka neuporabljenih ter dogodkovnih kartic. Namesto tega lahko po potrebi spreminjamo vidno podlago na zaslonu. Primer takega spreminjanja videnega je lahko izbira novih kartic, dosegljiva preko klika na temu namenjen gumb. Ob kliku na gumb se nariše posebno okno za pregled vseh neuporabljenih kartic, ki pa (lahko tudi v celoti) prekrije igralno ploščo. Po opravljeni izbiri se izbirno okno skrije, igralec pa se vrne nazaj k igri in igralni plošči.

Kartice morajo še vedno jasno predstavljati potrebne informacije o potrebnem in vloženem delu, vrednosti, tipu ter nazivu. Tekom igranja nam

morajo omogočati njihovo premikanje, izbiro novih kartic, računanje vrednosti postavitev, ter jasno ločevanje tipov z vsemi pritičnimi lastnostmi.

Diagrame hočemo risati sproti, upoštevati pa moramo kdaj igra predvideva dodajanje novih podatkov. Ker je čas razvoja žal omejen, se bomo omejili na dva izmed treh predstavljenih diagramov – kompleksnejši kumulativni diagram toka in preprostejši diagram porazdelitve dela. Glede na predstavljeno getKanban osnovo imamo določene zahteve.

Diagrami se morajo dinamično prilagajati zaslonu, sorazmerno s količino dosegljivih podatkov. V namizni inačici nam velikost papirja ne predstavlja ovir, v digitalni obliki z omejeno površino pa ne želimo gledati ogromnega diagrama, ko imamo le za en dan vnosnih podatkov. Podpirati moramo tudi barvno predstavitev točk in premic, ki jo predvideva igra, kot je to opisano v podpoglavju o diagramih.

Igra opisuje vloge, ki smo si jih že pogledali, v digitalni inačici pa ne potrebujemo prav vseh. Diagrami bodo risani sproti, avtomatsko, torej s tem opravilom igralcev ne bomo obremenjevali. Tudi finančni analitik ni potreben, saj bo igra sama računala trenutno stanje in priigran zaslužek.

Projektni vodja, analitiki, razvijalci in testerji so še vedno zahtevani, saj bo ekipo potrebno koordinirati in se vsak dan dogovoriti, v katere kartice bomo vložili sprotno delo. Kot možna dodatna vloga, ki je getKanban ne predvideva, se pojavlja urejevalec kartic – nekdo, ki bi vnaprej pripravil igralne kartice, vnesel njihove vrednosti, tipe itd. S takšnim spreminjanjem kartic bo igra dolgoročno bolj zanimiva za morebitno ponovno igranje. Vlogo kreatorja kartic bi praviloma zasedel učitelj oz. inštruktor.

Podatkovnega modela, kot takega ne potrebujemo, saj imamo opravka s precej majhno količino nespremenljivih in še manjšim naborom spremenljivih podatkov. Namesto podatkovnih baz se lahko poslužimo preprostega branja iz datotek – več o tem v naslednjem poglavju o realizaciji.

3.2 Razvojno orodje Unity

Zaradi narave subjekta – računalniška igra – je bil razvoj digitalne igre v celoti izveden z orodjem Unity. Gre za ogrodje, specializirano za gradnjo 2D in 3D iger, podpira pa razvojna okolja Windows, MacOS in Linux (*cross-platform game engine*).

3.2.1 Zakaj Unity?

Najpomembnejši razlog tiči v lahki, bolj ali manj intuitivni uporabi. Vse vizualne elemente igre, od plošče, prek kartic, do povratne informacije uporabniku je mogoče ustvariti preko prijaznega grafičnega vmesnika. Med drugim Unity precej olajša kreacijo številnih, a podobnih objektov, kot so v našem primeru kartice. Namesto n posameznih objektov, narišemo enega, ki služi kot osnovna predloga (*template*) za vse ostale. Predlogo nato uporabimo v kodi, kjer za posamezno inačico nato lahko nastavimo in spreminjamo posamezne lastnosti: bravo, velikost, besedilo...

Tako ustvarjene elemente lahko razporejamo po igralni površini kar s pristopom „*primi in potegni*“. Njihova manipulacija je torej možna preko vmesnika orodja Unity, poleg tega pa vsakemu objektu lahko pripišemo eno ali več različnih skript – v jeziku C# napisanih datotek, ki npr. definirajo obnašanje objekta – in s tem dosežemo željeno obnašanje igralnih elementov. Več o tem v naslednji sekciji.

Še enega izmed pomembnejših razlogov najdemo v dobri dokumentaciji in široki bazi uporabnikov. Prva se je skozi leta obstoja orodja širila in dopolnjevala, kar je privedlo do precej dobrega teoretičnega zaledja. Po drugi strani imamo na voljo izkušnje številnih uporabnikov, ki so v preteklih letih ustvarili ogromno tematskih delavnic in odgovorili na prenekatero začetniško vprašanje.

Zgoraj napisano, v povezavi z dejstvom, da je orodje prirejeno tudi za 2D igre (kar se odlično preslika na obstoječo namizno igro), nas je pripeljalo do lahke odločitve o uporabi izbranega orodja.

3.2.2 O ogrodju

Ogrodje je spisano v jezikih C++ in C#, prav tako pa podpira pisanje kode v sintaksi C#. Pred letom 2017 je Unity omogočalo uporabo prirejene oblike JavaScript (*UnityScript*), do Unity 5 pa je orodje ponujalo alternativo jeziku C# – programski jezik *Boo*.

Luč dneva je orodje ugledalo poleti 2005, ko so ga uspešno splavili sodelavci David Helgason, Joachim Ante in Nicholas Francis iz Nizozemske. Njihov cilj je bila kreacija cenovno dostopnega orodja za razvoj iger, ki bi ga lahko uporabljali vsi nadobudni amaterski programerji. Začetna podpora je bila omejena izključno na MacOS. Za svoje dosežke je ekipa za orodjem dobila številne nagrade.

Licenciranje je razdeljeno na štiri kategorije. Za potrebe diplomskega dela je uporabljena osebna (*Personal*) opcija, ki je edina povsem zastonj. Ostale nadgradnje so odvisne od zaslužka, ki ga v Unity zgrajene igre prinesejo, po vrsti pa si sledijo: Plus, Pro in Enterprise.

3.3 Opis zasnovanih konceptov

Čas je, da pogledamo, kako smo zgoraj bolj pomembne elemente igre zasnovali v računalniški različici, v poglavju, ki sledi temu, pa bomo na primeru pokazali njihovo uporabo.

3.3.1 Igralna plošča

Igralna plošča je sestavljena iz grafičnih elementov, ki jih nudi orodje Unity. Gre za gradnike, ki nekoliko ironično prav tako slišijo na ime plošča (*panel*). Ker bo potrebno ločevati igralno ploščo od plošč-gradnikov, bomo prvo ploščo v tem podpoglavju vedno omenjali kot *igralno površino*, ostale pa le kot *plošče*. Za referenco imamo na voljo sliko 3.1.

Največja plošča služi kot ozadje igralne površine. Njen namen je poskrbeti, da imamo zaslon v celoti prekrit z igro. Naslednji največji gradnik je

Slika 3.1: Igralna plošča računalniške igre z ozadjem.

