Izboljšanje pretoka dela v kompleksnem projektu

Tadej Volgemut

Endava, d. o. o., Ljubljana, Slovenija
tadej.volgemut@endava.com

Na kompleksnem projektu v reguliranem okolju smo se sprva osredotočali predvsem na delo posameznikov: kako povečati njihovo učinkovitost in v sprintu opraviti več. Kljub temu so pogoste spremembe zahtev, omejena testna okolja, redke namestitve in čakanje med posameznimi koraki še naprej zmanjševali predvidljivost dostave. Sčasoma smo ugotovili, da glavni problem ni hitrost dela posameznikov, temveč pretok dela skozi celoten sistem. V prispevku predstavljamo, kako smo prepoznali ozka grla, zakaj več ljudi ni nujno pomenilo hitrejše izvedbe ter kako smo Scrum ohranili kot okvir, sprotno delo pa začeli pregledovati in usklajevati z nekaterimi elementi pristopa Kanban. Posebej izpostavljamo vlogo QA pri povezovanju ekip in krajšanju poti do povratnih informacij. Ne predstavljamo dokončne rešitve, temveč izvedene spremembe, njihove prve učinke, praktične omejitve in izboljšave, ki jih še nismo uspeli uvesti.

Ključne besede:
pretok dela
sistemska ozka grla
Scrum
Kanban
dostava programske opreme

Univerzitetna založba Univerze v Mariboru

ISBN 978-961-299-188-3
DOI https://doi.org/10.18690/um.feri.7.2026.9

1 Vsi so delali, delo pa ni teklo

Ko sem se priključil projektu, je ta na prvi pogled deloval zelo aktivno in dobro strukturirano. Potekal je v reguliranem okolju, ob starejši in močno prepleteni kodi, večji ekipi ter omejenem testnem okolju. Razvijalci so razvijali, analitiki so usklajevali zahteve in skrbeli za dokumentacijo, QA je testiral, tehnični vodje pa so izvajali preglede kode in namestitve. Komunikacija z naročnikom je potekala redno. V Jiri [1] je bilo veliko nalog, na koledarju pa je bilo težko najti prost termin. Dela je bilo dovolj za vse.

Postopoma sem opazil vzorec, ki mi je bil znan tudi iz prejšnjih projektov. Ob koncu sprinta je bilo veliko nalog skoraj končanih, le malo pa jih je bilo dejansko pripravljenih za varno predajo. Spremembe so čakale na pregled in združitev kode, kar je bilo zaradi stare in močno prepletene kode že samo po sebi zahtevno. Naloge, pripravljene za namestitev in testiranje, so zahtevale dodatno usklajevanje, saj smo imeli eno testno okolje, na katerem je preverjanje izvajal tudi naročnik. Vmes so prihajale spremembe zahtev, ki so dodatno vplivale na že tako občutljiv načrt.

Prelomna točka zame je bila predaja paketa novih funkcionalnosti. Implementacijo smo zaključili približno v načrtovanem roku, začetno testiranje pa ni pokazalo večjih napak. Med preverjanjem testne različice je naročnik ugotovil, da nekateri njegovi notranji postopki še niso dovolj dobro opredeljeni. Postopke je razjasnil razmeroma hitro, temu pa so sledile spremembe zahtev, medtem ko je rok za produkcijsko namestitev ostal nespremenjen. Nova analiza in ocena sta pokazali, da roka s takšnim načinom dela ne bomo dosegli. Spremembe smo implementirali, za ustrezno testiranje pa je zmanjkalo časa. Morali smo zmanjšati obseg testiranja ter sprejeti s tem povezana tveganja, ki smo jih predstavili tudi naročniku.

V nekaj dneh smo implementirali več novih zahtev, ki so v testiranje prispele skoraj istočasno. Za preverjanje je zato ostalo še manj časa. Na prvi pogled je bilo videti, da je testiranje prepočasno. Ko smo pogledali celoten potek, smo ugotovili, da je težava drugje. Testiranje se je pokazalo kot mesto zastoja, čeprav je večji del čakanja nastal že v predhodnih korakih. Težava ni bila, da se premalo dela, temveč da ga je preveč ostajalo nedokončanega.

V prispevku ne predstavljamo idealnega končnega stanja. Nekatere spremembe smo uvedli, druge le delno, ene pa sploh ne. Nekatere začetne predpostavke smo po prvi analizi rezultatov pozneje popravili.

