Inovativni procesi zagotavljanja kakovosti razvoja programskih rešitev

Luka Prah, Mihael Kegl

Databox, Ptuj, Slovenija
luka.prah@databox.com, miha@databox.com

Članek obravnava sodobne pristope k zagotavljanju kakovosti (QA) v razvoju programske opreme, s poudarkom na proaktivni vlogi QA in njegovi vključitvi v zgodnje faze projektnega cikla. Predstavljen je koncept "shift-left", ki QA postavlja kot ključnega sogovornika pri analizi zahtev, oblikovanju funkcionalnosti ter uporabniške izkušnje. Prispevek temelji na praktičnih izkušnjah podjetja Databox, kjer QA proces vključuje jasno strukturirane faze: analiza zahtev, načrtovanje testiranja, priprava testnih primerov, raziskovalno in funkcionalno testiranje, avtomatizacija, regresija, poročanje ter poizidne aktivnosti. Posebna pozornost je namenjena tudi uporabi testnih orodij (Qase, Playwright, Asana) in pomenu komunikacije ter sodelovanja z razvojnimi in produktnimi ekipami. V članku so izpostavljeni tudi sodobni izzivi QA, kot so skalabilnost testiranja, upravljanje z regresijami in tehničnim dolgom, ter prilagajanje QA agilnemu razvoju. V zaključku članek ponuja priporočila za uspešno uvajanje inovativnih QA praks, kot so zgodnje vključevanje, CI/CD integracija, odgovornost za kakovost in retrospektiva kot orodje za nenehne izboljšave. Članek dokazuje, da QA ni zgolj faza testiranja, temveč strateški dejavnik, ki bistveno vpliva na kakovost, stabilnost in uporabniško izkušnjo končnega produkta.

Ključne besede:
zagotavljanje kakovosti,
shift-left pristop,
testni načrt,
QA strategija,
avtomatizacija.

Univerzitetna založba Univerze v Mariboru

ISBN 978-961-299-037-4
DOI https://doi.org/10.18690/um.feri.7.2025.18

1 Uvod

Namen članka

V podjetju Databox smo se z namenom izboljšanja stabilnosti in dviga kakovosti produkta lotili sistemske prenove procesa zagotavljanja kakovosti (QA). Članek opisuje ključne faze QA procesa, ki ga izvajamo pri razvoju novih funkcionalnosti in večjih projektov, z vključitvijo QA ekipe že v zgodnji fazi razvoja.

Pomen kakovosti v razvoju programske opreme

Kakovost v razvoju programske opreme pomeni več kot le odpravo napak. Gre za celosten pristop, ki zagotavlja, da končni izdelek izpolnjuje pričakovanja uporabnikov, je stabilen, varen, zmogljiv in dolgoročno vzdrževan. V sodobnem svetu, kjer uporabniki pričakujejo brezhibno delovanje in kjer konkurenca na trgu ne dopušča veliko napak, kakovost postaja, oziroma bi morala postati ključno merilo uspeha.

Razlogi, zakaj je kakovost ključnega pomena:

1. Zadovoljstvo uporabnikov

Uporabniki pričakujejo intuitivne, hitre in zanesljive aplikacije. Slaba uporabniška izkušnja pomeni izgubo zaupanja in posledično strank. Dobra kakovost pomeni večje zadovoljstvo in dolgoročno zvestobo.

2. Zmanjšanje stroškov

Napake, odkrite v poznejših fazah razvoja ali v produkciji, so bistveno dražje za odpravo kot tiste, ki jih zaznamo zgodaj. S tem, ko QA sodeluje že v fazi načrtovanja, se zmanjšajo stroški popravkov in ponovnega razvoja.

3. Hitrejši čas do trga

Zanesljiv QA proces omogoča hitrejše izdaje brez potrebe po dolgotrajnih popravkih po produkciji. Z avtomatizacijo testiranja in dobro pripravljeno testno dokumentacijo lahko ekipa razvija in izdaja hitreje.

4. Zmanjševanje poslovnega tveganja

Slaba kakovost lahko vodi v izgubo podatkov, varnostne ranljivosti, pravne težave ali negativno medijsko pokritost. QA deluje kot zaščitni mehanizem pred takšnimi tveganji.

5. Gradnja dobrega ugleda

Podjetje, ki sistematično izdaja kakovostno programsko opremo, si gradi ugled zaupanja vrednega ponudnika. To neposredno vpliva na konkurenčnost in rast podjetja.