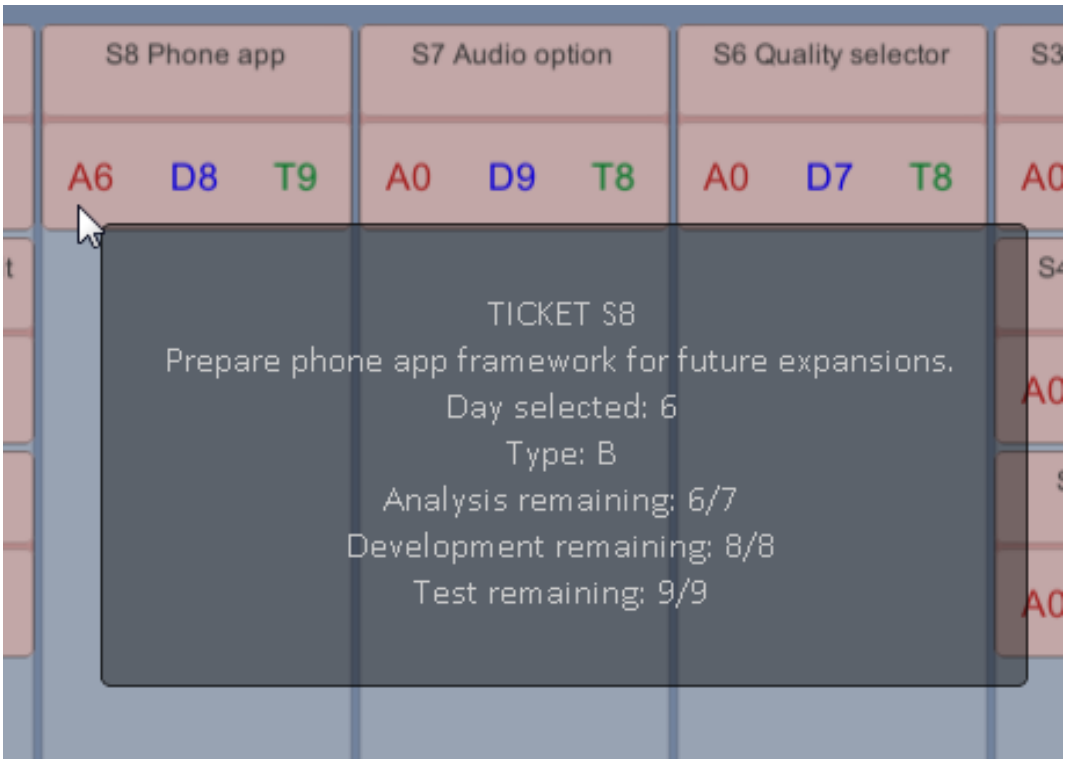
plošča, ki vsebuje igralno površino. Igralna površina je sestavljena iz mnogih komponent (gumbi, modalna okna, prikazna polja ipd.), vse pa niso vedno vidne, ampak se aktivirajo šele, ko igra predvideva njihovo uporabo.

Na igralni površini najdemo razporejene enako velike plošče, ki tvorijo že poznane stolpce: izbor (*Selected*), analiza (*Analysis – In Progress*), zaključena analiza (*Analysis – Done*), razvoj (*Dev – In Progress*), zaključen razvoj (*Dev – Done*), testiranje (*Test - In Progress*) in stolpec kartic, ki čakajo na namestitev (*Ready to Deploy*). Slednji so, kot že vemo, namenjeni prikazu delovnega toka.

Postavitvenemu stolpcu (*Deploy*) smo se izognili s prikazom relevantnih informacij na preostalem delu igralne plošče, ki se prvič prikaže po zaključku devetega dne in doživi posodobitev vsak zaključek finančnega cikla. S tem smo prihranili na uporabi zaslonske površine, ki je precej bolj omejena od igralne površine, na voljo v namizni inačici.

Ko igro poženemo, nas pričakajo vsi zgoraj opisani grafični elementi, narisani z orodjem Unity. Nad stolpci vidimo ime trenutnega koraka (*standup*), na desni strani stolpcev pa imamo tudi polje z informacijami o preteklih postavitvah, številu naročnikov, trenutnem zaslužku ipd. Poleg tega se inicializirajo vse kartice, ki jih uporabljamo v igri. Več o karticah v naslednjem razdelku, omenimo pa, da sta ob pričetku igre inicializirana kupček neuporabljenih kartic in razporeditev kartic na igralni površini (stanje po osmem dnevu).

3.3.2 Kartice – gradniki in format



Slika 3.2: Primer igralne kartice (S8) s prikaznim besedilom.

Kartice so, tako kot večina interaktivnih igrinih elementov, sestavljene iz dveh delov: vizualne prezentacije in podatkovnega zaledja. V orodju Unity smo najprej oblikovali vizualno predlogo, ki definira višino in širino kartice,

obenem pa vsebuje tekstovna polja za prikaz besedila in prednastavljeno barvno podlago. Barva kartic je odvisna od njihovega tipa, besedilo kartic se nekoliko spreminja s potekom igranja. Primer kartice vidimo na sliki 3.2. V zgornji polovici kartice najdemo nastavljen ID in kratek opis, v spodnjem delu pa lahko vidimo barvno predstavljen preostanek potrebnega dela, razdeljenega na skupine: analiza, razvoj in testiranje.

Vsakič, ko z miškinim kazalcem lebdimo nad neko kartico, se na igralni podlogi pojavi prosojni element z natančnimi podatki (ID, daljši opis dela, dan izbire ipd.) o kartici, prav tako viden na sliki 3.2.

Podatki o karticah se nastavijo ob pričetku igre, glede na predhodno pripravljene informacije. Slednje pripravi učitelj ali inštruktor, v ta namen pa sta na voljo dve tekstovni datoteki: ena opiše začetno postavitev (*TicketsInfo.txt*), druga pa vse še neuporabljene kartice (*DrawDeck.txt*).

```
S1;Add premium subs;Implement premium subscription model.;B;6;9;10;1;0;0;0;9;9;7;7;7;6;5;5;5;3;0;-2;-4;-4;-5
S2;Optimize streaming;Improve streming alghorytms.;B;10;12;13;1;0;0;2;9;9;7;7;7;6;5;5;5;3;0;-2;-4;-4;-5
S3;Implement chat;Add user chat to existing streams.;B;7;8;9;2;0;0;9;9;9;7;7;7;6;5;5;5;3;0;-2;-4;-4;-5
```

Slika 3.3: Primer formata kartic začetne postavitve.

V obeh primerih branja datotek se posamezna vrstica prevede v celotno kartico, format zapisa podatkov pa je drugačen glede na datoteko, saj začetna postavitev kartic zahteva več informacij. Razlike med nastavljanjem kartic nastanejo tudi zaradi drugačnega pristopa v obravnavi vrednosti različnih tipov kartic.

Na sliki 3.3 vidimo primer zapisa treh kartic z začetno postavitvijo. Za razliko od še neuporabljenih kartic, moramo podati še dejansko stanje vsake kartice. Zanima nas torej, kdaj je bila kartica izbrana ter koliko je preostalega potrebnega dela v analizi, razvoju in testiranju, na podlagi teh podatkov pa v kodi določimo stolpec, v katerem se kartica nahaja. Format izgleda takole:

- ID kartice: *S1*;
- kratek (prikazni) opis kartice: „*Add premium subs*“;
- daljši opis kartice: „*Implement premium subscription model*“;

- tip kartice, predstavljen s prvo črko barve, ki ponazarja željeni tip (izbiramo med **W**hite, **B**rown, **O**range): *B*;
- potrebno delo, ločeno na analizo, razvoj in testiranje: *6, 9, 10*;
- zaporedna številka dne, ko je bila kartica izbrana: *1*;
- dejansko stanje preostalega potrebnega dela (analiza, razvoj, testiranje): *0, 0, 0*;
- podatki o vrednosti kartice – *vsi preostali argumenti.*

Format kartic, ki tvorijo igralni kupček, je nekoliko krajši, saj ne potrebujemo dneva izbire (kartica še ni bila izbrana) in dejanskega stanja preostalega truda (ker je enak predpisanemu preostalemu trudu).

Vizualna predloga kartic ima dodeljeno skripto *Ticket.cs*, ki definira kartična polja in metode. Da vsebino kartic lahko nastavimo, se morata izvesti dve metodi v skripti *TicketHandler.cs*, ki v celoti prebereta omenjeni datoteki, razbereta podatke o posamezni kartici in jih v programski zanki aplicirata na vizualno predlogo. Od tu naprej instanciramo vse kartice, ki prikazujejo trenutno stanje na igralni plošči, ostale pa shranimo do takrat, ko jih igralec premakne v izbirni stolpec. Za lažji pregled obeh, vodimo dve globalni tabeli: tabelo kartic v igri in tabelo kartic v kupčku.

3.3.3 Kartice – ovrednotenje tipov

Poglejmo si še ločevanja tipov kartic oz. pripisovanje vrednosti na podlagi izbranega tipa. Kot že vemo, getKaban predvideva štiri tipe. V snovanju igre smo se osredotočili na tri, ki so bili najbolje opisani v pravilih. Naše vodilo pri snovanju kartičnega formata je bila čim večja nastavljivost.