2 Najprej smo se osredotočili na delo posameznikov

Naše prve razlage so bile precej običajne. Specifikacije morajo biti bolj jasne. Potrebujemo natančnejše načrtovanje sprinta. Razvijalci morajo prej zaključiti razvoj. QA mora hitreje testirati. Morda potrebujemo več ljudi. Ob kritičnem roku mora ekipa še malo stisniti.

Takšni odzivi niso bili nelogični. Ko opazujemo posamezen korak, se zdi smiselno povečati njegovo zmogljivost. Projekt pa je deloval kot povezan sistem. Če bi dodali še enega razvijalca, ne da bi skrajšali čakanje na pregled kode, namestitev ali okolje, bi samo ustvarili več dela, ki bi čakalo. Če bi QA testiral hitreje, spremembe pa bi še naprej prihajale v velikih paketih, bi se pritisk ponavljal v valovih.

Težavo smo najprej obravnavali predvsem z vidika kakovosti: kako bolje organizirati testiranje, hitreje izvesti regresijsko testiranje, s katerim preverjamo vpliv sprememb na že delujoče dele sistema, in preprečiti, da bi nestabilna različica prišla do naročnika. Postopoma je postalo jasno, da težave ne moremo rešiti samo z izboljšanjem testnega procesa. Težava je bila v celotnem dostavnem ciklu.

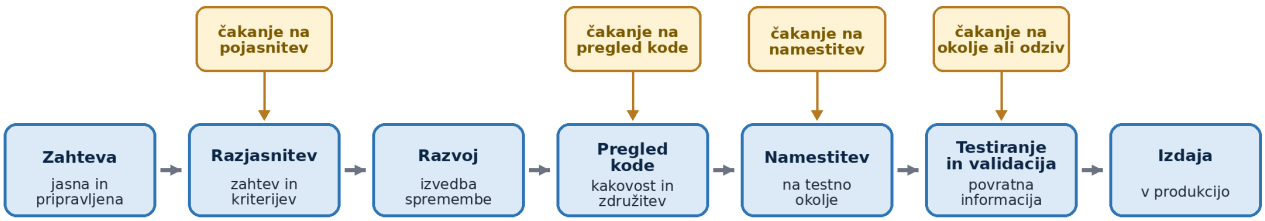
3 Nato smo si zastavili drugačno vprašanje

Načina dela nismo začeli spreminjati z novo metodologijo ali reorganizacijo. Najprej smo podrobneje pogledali, kje delo čaka. Namesto »Kdo še ni končal?« smo se vprašali »Kje se delo ustavlja in zakaj?«

Ko smo podrobneje pogledali celotno pot spremembe od zahteve do uporabnika, so se pokazale čakalne vrste, ki jih samo stanje nalog ni dobro razkrivalo:

- zahteva je čakala na pojasnilo ali ponovno potrditev, ker se je obseg sproti spreminjal,
- sprememba je čakala na pregled kode, pogosto pri zelo omejenem številu ljudi,
- zaključena koda je čakala na združitev in pripravo namestitve,
- testiranje je čakalo na prosto in dovolj stabilno okolje,
- več sprememb se je združilo v večji paket, zato sta bila povratna informacija in odkrivanje napak pozna,
- naročnik je čakal na stabilno različico, hkrati pa je za svoja preverjanja uporabljal isto okolje kot interna ekipa.

Poenostavljen potek z označenimi mesti čakanja prikazuje slika 1.



Slika 1: Poenostavljen prikaz mest čakanja v obravnavanem toku dela

Ko smo proces začeli opazovati kot celoto, smo prišli do ugotovitve, ki jo opisuje tudi teorija omejitev: izboljšanje koraka, ki ne omejuje sistema, ne poveča nujno zmogljivosti celotnega procesa. Najprej je treba prepoznati in razbremeniti omejitev [2]. V našem primeru je to pojasnilo, zakaj dodatni ljudje pri razvoju ali testiranju sami po sebi niso skrajšali časa dostave.

Noben korak sam zase ni pojasnil vseh zamud. Čakanje se je nabiralo skozi več zaporednih korakov. Omejeno število pregledovalcev, redke namestitve, skupno testno okolje, starejša koda z večjim regresijskim tveganjem in zahteve, ki so se spreminjale med izvedbo, so skupaj ustvarili sistem, v katerem je bilo lažje začeti novo delo in precej težje dokončati že začeto.

4 Scrum je ostal okvir, ne napoved dostave

To ni pomenilo, da smo Scrum opustili. Sprinti so ostali okvir za pogovor o prednostnih nalogah in načrtovanje, več pozornosti pa smo namenili dejanskemu toku dela.