6. Omogočanje skalabilnosti

Dobro strukturiran QA proces omogoča, da podjetje lažje uvaja nove funkcionalnosti, podpira večje število uporabnikov in integrira različne komponente sistema brez ogrožanja stabilnosti.
Vse zgoraj našteto kaže, da kakovost ni strošek, temveč naložba – dolgoročna in strateška. Vlaganje v kakovost pomeni vlaganje v stabilnost, rast in uspeh podjetja. Zato mora biti QA del DNK vsake razvojne ekipe.

2 Kakovost kot strateška komponenta v razvojnem procesu

2.1. Prehod iz reaktivnega v proaktivni QA pristop

Tradicionalno je bila kakovost programske opreme obravnavana kot zadnji korak v razvojnem ciklu – nekaj, kar se preverja tik pred izdajo produkta. V takšnem **reaktivnem pristopu** QA deluje kot "lovilec napak", kar pomeni, da poskuša odkriti napake, ki so se že zgodile med razvojem. Ta pristop pogosto vodi do visokih stroškov popravil, zamud pri izdaji ter nezadovoljstva uporabnikov.
V sodobnem razvoju programske opreme pa kakovost ne more biti zgolj posledica testiranja, temveč mora biti **vgrajena v proces razvoja od samega začetka**. To pomeni prehod na **proaktivni QA pristop**, kjer je QA vključen že v zgodnjih fazah projektnega cikla – pri oblikovanju zahtev, načrtovanju funkcionalnosti in oblikovanju uporabniške izkušnje.
Ključne značilnosti proaktivnega QA pristopa:

- **"Shift-left" filozofija**: QA se pomakne levo po časovnici razvoja – v fazo načrtovanja in analize zahtev. To omogoča zgodnjo identifikacijo potencialnih napak in nejasnosti.

- **Sodelovanje z drugimi vlogami**: QA aktivno sodeluje z produktnimi vodji (PM), oblikovalci (UX/UI) in razvijalci že pri oblikovanju funkcionalnosti. S tem postane pomemben sogovornik pri odločitvah.

- **Preventivno razmišljanje**: Namesto da bi se QA osredotočal le na to, kaj ne deluje, razmišlja o tem, kaj bi *lahko* šlo narobe – in kako to preprečiti.

- **Sistematično načrtovanje testiranja**: Testni načrt nastaja že sočasno z oblikovanjem zahtev. Vključuje oceno tveganj, predlog izboljšav ter določitev strategije testiranja (ročno, avtomatizirano, raziskovalno …).

- **Poudarek na uporabniški izkušnji (UX)**: QA ne testira le tehnične pravilnosti, temveč tudi smiselnost uporabniških tokov, odzivnost, dostopnost in vizualne detajle.

Koristi prehoda na proaktivni QA pristop:

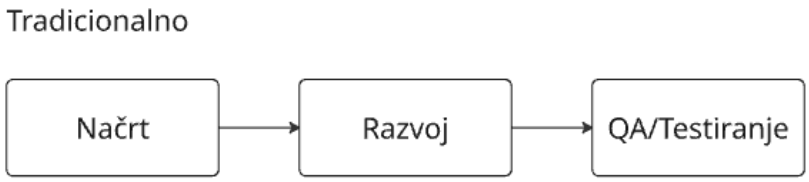
Prehod na proaktivni QA pristop prinaša številne prednosti, med katerimi je ena ključnih bistveno manjše število napak v poznejših fazah razvoja, ko so popravki običajno najdražji in najbolj zamudni. Zaradi zgodnjega vključevanja QA v razvojne aktivnosti se pospeši celoten proces, kar omogoča hitrejši razvoj in krajši čas do izdaje ("time to market"). Poleg tega se izboljša sodelovanje med ekipami, saj QA postane aktivni sogovornik produktnim vodjem, razvijalcem in oblikovalcem. Rezultat takšnega pristopa je višja kakovost končnega izdelka ter manj stresa in urgentnih situacij ob zaključevanju projektov.
Proaktivni QA pristop spreminja vlogo testiranja iz zadnje ovire v strateški gradnik uspešnega razvoja. QA postane ključni partner v procesu odločanja in zagovornik celostne kakovosti – od ideje do produkcije.