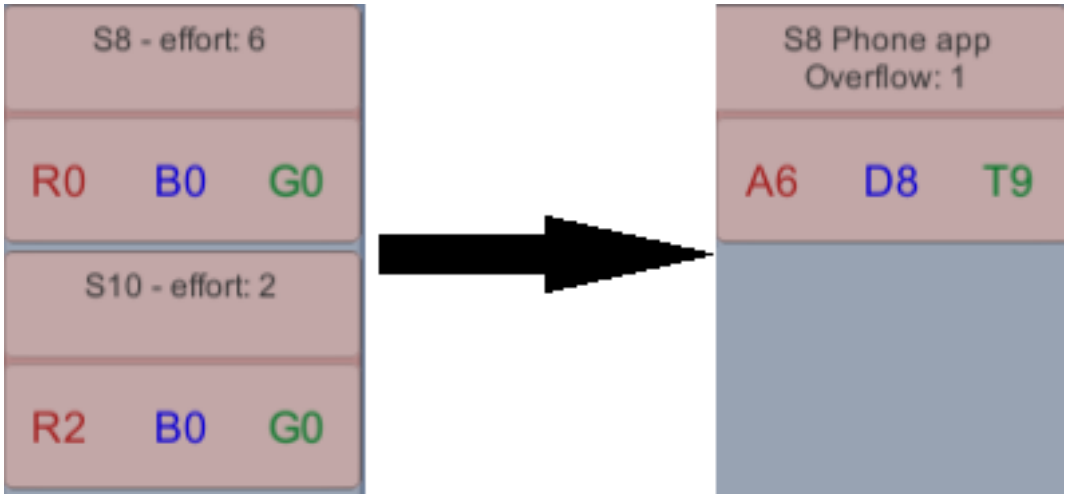
- Kartice standardnih zahtev: nanizamo deset števil, ločenih z vejico; števila predstavljajo naročnike in se shranijo v tabelo, katere indeks predstavlja potrebni čas (1-10). Po postavitvi kartice njen izračunan potrebni čas pove, kje v tabeli naj preberemo število pridobljenih naročnikov.

Če je kartica do postavitve potrebovala več kot deset dni, upoštevamo zadnje mesto v tabeli.

- Kartice nujnih zahtev: nastavimo zaporedno število dneva, ko pričakujemo zaključek dela in dve denarni nagradi; če je kartica postavljena v podanem času, se upošteva prvi, praviloma višji zaslužek, drugače upoštevamo drugo, nižjo nagrado.

- Kartice zahtev z določenim datumom: definiramo zaporedno število dneva, do katerega mora biti kartica zaključena in denarno vsoto, ki je lahko tudi negativna (v primeru globe).

3.3.4 Kartice – interakcija



Slika 3.4: Dodeljevanje kock in presežka.

Kartice lahko poljubno premikamo z mehaniko „*primi in potegni*“ (*drag-and-drop*). Tak pristop je bil vključen po upoštevanju pripomb uporabnikov, ki so testirali funkcionalno zaključeno beta verzijo programa. Ob premiku igra upošteva trenutni položaj kartice in omejitev WIP v naslednjem stolpcu.

Ko je potrebno karticam dodeliti kocke to storimo zelo enostavno: s klikom na izbrano kartico. Levi klik dodeli kocko, desni klik jo odvzame. Po-

dobno poteka dodeljevanje presežka meta kock, s to izjemo, da slednjega ni moč povrniti z desnim klikom. Tako, kot kocke dodeljujemo eno po eno, se tudi presežek prišteva po eno piko naenkrat.

Samo dodeljevanje kock ali presežka poteka na vizualno prilagojenih karticah, kot je to razvidno iz slike 3.4. Na levi strani slike vidimo dve kartici v koraku dodeljevanja kock, kjer zgornja polovica posamezne kartice sedaj prikazuje njen ID in aktualno potrebno delo, spodnji del pa nam barvno poda informacijo o trenutno dodeljenih kockah. Vidimo, da smo kartici *S10* dodelili dve rdeči kocki, kartici *S8* pa nobene.

Na desni strani slike 3.4 je le še kartica *S8*, po opravljenem metu. Kartica *S10* je že v naslednjem stolpcu, saj je zaključila potrebno delo, kartica *S8* pa sedaj prikazuje preliv, ki ji je na voljo, medtem ko njen spodnji del namesto stanja kock, ki sedaj ni več relevantno, kaže spet potrebno delo.

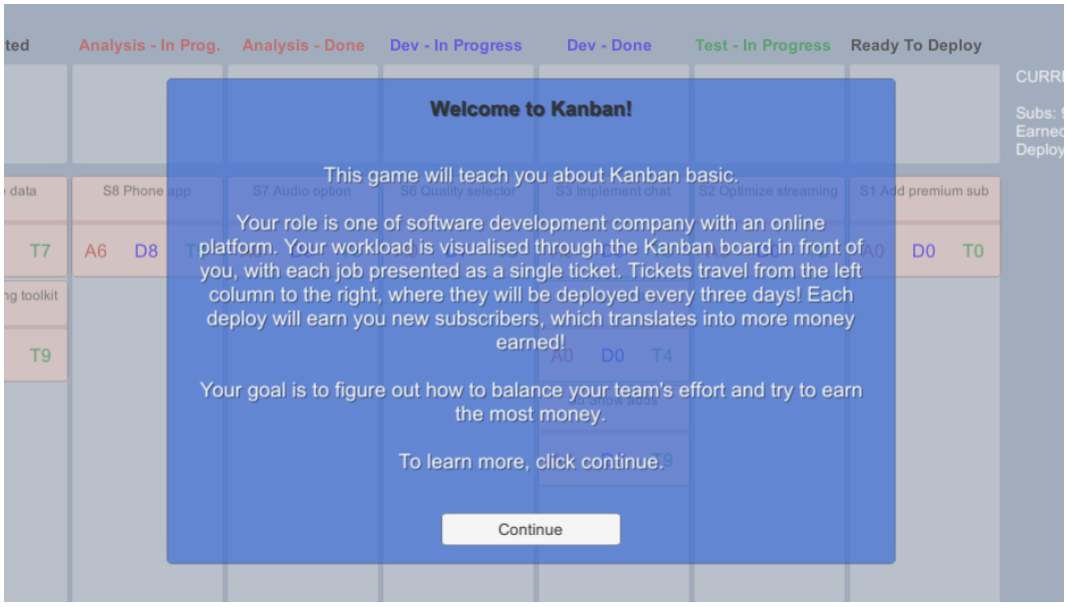
3.3.5 Modalna okna – obvestila, izbira kartic

Modalna okna so v igri uporabljena precej pogosto. Služijo komunikaciji z igralci, podajanju navodil in izbiri novih kartic. Njihovi grafični elementi sestojijo iz dveh glavnih plošč, besedila in gumbov.

Prva izmed obeh plošč je delno prosojna in prekrije vse elemente v ozadju, s čimer prepreči spreminjanje stanja na igralni plošči, dokler modalnega okna ne zapremo. Druga plošča je manjša in vsebuje naslov, besedilo, ter vsaj en gumb za nadaljevanje igre.

Vsi grafični elementi so spet sestavljeni preko funkcionalnosti, ki jih ponuja Unity. Slednji so ustvarjeni že ob pričetku igre, a so igralcu skriti do trenutka, ko jih prikliče interakcija z igralno površino. Primer pozdravnega modalnega okna lahko vidimo na sliki 3.5.