Vodnik po Scrumu sprintni nabor opisuje kot načrt, ki se med sprintom posodablja, v enem sprintu pa lahko nastane več uporabnih prirastkov, ki jih je mogoče dostaviti tudi med sprintom [3]. Težava je nastala, ko smo od sprintnega načrta pričakovali zanesljivo napoved, kaj bo ob koncu sprinta pripravljeno za dostavo. Obseg se je spreminjal, nujne zahteve so vplivale na prvotni načrt, okolje in namestitve pa niso omogočali enakomernega toka sprememb.

Pri delu smo uporabili nekatere elemente pristopa Kanban. Ti so se bolje ujemali z načinom dela na tem projektu, saj se delo pogosto ni zaključilo znotraj meja posameznega sprinta. Na rednih skupnih pregledih smo lažje videli, kaj je trenutno v delu, katere naloge čakajo in kje je potrebna pomoč ali dodatno usklajevanje. Pogovori niso bili več osredotočeni predvsem na to, kaj je naredil posameznik, temveč na to, kako posamezno nalogo premakniti naprej. Tako smo lažje sproti preverjali prednostne naloge, usklajevali odvisnosti in usmerjali pozornost v delo, ki je obstalo.

Cilj zato ni bil uvesti druge metodologije, temveč ločiti dve potrebi: sprint je lahko okvir za načrtovanje, dostava pa mora slediti dejanskemu toku dela. Začeti smo morali odpirati manj nalog, hitreje zaključevati tiste v teku in težave obravnavati tam, kjer so nastale, ne šele takrat, ko so dosegle QA ali naročnika.

5 Predlogi in njihov dejanski status

Prve ukrepe smo izbrali na podlagi pregleda čakalnih vrst in izkušenj ekipe. Za višje vodstvo smo pripravili konkretne predloge, usmerjene predvsem v skrajšanje čakanja na ključnih točkah procesa dostave. Vseh nismo izvedli v enakem obsegu, zato tabela 1 ločuje namen predloga od njegovega dejanskega statusa.

Tabela 1: Podani predlogi z argumenti in izidi

Predlog	Argument in dejanski izid	Status
Dodatno testno okolje	Ločiti interno testiranje od naročnikovega preverjanja. Predlog je bil dobro sprejet, vendar še ni bil izveden; po analizi ozkega grla je dobil nižjo prioriteto.	**NE**
Pogostejše in manjše namestitve	Skrajšati povratno zanko in zmanjšati velikost paketov sprememb. Pogostejše namestitve, s tem pa tudi manjše pakete, na testno okolje smo dosegli, pozneje pa ugotovili, da same po sebi niso bile tako nujne, kot smo sprva predvidevali.	**DA**
Razpršitev odgovornosti	Preglede kode, združevanje sprememb in namestitve razdeliti med več ljudi. Dogovor celotne ekipe je zmanjšal odvisnost od posameznika in skrajšal čakanje na pregled kode.	**DA**
Dogovorjeni notranji odzivni časi	Za ključne korake določiti pričakovani odzivni čas. Dogovorili smo se, v kolikšnem času naj bodo zaključeni pregledi kode, nujni popravki in priprava namestitev.	**DA**
Omejitev nedokončanega dela	Pred odpiranjem novih nalog pomagati sprostiti korak, kjer se delo kopiči. Formalne omejitve števila hkrati začetih nalog nismo uvedli, smo pa z rednimi tedenskimi pregledi usmerjali medsebojno pomoč.	**DELNO**
Zgodnejša vključitev QA	QA vključiti v pregled zahtev, oceno tveganj in načrtovanje. Vključenost pri pregledu in potrjevanju zahtev smo dosegli, pri načrtovanju pa le delno.	**DELNO**

Zgodnejša vključitev QA v pregled zahtev ni bila samo organizacijska sprememba. Vlogo QA je razširila iz zaključnega preverjanja v zgodnejše odkrivanje nejasnosti in tveganj. Tak pristop je skladen s proaktivnim zagotavljanjem kakovosti, opisanim tudi v lanskem prispevku OTS [4].

6 Kaj se je spremenilo – in kaj ne

Pri uvajanju izboljšav nismo začeli z razpravo o metodologijah. Izhajali smo iz težav, ki jih je ekipa že občutila: čakanja na pregled kode in namestitve, preobremenjenosti posameznikov, nestabilnih različic in poznih povratnih informacij.