Koncept "shift-left" in vključevanje QA že v zgodnjih fazah

Koncept **"shift-left"** označuje premik aktivnosti zagotavljanja kakovosti (QA) **v zgodnejše faze razvojnega procesa**. Gre za zavestno spremembo miselnosti in organizacije dela, kjer QA ni več le faza pred izdajo produkta, temveč integralni del načrtovanja, analize in dizajna.

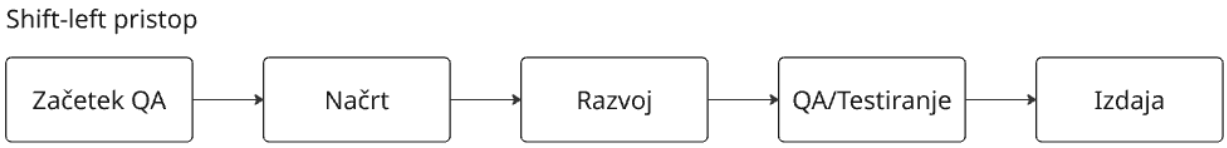
Kaj pomeni "shift-left"?

V klasičnem »waterfall« pristopu QA sledi po fazah razvoja in implementacije. V sodobnih, agilnih okoljih pa se QA premika levo – proti začetku časovnice projekta:



Slika1: Tradicionalni pristop testiranja.

Shift-left pristop: Start QA → Načrt → Razvoj → QA/Testiranje → Izdaja. To pomeni, da je QA vključen že v zgodnjih fazah projekta, kjer sodeluje pri analizi poslovnih zahtev, definiranju kriterijev sprejema (AC), načrtovanju funkcionalnosti in tehnične arhitekture, oblikovanju uporabniške izkušnje (UX/UI) ter oceni tveganj in izvedljivosti.



Slika2: Shift-left pristop testiranja.

Zgodnje vključevanje QA je izjemno pomembno, saj omogoča, da se napake odkrijejo že med razmislekom o funkcionalnosti, ko je njihovo odpravljanje še najcenejše in najmanj moteče. Poleg tega se s tem pristopom izognemo napačnim predpostavkam, ki bi se sicer lahko prenesle v razvoj in povzročile večje težave kasneje. QA s svojim sodelovanjem v zgodnjih fazah pripomore tudi k boljši in natančnejši dokumentaciji, saj zastavlja ključna vprašanja že pri pripravi specifikacij (PRD) ali oblikovalskih zasnov (Figma). Pomembno je tudi, da se testni načrt začne razvijati vzporedno z načrtovanjem funkcionalnosti, kar bistveno pohitri in olajša kasnejšo izvedbo testiranja.

Tabela1: Konkretni primeri vključevanja QA v zgodnje faze.

Faza	Vloga QA
Kick-off projektov	Prisotnost na sestanku, razumevanje vizije, izpostavitev tveganj
Analiza zahtev	Identifikacija testabilnih zahtev, vprašanja o nejasnostih
PRD + Figma	Predlogi izboljšav za UX/UI, priprava testnega skeleta
Ocena virov	Sodelovanje pri planiranju QA virov in časovnic

Pristop "shift-left" prinaša številne prednosti, med katerimi izstopa zmanjšanje stroškov zaradi zgodnjega odkrivanja napak, kar pomeni manj popravkov v kasnejših fazah razvoja. Ob tem se poveča tudi učinkovitost QA ekipe, saj se testiranje izvaja bolj ciljno in pravočasno. Takšen pristop spodbuja boljše sodelovanje z ostalimi

oddelki, kot so produktni vodje (PM), razvojne in oblikovalske ekipe, kar vodi k bolj usklajenemu razvoju. Rezultat je večja kakovost končnega izdelka, obenem pa takšen sistem omogoča lažje in bolj obvladljivo skaliranje QA procesov tudi pri večjih in kompleksnejših projektih.
Koncept "shift-left" omogoča, da QA postane **partner pri oblikovanju izdelka**, ne zgolj nadzornik njegovega delovanja. Vključevanje QA v zgodnjih fazah vodi do premišljenega, zanesljivega razvoja in bistveno boljše uporabniške izkušnje – kar je v končni fazi tudi glavni cilj vsake programske rešitve.