Vsebina modalnih oken je prirejena skozi kodo, v jeziku C#. Bolj kompleksni primer takega prirejanja najdemo, ko odpremo meni za izbiro novih kartic in igra prikaže zadnje tri kartice z vrha kupčka. To stori preko skripte, vezane na modalno okno, ki ob prikazu slednjega iz tabele neuporabljenih kartic prebere zadnje tri položaje. Vsaka izmed prikazanih kartic ima tudi

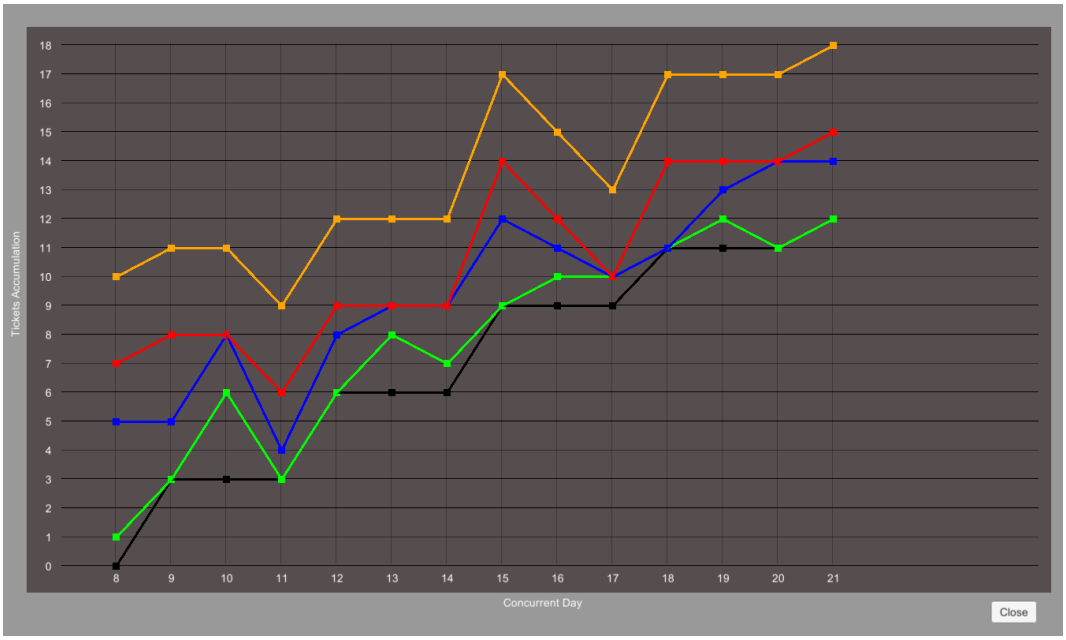


Slika 3.5: Pozdravno modalno okno.

gumb, ki je namenjen njeni izbiri. Ob kliku nanj se kartica preseli na igralno površino, če nam stolpec *Selected* to omogoča (omejitev WIP!).

Dokler imamo na razpolago več kot tri neuporabljene kartice, na površini modalnega okna programsko prikažemo tudi smerni gumb, za premik globje po omenjeni tabeli. Ko se s klikom nanj premaknemo naprej, v kodi preberemo nove položaje v tabeli in posodobimo prikaz kartic. V vsakem trenutku poznamo naš trenutni položaj med karticami, to informacijo pa uporabimo za prikaz še drugega smernega gumba, za brskanje po tabeli v nasprotni smeri.

Efektivno torej gledamo tri izbirne kartice naenkrat, pri čemer se lahko premikamo naprej in nazaj po celotnem naboru kartic. Prikaz izbire je prilagodljiv in nastavi širino modalnega okna glede na ostanek kartic v kupčku: ko sta na voljo le še dve, bosta v nekoliko ožjem oknu tudi prikazani le dve kartici; enako velja za eno kartico.



Slika 3.6: Primer CFD diagrama.

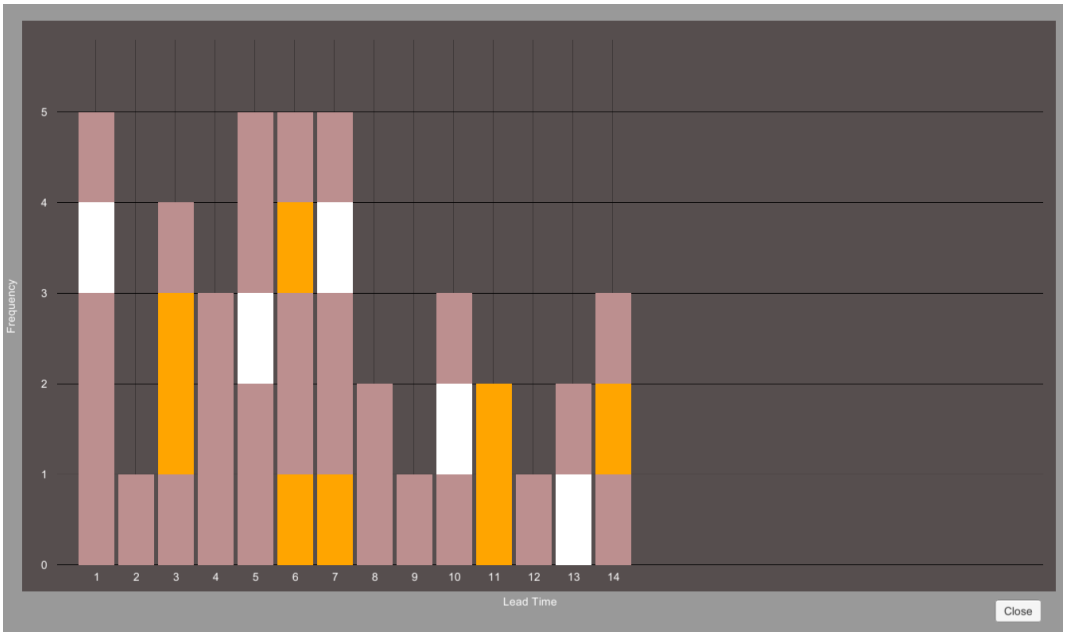
3.3.6 Diagrami

Igra vodi potrebne informacije za izrisovanje dveh diagramov: *CFD* in *LTDC*. CFD se posodobi vsak dan, LTDC pa razširimo le ob novih postavitvah kartic (*deploy*). Oba diagrama sta dosegljiva preko namenskih gumbov na igralni podlagi.

V primeru LTDC smo zgled igre getKanban vizualno razširili; sedaj imamo namesto z barvnimi križci (slika 2.9) opravka s stolpčnim diagramom, ki je bolj smiseln za prikazovanje frekvenčnih razredov. Stolpci so sestavljeni iz manjših barvnih ploskev. Vsaka ploskev predstavlja eno kartico, glede na uporabljeno barvo pa vidimo tudi tip take kartice. Ploskve se dodajajo od spodaj navzgor in po vrsti prikazujejo, kako je naraščala frekvenca izbranega potrebnega časa, kot je to razvidno iz slike 3.7.

CFD po vizualni plati ne odstopa od primera iz namizne igre in sledi enakim barvnim osnovam (slika 3.6).

Oba diagrama prikazujemo dinamično, kar pomeni, da je njuna velikost odvisna od števila in vsebine prikazanih podatkov. Vizualni gradniki obse-



Slika 3.7: Primer LTDC diagrama.

gajo vnaprej pripravljeno modalno okno, ki vsebuje več predlog: sliko osi x, sliko osi y in dve tekstovni polji za oznako vrednosti ob obeh oseh.

Predloge se ob prikazu modalnega okna skozi kodo skripte *ModelPanel-Graphs.cs* prilagodijo izbranemu diagramu. Predlogo za os x npr. uporabimo za kreacijo vseh vzporednic omenjene osi in tako ustvarimo vse vodoravne črte diagrama. Enako velja za navpične črte in nenazadnje tudi za oznake osi. Oznakam glede na vidni diagram nastavimo različno vsebino. Koda risanja obeh diagramov je relativno kompleksna, v veliki meri zaradi dinamičnega prilagajanja velikosti prikaza.

Vhodni podatki so različni za oba diagrama. LTDC pričakuje *zunanji* seznam (*List*, struktura C#), ki mu podamo *notranje* sezname znakovnih nizov (*List<string>*). Znakovni nizi so tipi postavljenih kartic, polja zunanjega seznama pa predstavljajo potrebni čas. Ko npr. postavimo kartico nujne zahteve s potrebnim časom petih dni, se na peto mesto v zunanjem seznamu na konec notranjega seznama shrani niz "W". Kartica standardne zahteve s potrebnim časom treh dni, bi na konec notranjega seznama na tretjem polju

shranila niz "B".

Tudi v obravnavi CFD imamo opravka z zunanjim seznamom, ki vsebuje notranje sezname. Tokrat notranji seznami pričakujejo števila (*List<int>*). Polja zunanjega seznama predstavljajo zaporedne dni razvoja, vsebina notranjih seznamov pa poda število kartic v posamezni kategoriji, kot smo jih našteli v opisu igre getKanban.