Največji neposredni učinek je imela razpršitev odgovornosti. Pregledi kode, združevanje sprememb in namestitve niso bili več naloga enega človeka. Odgovornost smo razdelili med več članov ekipe. Dogovorjeni notranji odzivni časi so pomagali, da ključni koraki niso ostali v čakalni vrsti. Pregled kode je postal dnevna prednostna naloga, s triažo pa smo določili kritične spremembe, ki smo jih obravnavali najprej. V najslabših primerih je sprememba na pregled čakala nekaj dni. Ta čas smo skrajšali na nekaj ur.

Po razpršitvi odgovornosti in dogovoru o odzivnih časih smo dosegli tudi pogostejše namestitve na testno okolje. Iz ene namestitve na teden smo prešli na najmanj dve in s tem skrajšali povratno zanko. Šele poznejša analiza ozkih grl je pokazala, da pogostost namestitev ni bila tako pomembna, kot smo sprva predvidevali. Ukrep smo torej uspešno izvedli, spremenilo pa se je naše razumevanje njegovega pomena. Zato je tudi dodatno testno okolje dobilo nižjo prioriteto in še ni bilo vzpostavljeno.

Pri ločevanju časa dela od časa čakanja smo ugotovili, da je bilo največ slednjega pri pregledih kode in korakih, povezanih z namestitvijo. V analizi smo ta sklop poimenovali »ozko grlo namestitve«; vanj smo vključili pregled kode, združevanje sprememb, pripravo različice in samo namestitev. To je bilo delovno poimenovanje celotne poti od zaključene implementacije do različice, pripravljene za testiranje, ne le same izvedbe namestitve. Analiza je pokazala, da večina čakanja ni nastala zaradi termina namestitve ali zasedenosti okolja, temveč pri pregledih kode, združevanju sprememb in pripravi različice. Največ aktivnega dela je bilo še vedno v testiranju, predvsem zaradi nestabilnih različic, velikega števila sprememb in ponavljajočega se regresijskega testiranja. Dodatni testerji tega niso rešili, saj je več ljudi samo čakalo na novo različico.

V prejšnjem prispevku smo opisali, kako smo z notranjim testnim ogrodjem in avtomatizacijo del preverjanja kakovosti približali sami implementaciji [5]. Na tem projektu smo podobno načelo razširili na celoten dostavni proces. QA ni spremljal samo testiranja, temveč tudi pot spremembe od zahteve do uporabnika.

Čas dostave se je skrajšal, vendar ne dramatično. Prvi opazni rezultati so bili hitrejši pregledi kode po razpršitvi odgovornosti, krajša povratna zanka in manj frustracij pri usklajevanju z naročnikom. Zmanjšalo se je tudi število napak, ki jih je prijavil naročnik. Ta je izboljšanje večkrat izpostavil na skupnih sestankih. Ker nimamo daljšega niza primerljivih meritev, to obravnavamo kot opažanje, ne kot dokončen dokaz. Pomembno je bilo tudi skupno razumevanje, da preobremenjenost QA, nedokončano delo ob koncu sprinta in nezadovoljstvo naročnika niso tri ločene težave, temveč simptomi istega stanja.

Sprememb zato ne moremo prikazati z eno samo primerjavo stanja pred in po izboljšavah. Lahko pa pokažemo, kateri čakalni časi so se skrajšali in kje omejitve ostajajo.

7 Kaj je ostalo odprto

Prvi rezultati ne pomenijo, da smo odpravili vse omejitve. Na treh področjih vidimo smiselne naslednje korake, vendar jih v opisanem obdobju nismo izvedli v celoti.

Dodatno testno okolje bi ločilo interno preverjanje od naročnikovega in zmanjšalo usklajevanje pri uporabi skupnega okolja. Po analizi smo ugotovili, da to ni bilo glavno ozko grlo, zato je dobilo nižjo prioriteto. Za vzpostavitev bi potrebovali tudi čas ključne osebe, ki ga zaradi drugih obveznosti ni imela na voljo.

Formalna omejitev dela v teku bi jasneje določila, kdaj mora ekipa pred začetkom nove naloge pomagati sprostiti korak, kjer se delo kopiči. Takšna omejitev zahteva skupen dogovor o pravilih in dosledno uporabo na ravni celotne ekipe. Tega dogovora še nismo uvedli; kot praktičen vmesni korak smo uporabili tedenske preglede in usmerjanje medsebojne pomoči.

Doslednejša vključitev QA v načrtovanje bi omogočila zgodnejšo oceno testnega obsega, tveganj in vpliva sprememb na rok. Vključitev smo dosegli pri pregledu in potrjevanju zahtev, pri načrtovanju pa je ostala delna. Redna vključitev QA v načrtovanje bi ob omejeni ekipi hkrati zmanjšala čas, ki je ostal za testiranje, zato te spremembe v opazovanem obdobju še nismo zaključili.