3 Proces zagotavljanja kakovosti pri razvoju večjih funkcionalnosti

3.1. Koraki QA cikla na primeru Databox

Pri razvoju večjih funkcionalnosti je ključno, da je QA cikel jasno definiran, strukturiran in hkrati dovolj prilagodljiv, da omogoča učinkovito sodelovanje vseh vpletenih ekip. V podjetju Databox uporabljamo uveljavljen pristop, ki temelji na preizkušenih korakih zagotavljanja kakovosti skozi celoten življenjski cikel funkcionalnosti – od začetne ideje do izdaje in tudi po njej.
QA cikel se začne z **analizo zahtev**, kjer QA inženir pregleda tehnične in poslovne specifikacije (PRD), Figma dizajne ter aktivno sodeluje na kick-off sestankih. Namen te faze je razumeti obseg funkcionalnosti, izpostaviti morebitne nejasnosti in že zgodaj definirati testabilne kriterije.
Sledi **načrtovanje testiranja**, kjer QA pripravi začetni Test Plan, opredeli strategijo testiranja (ročni testi, avtomatizacija, regresija, raziskovalno), identificira potrebne tipe testov in oceni potrebne vire (čas, okolja, orodja, ljudi). Na tej točki se začne tudi razmislek o tveganjih in prioritetah.
V naslednji fazi QA pristopi k **pripravi testnih primerov**, ki temeljijo na potrjenih kriterijih sprejema. Najprej se zgradi skelet primerov, nato pa se ti dopolnjujejo skozi testiranje, zlasti pri bolj odprtih ali kompleksnih funkcionalnostih.
Vzpostavitev **testnega okolja in testware** je pomembna podlaga za učinkovito testiranje. QA v sodelovanju z DevOps zagotovi, da testno okolje odraža produkcijsko, da so podatki ustrezno pripravljeni, ter da so vzpostavljena vsa potrebna orodja (npr. Qase, Playwright, orodja za beleženje napak).
Testiranje se začne z **raziskovalnim pristopom**, kjer QA prosto raziskuje funkcionalnost in išče očitne vizualne ali uporabniške pomanjkljivosti. To testiranje pogosto razkrije področja, ki jih specifikacije ne pokrivajo, a so za končno izkušnjo pomembna.
Ko so osnovne zadeve preverjene, QA nadaljuje s **funkcionalnim testiranjem**, kjer sistematično preverja vse predvidene tokove uporabe, robne primere in napake. Vse zaznane napake se zabeležijo v orodju (Asana), hkrati pa QA sodeluje pri prioritizaciji njihove odprave.
Po odpravi ključnih napak in ponovnem testiranju le teh, sledi **regresijsko testiranje**, kjer QA preveri, ali nove spremembe niso vplivale na obstoječe delovanje sistema. Ta faza je pogosto podprta z avtomatizacijo – v našem primeru s pomočjo Playwright skript.
Zaključek testiranja vključuje pripravo testnega poročila (kratkega ali razširjenega), preverjanje odprtosti vseh QA nalog in komunikacijo z ostalimi deležniki, da je funkcionalnost pripravljena za produkcijo.
Po izdaji sledijo še aktivnosti, ki vključujejo obvezni dimni test (smoke test) v produkcijskem okolju, spremljanje delovanja (monitoring), sodelovanje v retrospektivi ter posodabljanje testnih primerov glede na novo pridobljene izkušnje. Vse zaznane napake se prenesejo v "Product bugs" projekt, ker se prioritizirajo in planirajo za aktivnosti po izdaji, QA pa zagotovi, da se znanje iz projekta prenese naprej. Le to zagotavljamo tudi z redni QA Guild sestanki, ker se strukturirano in planirano deli znanje z ekipo, ter tudi širše drugim ekipam in obratno. Takšen strukturiran, a prilagodljiv QA proces omogoča zanesljivo uvajanje večjih funkcionalnosti brez ogrožanja kakovosti produkta ali uporabniške izkušnje.

4 Praktični primer: Proces testiranja nove funkcionalnosti

Pri razvoju kompleksnih funkcionalnosti, kot je na primer nov modul za konfiguracijo vizualizacij v analitični platformi, je uspešnost testiranja neposredno povezana s kakovostjo sodelovanja med različnimi ekipami ter z uporabo ustreznih orodij in dokumentacije. V nadaljevanju predstavljamo praktičen potek QA procesa v takšnem projektu.