3.3.7 Dogodkovne kartice

Ob začetku novega dne igra avtomatsko izbere naslednjo dogodkovno kartico. Dogodkovne kartice so definirane v tekstovni datoteki *EventCards.txt* in omogočajo dodaten vpliv na potek igranja. Nastavi jih ista oseba, ki je nastavila ostale kartice. Vsaka dogodkovna kartica je podana v svoji vrstici, koda pa med njimi izbira po vrsti. Ko program pride do konca nanizanih kartic, se vrne na njihov začetek. Na voljo imamo pet različnih formatov (z manjšimi variacijami), ki krojijo naše možnosti. Poglejmo si jih na primerih.

- **Sprememba preostalega potrebnega dela**: „DEV;3;FLU IS RAMPAGING THROUGH YOUR TEAM, RAISE THE EFFORT NEEDED ON ALL CARDS IN DEVELOPMENT!“ Parametri so ločeni s podpičjem. Prvi del predstavlja izbiro stolpca kartic, drugi del pove spremembo potrebnega dela, zadnji del pa je spremno besedilo dogodkovne kartice. Sprememba truda namesto pozitivnega ali negativnega truda razume tudi znak *, ki pomeni znižanje preostalega potrebnega dela na 0.

- **Sprememba števila naročnikov**: „S;-30;DUE TO RIVAL COMPANY, YOU LOSE SUBSCRIBERS!“ Prvi argument pove, da spreminjamo število naročnikov, drugi predstavlja dejansko spremembo, ki je lahko pozitivno ali negativno število. Zadnji del je spet le spremno besedilo. V primeru negativnih števil ne moremo pasti nižje od 0!

- **Sprememba zaslužka**: „M;300;YOUR WEB PLATFORM GOT A FIRST PRIZE AT PRESTIGIOUS CONVENTION!“ Pristop je enak, kot v prejšnjem

primeru, le da s prvim parametrom povemo, da želimo vplivati kar na obstoječi zaslužek, z drugim pa namesto naročnikov naslavljamo kar denarno vsoto.

- **Zaklepanje omejitve WIP**: „WIP;Due to poor understanding of Kanban and overloaded internal processes you're not allowed to change WIP limits today!“ Kot je razvidno iz besedila, s to kartico zaklenemo obstoječe omejitve WIP za celotni dan.

- **Brez dogodka**: „PASS;Smooth sailing – your team encounters no issues or boons, nothing special happens.“ V tem primeru s prvim parametrom napovemo kartico brez dogodka, ki na igranje nima vpliva.

Dogodkovne kartice so prebrane ob začetku igre s pomočjo skripte *EventCards.sc*, ki jih shrani v tabelo. Tabela je kasneje uporabljena v isti kodi, ki je zadolžena za obravnavo igralnih korakov.

Po upoštevanju lastnosti predstavljene dogodkovne kartice se potek igranja ponovi glede na zgoraj opisano. Igra se odvija vse do zaključka na enaindvajseti dan.

Poglavje 4

Prikaz delovanja programa

Igranje računalniške igre poteka po vnaprej definiranih korakih. Vsak korak pričakuje točno določeno dejavnost, ki si jih bomo pogledali na primerih. Za lažje sledenje poteku igranja je na koncu poglavja na voljo tabela 4.1 z opisi korakov.

4.1 Prvi korak: premikanje in izbira kartic

Ob pričetku igre nas pričaka modalno okno z uvodnim pozdravom in začetnimi napotki, ki smo ga že srečali pri opisovanju komponent v prejšnjem poglavju. Ob zaprtju okna se pojavi naslednje, z navodili za trenutni korak. Izvemo o premikanju in izbiri kartic, ter o tem, kaj nam kartice sploh prikazujejo. Igro začnemo na deveti dan, z razporeditvijo kartic na sliki 4.1.

Stolpec *Test – In Progress* ima le eno kartico (S2), omejitev WIP pa je nastavljena na tri kartice. Ker imamo v stolpcu *Dev – Done* že tri čakajoče kartice, prestavimo kartici S4 in S5 v testiranje.

S premikom kartic smo sprostili razvojna stolpca, ki ima omejitev WIP nastavljeno na štiri kartice. Iz stolpca *Analysis – Done* torej lahko v razvojni stolpec *Dev – In Progress* prestavimo (edino) kartico S7. S tem smo sprostili stolpca analize z omejitvijo WIP na dve kartici. Iz stolpca *Selected* izberemo novo kartico za obravnavo v analizi, tako da kartico S10 prestavimo v stolpec

Slika 4.1: Začetno stanje igralne plošče.

Slika 4.2: Vmesnik za izbiro novih kartic.

Analysis - In Prog..

Preden lahko nadaljujemo, igra od nas zahteva, da napolnimo stolpec *Selected*. Ta ima omejitev WIP nastavljeno na tri kartice, manjkata nam torej še dve. S klikom na gumb *Draw Deck* (kupček novih kartic), se prikaže pregled neuporabljenih kartic (slika 4.2). Izberemo dve, S12 in S12. Ob tem se prikaže gumb za nadaljevanje v naslednji korak.

Slika 4.3: Stanje igralne plošče po premikih in dodeljenih kockah.

4.2 Drugi korak: dodeljevanje kock

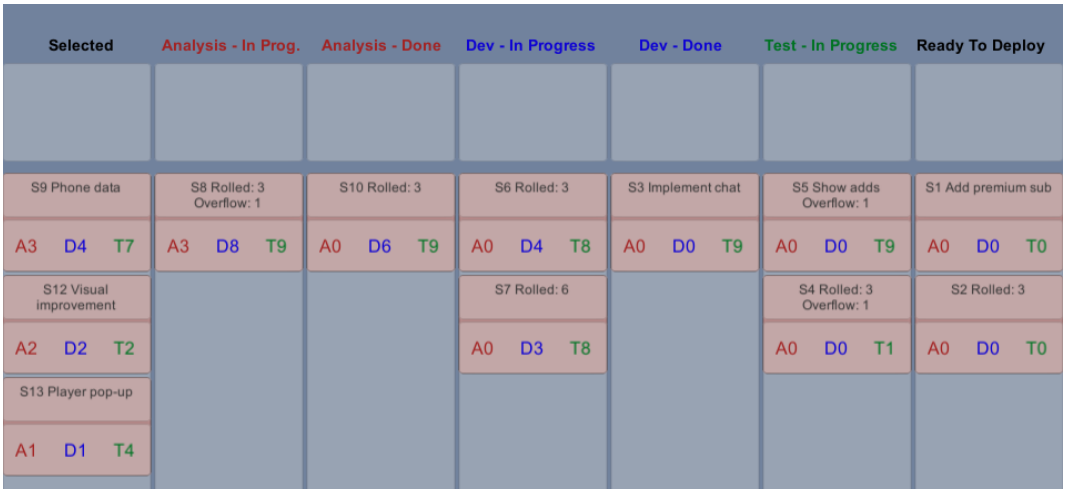
Ob nadaljevanju se prikaže novo okno, ki nam pojasni mehanizme na voljo v trenutnem koraku. Po zaprtju okna vidimo, da kartice, ki jim lahko dodelimo kocke, sedaj prikazujejo prirejene informacije: vidimo aktualno potrebno delo in barvni prikaz dodeljenih kock. Ob desni strani plošče se prikaže število razpoložljivih kock.

Odločiti se moramo, katerim karticam bomo dodelili kocke. Na sliki 4.3 lahko vidimo rezultat naših odločitev. V stolpcu *Test – In Progress* smo

karticama S2 in S4 z najnižjim potrebnim delom dodelili po eno zeleno kocko. Modre kocke imamo kar tri, zato smo dve dodelili kartici z višjim potrebnim delom, S7, preostalo kocko pa je dobila kartica S6. V stolpcu *Analysis – In Prog.* imamo opraviti samo z dvema karticama, prav tako pa imamo na voljo le dve rdeči kocki. Karticama S8 in S10 dodelimo po eno kocko.

Po uporabi vseh kock, nadaljujemo v naslednji korak.

4.3 Tretji korak: met kock in dodeljevanje presežka



Slika 4.4: Rezultati metov s presežki.