Te usmeritve zato niso seznam idealnih rešitev, temveč nadaljevanje postopnih izboljšav. Njihov vrstni red bomo tudi v prihodnje določali glede na dejansko čakanje in učinek na celoten proces dostave.

8 Kaj lahko začnemo delati drugače

Začeli smo pozno, vendar dovolj zgodaj, da so bili prvi rezultati že vidni. Nekateri so potrdili našo smer, drugi pa pokazali, da smo sprva napačno ocenili pomen posameznih ukrepov. Izkušnjo lahko strnemo v pet praktičnih korakov.

1. *Narišite dejansko pot ene spremembe od zahteve do uporabnika.* Ko smo razjasnili vse faze od zahteve do dostave, smo dobili skupno sliko, na kateri smo lahko gradili nadaljnjo analizo.
2. *Pri vsaki fazi ločite čas dela od časa čakanja.* Tako smo ugotovili, da je bilo največ čakanja pri pregledih kode in namestitvi, največ časa pa smo aktivno porabili za testiranje. To nista isti težavi in ne zahtevata istega ukrepa.
3. *Poiščite mesto, kjer se delo kopiči, in ga ne polnite z novimi nalogami.* Skrbeli smo, da noben razvijalec ne bi ostal brez dela, zato smo sprva po zaključku načrtovanih nalog odpirali nove. S tem smo dodatno polnili vrsto za preglede kode. Razvijalce, ki so zaključili prej, smo nato usmerili v pomoč drugim in v preglede kode.
4. *Izberite majhno spremembo, ki skrajša povratno zanko.* Pri nas sta bili prvi spremembi razpršitev odgovornosti in pohitritev korakov pred testiranjem. Poznejša analiza je pokazala, da pogostejše namestitve in dodatno okolje niso imeli tako visoke prednosti, kot smo sprva mislili.
5. *Uspeha ne merite samo po zasedenosti ljudi.* Pomembnejše je, koliko dela je varno dokončanega in kako rezultat doživlja naročnik. Manj frustracij je bilo za nas pomembno opažanje, čeprav še vedno predvsem opisni in ne številčni pokazatelj izboljšanja procesa dostave.

9 Zaključek

Pri nedokončanem delu ob koncu sprinta je mamljivo najprej pomisliti, kdo bi lahko delal več. Tudi mi smo. Več pritiska ni bistveno izboljšalo stanja, saj je čakanje med fazami ostalo.

Največji premik se ni zgodil takrat, ko smo začeli delati več. Zgodil se je, ko smo začeli opazovati, kje in zakaj delo čaka in kaj lahko kot ekipa naredimo, da bo hitreje prišlo do uporabnika.

S tem vseh omejitev nismo odpravili, za trdne sklepe pa še nimamo dovolj primerljivih meritev. Videli pa smo, da se je čakanje na pregled kode skrajšalo iz nekaj dni na nekaj ur, naročnik pa je izboljšanje večkrat izpostavil na skupnih sestankih. To razumemo kot prvi rezultat postopnega izboljševanja, ne kot dokaz, da smo težavo že rešili.

Vloga QA se je pri tem razširila. Poleg preverjanja izdelka smo začeli spremljati tudi pot spremembe do uporabnika in opozarjati na korake, kjer je delo najdlje čakalo.

Literatura

[1] www.atlassian.com/software/jira, Orodje Jira, obiskano 26. 7. 2026

[2] RAHMAN Shams-ur, »Theory of Constraints: A Review of the Philosophy and Its Applications«, International Journal of Operations & Production Management, let. 18, št. 4, 1998, str. 336–355.

[3] SCHWABER Ken, SUTHERLAND Jeff, »Scrum vodič: Ultimativni vodič o Scrumu – pravila igre«, november 2020, slovenski prevod KUTNIK Voranc, GOLOB Matej

[4] PRAH Luka, KEGL Mihael, »Inovativni procesi zagotavljanja kakovosti razvoja programskih rešitev«, v OTS 2025 – Sodobne informacijske tehnologije in storitve: Zbornik 28. konference, 2025, str. 199–206.

[5] VOLGEMUT Tadej, »Zagotavljanje kakovosti pri integriranju z zunanjimi viri podatkov«, v OTS 2025 – Sodobne informacijske tehnologije in storitve: Zbornik 28. konference, 2025, str. 207–216.