Ključ do uspeha je zgodnja **vključitev produktnega vodje (PM)**, **oblikovalcev (UX/UI)** in **razvijalcev**, skupaj z QA inženirji. Proces se začne z **izhodičnim sestankom**, kjer PM predstavi specifikacije (PRD), oblikovalci priložijo Figma prototipe, QA pa že na tem mestu zastavi pomembna vprašanja glede robnih primerov, nejasnosti v tokovih ter predlaga izboljšave.

Sledi priprava testnega plana in skelet testnih primerov. Pri tem QA aktivno uporablja orodja, kot so:

- **Qase** [2] za strukturirano dokumentacijo testnih primerov, testnih planov in izvedbo testnih ciklov;
  Slack za tekočo komunikacijo z razvojem, PM in ostalimi deležniki (vključno z objavo napredka in blockerjev v #QA kanalih);
  Asana [3] za organizacijo nalog, označevanje QA pripomb ter beleženje napak in predlogov za izboljšave znotraj projekta.

QA vodi testiranje po korakih: začne z raziskovalnim testiranjem, nato izvede funkcionalne preglede, nato pa pripravi regresijski testni cikel. Med testiranjem aktivno dopolnjuje testne primere, poroča o napakah ter sproti komunicira prioritete z razvojno in produktno ekipo.

Celoten proces se dokumentira v Qase in Asana, končni izdelek pa je testno poročilo, ki vključuje stanje testnih primerov, statistiko uspešnosti, odprte težave ter končno priporočilo glede pripravljenosti za izdajo. Komunikacija v vsaki fazi je ključna – QA redno obvešča vse vpletene o stanju testiranja, tveganjih in morebitnih blokadah.

Ta pristop jasno pokaže, da QA ni le zaključna kontrola, temveč most med načrtovanjem, izvedbo in izdajo – z osredotočenostjo na kakovost, komunikacijo in uporabniško vrednost.

5 Avtomatizacija testiranja

Avtomatizacija testiranja je ključni element sodobnega QA pristopa, še posebej pri agilnem razvoju, kjer so hitre in pogoste izdaje stalnica. Čeprav ročno testiranje ostaja nepogrešljivo pri raziskovalnih in kompleksnih primerih, avtomatizacija bistveno prispeva k hitrosti, ponovljivosti in zanesljivosti testiranja.

Odločitev za avtomatizacijo temelji na več dejavnikih. Prvi pomemben kriterij je pogostost izvajanja testov – scenariji, ki se večkrat ponavljajo, kot so npr. dimni testi ali regresijski prehodi, so idealni kandidati za avtomatizacijo. Prav tako je pomembna stabilnost funkcionalnosti – avtomatiziramo le tiste dele sistema, ki so dovolj dodelani in imajo jasno definirane uporabniške tokove. Poleg tega avtomatizacija omogoča dolgoročen prihranek časa in stroškov, saj nadomešča ročno izvajanje ponavljajočih se testov. Zmanjšuje tudi možnost napak zaradi človeškega faktorja, saj zagotavlja doslednost pri izvajanju.

V podjetju Databox za avtomatizirano testiranje uporabniškega vmesnika, API in vizualnih testov uporabljamo orodje **Playwright** [1], ki omogoča zanesljivo izvajanje end-to-end testov v več brskalnikih. Playwright je prilagojen za moderne spletne aplikacije in omogoča testiranje v okoljih Chromium, Firefox in WebKit. Poleg preverjanja interakcij in vizualnih komponent omogoča tudi napredno validacijo prek nadzora omrežnih zahtevkov in zalednih podatkov.

Izdelava avtomatiziranih testnih skript pa ni zgolj tehnična naloga, temveč zahteva strateški pristop. Ključnega pomena je dobra organizacija kode, na primer z uporabo vzorca Page Object Model, ki ločuje selektorje od logike testa. Prav tako je pomembna ponovna uporaba funkcij in pomočnikov (helperjev), pregledno poimenovanje