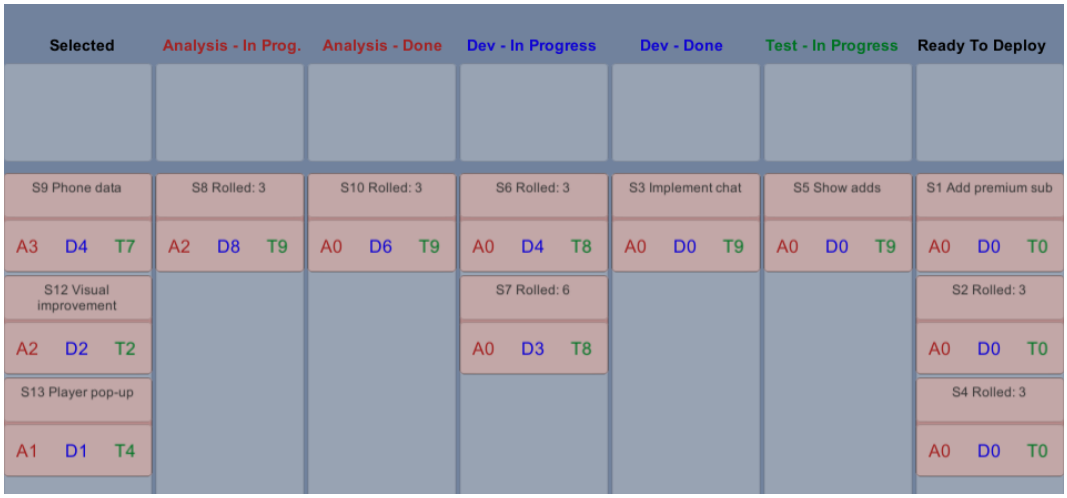
Ponovno nas najprej naslovi okno z navodili za tekoči korak. Predstavitev informacij na karticah se je znova spremenila: tokrat spet vidimo potrebno delo, ki pa se mu pridružijo rezultati metov kock in morebitni presežki. Na sliki 4.4 vidimo, kako se je obnesla dodelitev kock iz prejšnjega koraka.

Met kocke, dodeljene kartici S2, je prinesel tri pike. Kartica je potrebovala dve piki za zaključek potrebnega dela, kar je manj od izvedenega meta, zato je bila prestavljena v naslednji stolpec (*Ready to Deploy*), odvečna pika pa se je na ostalih karticah v stolpcu *Test – In Progress* zabeležila kot presežek.

Kartica S4 v istem stolpcu je potrebovala štiri pike za napredovanje, met pa je prinesel le tri.

V stolpcu *Dev – In Progress* je met kocke, dodeljene kartici S6 prinesel tri od sedmih pik, kartici S7 pa šest od devetih. Presežka ni bilo v nobenem primeru.

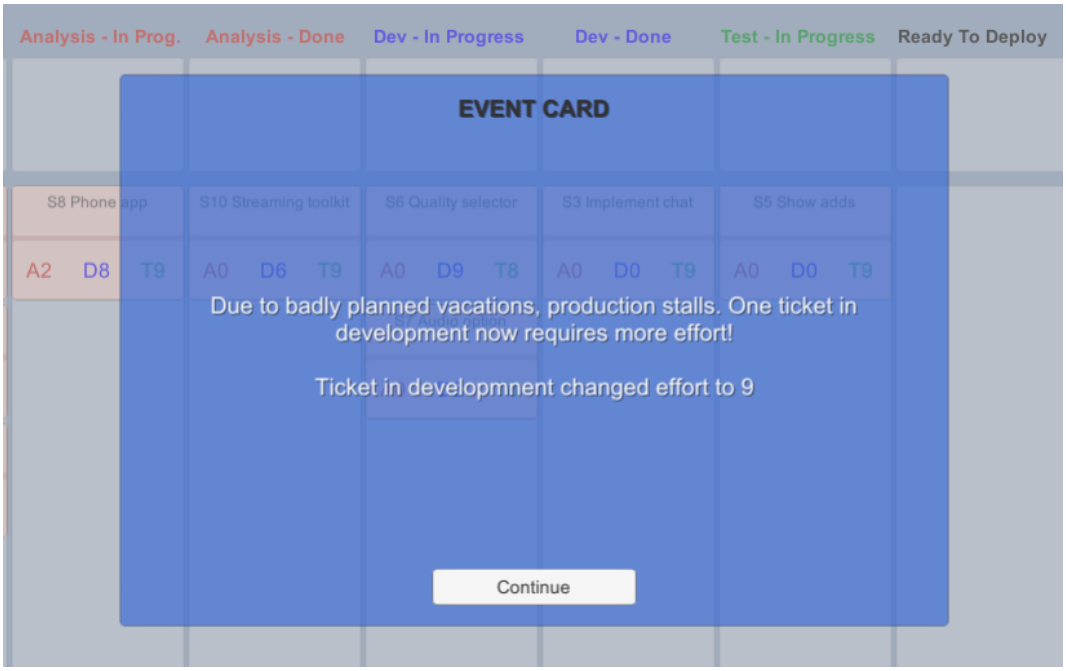
Kartica S10 v stolpcu *Analysis – In Prog.* je potrebovala dve piki, met pa je prinesel eno več. Kartica je bila prestavljena v naslednji stolpec (*Analysis – Done*), presežek pa je označen na kartici S8. Slednja je potrebovala šest pik, a je met vrnil le tri.



Slika 4.5: Stanje po dodeljenih presežkih.

Ostane nam še dodeljevanje presežka. V stolpcu *Test – In Progress* nam je na voljo ena pika, ki jo porabimo s klikom na kartico S4. S tem zaključimo njeno potrebno delo, zato se kartica samostojno premakne v stolpec *Ready to Deploy*. Stolpec *Analysis – In Prog.* prav tako nudi eno piko presežka, ki jo uporabimo na edini prisotni kartici, S8.

Zgoraj opisano stanje po dodeljenem presežku je prikazano na sliki **??**. Nadaljujemo na naslednji korak.



Slika 4.6: Dogodkovna kartica.

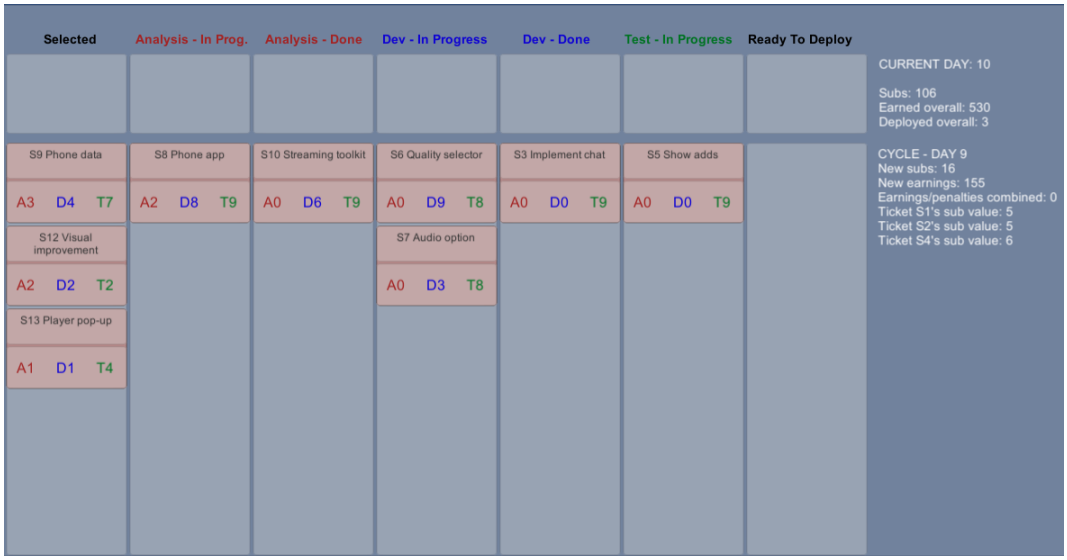
4.4 Četrti korak: zaključek dneva, postavitev kartic

Zadnji korak se izvede v ozadju. Igra ugotovi, da smo zaključili deveti dan, ki obenem predstavlja konec finančnega cikla. Kartice v stolpu *Ready to Deploy* so odstranjene. Izračuna se njihov potrebni čas in vrednost v obliki novih naročnikov. Takšne kartice smatramo za postavljene (*deployed*).