testov in pisanje razlagalnih komentarjev za lažje vzdrževanje. Testne skripte je treba redno posodabljati glede na spremembe v uporabniškem vmesniku ali poslovni logiki, da ohranimo njihovo učinkovitost in zanesljivost [4].
Za maksimalen učinek je avtomatizacija vključena tudi v **CI/CD procese**. Integracija z orodji, kot so GitHub Actions, GitLab CI ali Jenkins, omogoča avtomatsko izvajanje testov ob vsaki spremembi kode ali odprtem pull requestu. Na ta način se v zgodnji fazi odkrijejo regresije, preprečijo izdaje z napakami, razvijalci in QA pa prejmejo hitro povratno informacijo. Avtomatizirani testi tako postanejo sestavni del vsakodnevnega delovnega toka in ne zgolj orodje, ki se uporablja občasno.
Avtomatizacija testiranja dopolnjuje ročne preglede in omogoča višjo stopnjo zaupanja v stabilnost programske opreme. Z učinkovito uporabo orodij, kot je Playwright, ter vključevanjem testiranja v CI/CD procese, lahko QA ekipe bistveno povečajo hitrost in zanesljivost razvoja ter omogočijo kakovostne izdaje brez nepotrebnega tveganja.

6 QA kultura in odgovornost

V sodobnih razvojnih okoljih QA ni več zgolj vloga, temveč sestavni del kulture podjetja. Eden najpomembnejših konceptov, ki se v zadnjih letih vse bolj uveljavlja, je **QA kot lastnik kakovosti**. To pomeni, da QA ni tam le zato, da poišče napake, temveč ima popolno odgovornost za kakovost funkcionalnosti – od začetne specifikacije do končne izdaje. QA aktivno skrbi za preverjanje vseh vidikov funkcionalnosti, od tehnične pravilnosti do uporabniške izkušnje, in zagotavlja, da izdelek izpolnjuje visoke standarde kakovosti.
Za uspešno izvajanje takšne vloge je ključna **samoiniciativnost**, saj QA pogosto prevzema naloge brez izrecnih navodil, prepoznava morebitna tveganja še pred razvojem in predlaga izboljšave še preden se pojavijo težave. Enako pomembna sta **transparentnost in sodelovanje z ekipo** – QA mora komunicirati jasno, pogosto in konstruktivno z razvijalci, produktnimi vodji in oblikovalci. Le z odprto komunikacijo in skupnim razumevanjem ciljev je mogoče zagotoviti usklajeno in učinkovito testiranje.
Posebno pomembno vlogo pri vzdrževanju kakovosti in učenju iz preteklih izkušenj ima **retrospektiva**. To ni zgolj formalnost ob zaključku projekta, temveč dragoceno ogrodje za izboljšave. QA na retrospektivo prinese vpogled v to, kaj je šlo dobro, kje so bile težave, kako učinkovita je bila komunikacija in kateri procesi bi lahko bili boljši. Takšna povratna informacija ne koristi le QA ekipi, temveč vsem v projektu, saj omogoča stalno izboljševanje razvojnega procesa.
Kultura kakovosti se torej ne vzpostavi s pravilniki in orodji, temveč z aktivnim vključevanjem QA v vse faze razvoja, z jasno dodeljeno odgovornostjo in s skupno zavezanostjo ekipe k izboljševanju. QA v tem kontekstu ni le »lovilec napak«, ampak partner v razvoju, usmerjen v ustvarjanje najboljših možnih rešitev.

7 Izzivi in priložnosti sodobnega QA

Z razvojem tehnologij, UI (Umetne inteligence), hitrejšimi cikli izdaj in vedno večjimi zahtevami uporabnikov se QA sooča z novimi izzivi – hkrati pa se odpirajo številne priložnosti za nadgradnjo vloge in pristopov. V nadaljevanju so predstavljeni trije ključni vidiki, s katerimi se sodobne QA ekipe pogosto srečujejo.
Eden glavnih izzivov je **skalabilnost testiranja**. Ko produkt raste in postaja kompleksnejši, se povečuje tudi število funkcionalnosti, scenarijev, kombinacij in platform, ki jih je treba preizkusiti. Ročno testiranje vseh možnosti postane neobvladljivo, zato je ključnega pomena uvedba avtomatizacije, pametno upravljanje testnih primerov, prioritizacija testov in uvajanje orodij, ki omogočajo učinkovito razširjanje QA aktivnosti brez linearnega povečevanja števila testerjev.
Poleg tega QA ekipe pogosto prevzemajo breme **upravljanja z regresijo in tehničnim dolgom**. Z vsakim novim dodatkom ali spremembo v kodi se poveča tveganje, da bo nekaj obstoječega prenehalo delovati. Če regresijsko testiranje ni dobro načrtovano ali avtomatizirano, se napake lahko prikradejo v produkcijo. Hkrati je naloga QA, da opozarja na tehnični dolg – zastarele teste, preobremenjeno infrastrukturo za testiranje ali ponavljajoče se težave,