Prvič se posodobita se tudi oba diagrama (CFD, LTDC), zato se prikažeta oba gumba za njun prikaz. Na desni strani igralne podlage, ob stolpcih, se posodobi trenutni dan, stanje naročnikov ter naš celotni zaslužek. Pod temi informacijami se pojavijo nove, ki opisujejo rezultate zadnjega finančnega cikla – število postavljenih kartic, število pridobljenih naročnikov in denarne nagrade ali globe.

Pred pričetkom naslednjega dne se pred igralcem pojavi še dogodkovna kartica. V konkretnem primeru (slika 4.6) smo soočeni s slabo planiranimi

dopusti, ki podaljšajo delo v razvojnem oddelku. Prva kartica v stolpcu *Dev – In Progress* sedaj zahteva devet pik potrebnega dela, kar pomeni, da je slednje naraslo za kar pet pik! Slika 4.7 prikazuje stanje ob pričetku desetega dne.



Slika 4.7: Začetek novega dne.

Prvi korak: *PREMIKANJE IN IZBIRA KARTIC*
• Omogoča izbiro novih kartic. • Omogoča spreminjanje omejitev WIP (pod stolpci). • Možno je premikanje kartic v naslednji stolpec, če so izpolnjeni vsi pogoji. • Po zaključku prvega dne, se ob začetku naslednjega v tem koraku pojavita gumba za prikaz obeh diagramov. • Napredovanje v naslednji korak mogoče le, ko je stolpec *Selected* poln.
Drugi korak: *DODELJEVANJE KOCK*
• Omogoča dodeljevanje kock posameznim karticam. • Že dodeljene kocke se v tem koraku še lahko odstrani z izbrane kartice. • Nadaljevanje v naslednji korak je mogoče šele, ko uporabimo vse kocke.
Tretji korak: *MET KOCK IN DODELJEVANJE PRESEŽKA*
• Ob vstopu v ta korak, se izvede met vseh kock. • Rezultati meta so prikazani na vsaki kartici posebej. • Rezultati meta so upoštevani takoj in se odštejejo od potrebnega dela. • Kartice, ki zaključijo potrebno delo, so takoj prestavljene v naslednji stolpec. • V primeru presežka, je razpoložljivo število pik prikazano na vseh karticah, ki ga lahko koristijo. • Presežek dodelimo s klikom na kartico; izbire ne moremo razveljaviti.
Četrti korak: *ZAKLJUČEK DNEVA, POSTAVITEV KARTIC*
• Postavitev kartic se izvede le vsake tri dni. • Posodobita se oba diagrama (CFD vsakič, LTDC pa le na tri dni). • Ob postavitvi kartic, se izračunajo in prikažejo novi rezultati. • V tem koraku se aktivira dogodkovna kartica.

Tabela 4.1: Tabela korakov računalniške igre.

Poglavje 5

Uporabniške izkušnje

Beta inačica razvite igre je bila predstavljena manjši, zaključeni ekipi programerjev, v obsegu sedmih razvijalcev ter testerja in vodje ekipe. Prostovoljci so stari od 23 do 35 let, njihove izkušnje v industriji pa segajo od dveh do petnajst delovnih let. Razvojni del ekipe programira v jeziku Java. S Kanbanom imajo različne izkušnje, vsi pa so o njem že slišali (na fakulteti, v sklopu agilnih konferenc ipd.). Bolj so domači v SCRUM metodoloških vodah. Večina ni vedela, da je virtualna tabla, ki jo vsakodnevno uporabljajo za prikaz stanja razvoja programske opreme, v resnici osnovni princip Kanbana. Samo ena oseba je poznala Kanban pristop do potankosti.

Udeleženci so igro odigrali v skupini, predhodno so si razdelili vloge. Po igranju so v delno strukturiranem intervjuju podali svoja mnenja – anketam smo se zaradi omejenega števila testne množice zaenkrat izognili. Zanimali so nas vtisi igralcev, osredotočili pa smo se na intuitivnost poteka igranja, na razumevanje navodil in na osvojeno znanje. Preverili smo, kako so doumeli posamezne diagrame in ostale elemente igre. Velja še enkrat poudariti, da je šlo za delovno obliko programa in ne končne verzije.

Spodaj opisani vtisi so botrovali nekaterim popravkom igre na področju uporabniške izkušnje. Sem spada predvsem mehanizem „*primi in potegni*“, ki je igranje naredil bolj intuitivno.

5.1 Oblika delno strukturiranega intervjuja

Zastavljenih smo imeli pet sklopov vprašanj, vsak sklop pa se je nanašal na lastno področje. Področja smo razdelili na intuitivnost igrinih kontrol, razumevanje podanih navodil, privlačnost poteka igranja, kvaliteto povratne informacije in za konec še osvojeno znanje.

V prvem sklopu smo preverili, če so udeleženci imeli morebitne težave pri manipuliranju kartic, predvsem kar se tiče njihovega premikanja in pripisovanja kock ali presežka. Odgovori so bili tesno povezani z naslednjim sklopom, razumevanjem navodil, saj igra skozi njih poskuša igralca postopno naučiti uporabe razpoložljivih mehanizmov. Tu smo torej iskali predvsem potencialne nejasnosti in izboljšave spremnega besedila, glede na uporabniške vtise.

Tretji sklop je imel opravka s samo privlačnostjo igre. Preverili smo ali so dejavnosti v očeh prostovoljcev smiselno razdeljene na korake, obenem pa nas je zanimalo, če je igra dovolj vizualno zanimiva. V prejšnjih sklopih intervjuja smo bili osredotočeni na vsebino, tukaj pa smo dali prednost prezentaciji. Slednja nas je zanimala v povezavi z osredotočenostjo igralcev.

Četrti sklop je ponudil vpogled v kvaliteto povratne informacije – govorimo o spremembah prikazanih podatkov na karticah v odvisnosti od tekočega koraka, interpretaciji diagramov in sprotnem posodabljanju stanja iz dneva v dan (število naročnikov, zaslužek, globe in nagrade). Preverili smo, kako so uporabniki razumeli beleženje kock, presežka in ostalih prikazanih elementov (npr. potrebno delo).

V zadnjem sklopu smo prostovoljce povprašali o osnovnih principih Kanbana. Zanimala nas je pravilnost njihovega razumevanja, osvojeno znanje pa smo primerjali z vtisi udeležencev – so bili mnenja, da so se med igranjem nečesa naučili?

5.2 Intuitivnost, navodila in potek igranja

Sama navodila so igralci razumeli in zato niso potrebovali dodatnih razlag. Kartice, gumbi in potek igranja jim niso predstavljali večjih ovir. Vsak korak

igranja se prične z osvežitvijo relevantnih pravil in mehanik, kar se je izkazalo za dobrodošel pristop.

Kljub razumljenim osnovam igranja, so si igralci zaželeli nekoliko širšo uvodno razlago, še raje pa bi videli voden potek prvega dne – nekaj, kar zaradi časovne omejitve razvojnega obdobja in precejšnjega obsega programskega dela naloge, žal ni bilo izvedljivo. Igre se takih pristopov sicer poslužujejo zelo pogosto, pod nazivom vaje (*tutorial*); gre za didaktično obliko, kjer igralci nastopajo samostojno, a v sklopu zelo natančnih navodil.

V primeru naše igre bi npr. bilo smiselno nanizati takšna navodila – izberi novo kartico, premakni kartico, ki je zaključila razvoj, pripiši kocko k edini kartici v testiranju, ipd. – obenem pa sproti onemogočati vse ostale nerelevantne funkcionalnosti glede na zadnje navodilo. Slednje bi nato sprostili z začetkom desetega dne, po uspešno opravljeni vaji. Na tak način bi zagotovili pravilno razumevanje igre in njenih mehanizmov, brez nevarnosti preobremenitve igralcev z večjo količino informacij naenkrat.

Ker tak pristop ni bil implementiran, predlagamo alternativo: učitelj oz. inštruktor, ki je pripravil kartice, bi lahko prevzel vlogo dajalca napotkov.

Igralci so pokazali, da razumejo vlogo kartic (vizualizacija dela jim je, kot že omenjeno, domača), jasna pa sta jim bila tudi oba diagrama. Dolžina igre se jim je zdela primerna, prav tako so bili zadovoljni z razpletom. Omenjena je bila želja po animacijah in nadgradnji igrinega izgleda, kar bi jih po njihovih besedah dodatno motiviralo.