ki jih razvoj še ni naslovil. Proaktivno zaznavanje in beleženje teh področij prispeva k dolgoročni stabilnosti produkta.
Sodobna QA vloga se mora tudi prilagoditi zahtevam **agilnega razvoja**, kjer se funkcionalnosti razvijajo iterativno in v zelo kratkih ciklih. To pomeni, da mora biti QA sposoben hitro razumeti nove zahteve, pripraviti ustrezno testno strategijo in izvajati teste znotraj istega sprinta kot razvoj. V agilnem okolju QA ni več zadnja postaja, temveč aktiven partner v razvojnem procesu – redno sodeluje pri izpiljevanju nabora zahtev, planiranju, pregledu sprintov in dnevnih usklajevanjih.
Kljub številnim izzivom pa prav tu ležijo tudi največje priložnosti QA. S pravo kombinacijo tehničnega znanja, razumevanja produkta, komunikacijskih veščin in strateškega razmišljanja lahko QA postane osrednji člen v razvoju kakovostnih digitalnih rešitev. Z vlaganjem v avtomatizacijo, izboljševanjem procesov in krepitvijo sodelovanja z ostalimi ekipami QA ne samo ohranja svojo vrednost, ampak jo sčasoma še povečuje.

8 Zaključek

Zagotavljanje kakovosti v razvoju programske opreme ni več izolirana aktivnost, temveč postaja osrednji element celotnega razvojnega procesa. V članku smo predstavili sodoben, proaktiven in strateški pristop k QA, ki temelji na zgodnjem vključevanju QA inženirjev v faze načrtovanja, sodelovanju z različnimi oddelki ter uvajanju avtomatizacije in nenehnega izboljševanja. Ključni elementi uspešnega QA procesa vključujejo: jasno analizo zahtev, dobro strukturirane testne načrte, prilagodljivo uporabo testnih orodij, aktivno komunikacijo, učinkovito regresijsko testiranje, testno dokumentacijo, ter poizidne aktivnosti in retrospektivo.
Proaktivni pristop, uveljavljen v podjetju Databox, jasno pokaže, da QA ni le funkcija iskanja napak, ampak partner pri razvoju, odgovoren za stabilnost, uporabniško izkušnjo in dolgoročno kvaliteto produkta.
Na podlagi predstavljenih praks lahko podamo nekaj ključnih **priporočil za uvajanje inovativnih QA procesov**:

- Vključite QA že v fazi načrtovanja funkcionalnosti, kjer lahko prispeva k definiciji zahtev, sprejemnih kriterijev in oceni tveganj.
- Vzpostavite kulturo odgovornosti, kjer QA ni zgolj kontrola, temveč nosilec kakovosti, ki ima moč sooblikovati rešitve.
- Uporabljajte sodobna testna orodja (kot so Playwright, Qase, Testmo, Testrail ipd.), ki omogočajo avtomatizacijo in strukturirano dokumentacijo.
- Poskrbite za CI/CD integracijo testiranja, ki omogoča hitro in zanesljivo validacijo ob vsaki spremembi kode.
- Krepite sodelovanje med QA, razvojem, produktnimi vodji in oblikovalci – kakovost je skupna odgovornost, ne ločena funkcija.
- Ne pozabite na retrospektivo – sistematična analiza opravljenega testiranja je ključno orodje za nenehno izboljševanje procesov.

Sprejemanje inovativnih QA praks pomeni dolgoročno vlaganje v stabilnost, hitrost in kakovost razvoja. Podjetja, ki uspešno vključijo kakovost kot strateško komponento, si zagotovijo konkurenčno prednost, boljšo uporabniško izkušnjo in trajnostno rast.

Literatura

[1] https://playwright.dev/, Microsoft, obiskano julij 2025

[2] https://qase.io/, Qase, obiskano julij 2025

[3] https://asana.com/, Asana, obiskano julij 2025

[4] OTS 2023 Sodobne informacijske tehnologije in storitve: Zbornik 26. konference, Testno ogrodje za razvijalce - ali kako doseči, da razvijalci celovito testirajo, Mitja Krajnc, Tadej Ciglič, Boris Ovčjak, Databox