5.3 Povratna informacija in osvojeno znanje

Nekaj prostovoljcev si je zaželelo boljšo povratno informacijo o pravilnem napredovanju skozi posamezne dni. Z definicijo omenjene *pravilnosti* so sicer imeli precej težav. V skupinskem pogovoru smo razbrali, da s tem mislijo čim bolj optimalno pot skozi igro in deviacije od slednje. Takšen pristop bi potreboval programsko logiko, ki bi spremljala igralčeve odločitve in ga po potrebi opozarjala o morebitnih napakah. Problem takega mehanizma je v

razvrednotenju igre, saj je iskanje boljšega pristopa pravzaprav poglavitno gonilo. Agresivno usmerjanje igralcev bi, po definiciji, vsaj do neke mere prevzelo vlogo odločanja namesto njih. S tem bi bil omejen izziv, ki ga igra trenutno ponuja, kar pa lahko hitro vodi v pomanjkanje igralčevega zanimanja in v slabši fokus.

Kot boljšo pot zgoraj zapisanemu so podali idejo po nadaljnji igrifikaciji – vpeljavo dosežkov (*achievements*). Igra bi v tem primeru še vedno spremljala naš napredek, namesto neposrednega usmerjanja pa bi tiho beležila trenutno stanje na igralni plošči. Ob izpolnitvi vnaprej določenih parametrov bi nas nato lahko nagradila s prirejenim sporočilom na zaslonu. Primer takega sporočila bi bila npr. pohvala ob postavitvi treh ali več kartic ob zaključku finančnega cikla, oz. graja ob ignoriranju neke kartice več dni zapored.

Igralci so razumeli pomembnost vizualizacije in omejevanja sprotnega dela, ni pa jim bilo najbolj jasno, da jim Kanban v resnici pusti proste roke pri iskanju pristopa, ki njihovi ekipi najbolj odgovarja. Oseba, ki je s Kanbanom dobro seznanjena, je bila mnenja, da bi bile Kanban prvine lahko še boljše razložene skozi vizualne elemente igre in manj skozi pritično besedilo, kar se navezuje na idejo o dosežkih in animacijah. Poudarjeno je bilo slabo razumevanje potrebnega časa in iskanje ravnotežja med delovno obremenitvijo v povezavi z njim.

Poglavje 6

Zaključek

6.1 Rezultat – digitalna igra

Rezultat našega truda je delujoča igra. Cilj je bil čim bolj zvest prenos namizne igre na računalniške zaslone, tekom iteracij in razvoja pa smo pristali na nečem tretjem – igri, prirejeni omejitvam in potrebam novega okolja. Na tej točki je torej na mestu vprašanje, kako učinkovit je naš pristop in ali igra potrdi na začetku opisane trditve o igrifikaciji; smo z igranjem res uspeli vplivati na koncentracijo igralcev in obenem izboljšati njihovo znanje?

Študija namizne igre je že leta 2016 prišla do zanimivih rezultatov [7]. Pokazala je delno učinkovitost, najbolj opazno v razlagi Kanban osnov, kot je vizualizacija in proces manipulacije kartic. Žal se je v nadaljnji raziskavi pokazalo, da so udeleženi prostovoljci pogosto napačno precenili svoje osvojeno znanje. Rezultati glede razumevanja diagramov, omejitev WIP in zastojev v razvoju so bili nekoliko bolj spodbudni. Boljše razumevanje je bilo razvidno, negativnim mnenjem nekaterih prostovoljcev navkljub – zaželeli so si namreč natančnejšo razlago diagramov. GetKanban je v okviru raziskave med udeleženimi sicer požel precej uspeha, ne glede na morebitne ugotovitve o efektivnosti.

Ugotovitve raziskave lahko primerjamo z našimi izkušnjami in z vtisi prostovoljcev iz prejšnjega poglavja. Zanimivo je, da je udeležence raziskave

motila dolžina igre, medtem ko naša, precej manjša skupina z dolžino ni imela težav. Razlog se verjetno skriva v diagramih, ki jih naša ekipa ni več risala sama, in so lahko precej zamudni. Ugotovitve raziskave o le delno izboljšanem znanju ne odstopajo od naših ugotovitev. Res je, da smo uspeli podati najbolj osnovne informacije (vizualizacija, omejitev WIP), a so igralci kljub temu imeli težave v razumevanju. Podobno kot v omenjeni študiji igralci niso vedeli, da cilj Kanbana ni optimizacija produkcije v smislu čim večje prepustnosti za vsako ceno, temveč iskanje ravnotežja med potrebnim časom in izkoriščenostjo virov.

Prav tako, kljub trditvam, da so igralci razumeli diagrame, ni bilo opaziti, da bi slednji bili uporabljeni kot podlaga za nadaljnje odločanje ali identifikacijo zastojev – do enakih ugotovitev je prišla omenjena študija.

Kljub temu lahko rečemo, da so se igralci zabavali. Njihov fokus je bil konstanten, med seboj pa so tudi sodelovali kot ekipa. Igra je torej vsaj do neke mere dosegla svoj namen: igralce je pritegnila, jim predočila simulacijo pohitrenega delovnega okolja in jim istočasno predstavila vsaj osnove Kanbana.

6.2 Nadaljnje delo

Prostora za izboljšave je nemalo. Vtisi uporabnikov so nakazali nekaj potrebnih sprememb, med katerimi najbolj izstopa boljša povratna informacija glede naših odločitev v igri. Ker je naša želja izobraziti igralca, je *kvalitetna* povratna informacija bistvenega pomena!

Za boljšo simulacijo realnega sveta bi bilo dobro vključiti manjkajoči tip kartic, naročila z neotipljivo vrednostjo. V okviru izvedbe tega koraka bi nekdo moral definirati kako se neotipljivost lahko manifestira, v kakšnem formatu jo najbolje predstavimo in kako jo nenazadnje upoštevamo v igri.

Na podlagi uporabniških mnenj je bila že nekoliko spremenjena interakcija s karticami, kar se najbolje vidi v implementaciji funkcionalnosti „*primi in potegni*“ (*drag-and-drop*) nad karticami, z namenom prehajanja med stolpci

na igralni podlagi.

Grafična izboljšava bi se začela z vključitvijo oblikovalcev, igra pa definitivno kliče *vsaj* po animatorju. Na ta način bi igrifikacijo lahko dopolnili z grafičnim metanjem kock in s tem način pozitivno vplivali na igralčevo pozornost.

Za konec omenimo še diagrame – ti sicer služijo namenu že v trenutnem stanju, glede na raziskave pa je jasno, da jih v večini primerov igralci niso upoštevali, ko bi jim lahko pomagali pri različnih odločitvah. Potrebno bi bilo torej najti boljši način vključitve diagramov v potek igranja in jih skozi igralne mehanizme približati igralcem, ki bi jih nato dejansko uporabili v razmisleku in organizaciji.

Literatura

[1] The getKanban Board Game – Facilitator's Guide. Dosegljivo: `http://bit.ly/v5-1_0_1`, 2018. [Dostopano: 3. 12. 2018].

[2] Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does gamification work? — a literature review of empirical studies on gamification. *2014 47th Hawaii International Conference on System Science*, 2014.

[3] Henrik Kniberg and Mattias Skarin. *Kanban and Scrum: Making the Most of Both.* C4Media, Publisher of InfoQ.com, 2010.

[4] Martin Kropp, Sonja Hof, and Marla Landolt. Use of gamification to teach agile values and collaboration. *ITiCSE'17*, 2017.

[5] Viljan Mahnič. Kanban in software engineering education: an outline of the literature. *World Transactions on Engineering and Technology Education*, 17(1), 2019.

[6] Bruce Sharlau. Games for teaching software development. *ITiCSE'13*, 2013.

[7] Villie T. Heikkilä, Maria Paasivaara, and Casper Lassenius. Teaching university students kanban with a collaborative board game. *2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion*, 38(1), 2016.

[8] 12th Annual State of Agile Report. Dosegljivo: https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report, 2018. [Dostopano: 1. 2. 2019].

[9] Christiane Gresse von Wangenheim, Rafael Savi, and Adriano Ferreti Borgatto. Scrumia: An educational game for teaching scrum in computing courses. *The Journal of Systems and Software 86 (2013) 2675–268*, 2013.