LEKCIJA 2

Modeli životnog ciklusa softvera - uloga testiranja, DevOps i TDD

Predmet: Testiranje softvera | III razred - Tehničar informacionih tehnologija

U prethodnoj lekciji posmatrali smo SchoolTrip kao softverski proizvod i zaključili da to što aplikacija radi ne znači automatski da je kvalitetna. Sada nas zanima kako softver nastaje i kada testiranje treba da se uključi u razvoj.

Ključno pitanje

Kako je moguće da aplikacija sa očiglednim problemima stigne do korisnika - i šta razvojni tim može da uradi da probleme otkrije ranije?

1. Životni ciklus softvera

Softver ne nastaje samo programiranjem. Od početne ideje do korišćenja proizvoda prolazi se kroz više aktivnosti. Taj tok nazivamo životnim ciklusom softvera.

Pojednostavljen tok: zahtevi -> planiranje i dizajn -> programiranje -> testiranje -> isporuka -> održavanje.

ISO/IEC/IEEE 12207:2026 daje zajednički okvir za procese tokom punog životnog ciklusa softvera - od razvoja i rada do održavanja i povlačenja proizvoda. Standard ne propisuje jedan jedini model razvoja; procesi se mogu organizovati na različite načine.

Primer - SchoolTrip

Zahtev glasi: „Na ekskurziju mogu da se prijave samo učenici od 16 do 18 godina.“ Ako programer napravi aplikaciju, a tester je dobije tek na kraju i tada otkrije da forma prihvata -5 ili 150 godina, problem je postojao ranije, ali je otkriven kasno.

Što se problem kasnije otkrije, može biti potrebno više vraćanja na prethodne aktivnosti: izmena zahteva ili koda, ponovno testiranje i nova isporuka.

2. Kada treba uključiti testiranje?

Testiranje ne mora da počne tek kada je kompletna aplikacija završena. Tester može da analizira i zahteve pre nego što postoji programski kod.

Za zahtev „16 do 18 godina“ tester može odmah da pita:

- Da li su 16 i 18 dozvoljene granične vrednosti?
- Šta se dešava sa učenikom koji ima 15 ili 19 godina?
- Šta ako polje za godine ostane prazno?
- Šta ako korisnik unese negativnu ili nerealno veliku vrednost?
- Da li se proverava broj godina ili datum rođenja?

Zaključak: testiranje je korisno uključiti rano i sprovoditi tokom razvoja, a ne posmatrati ga samo kao poslednju proveru pred objavljivanje proizvoda.

3. Modeli životnog ciklusa

Model životnog ciklusa opisuje način na koji su aktivnosti razvoja organizovane i povezane. Za početno razumevanje korisno je uporediti sekvencijalni i iterativni/agilni pristup.

Waterfall / sekvencijalni pristup	Iterativni / agilni pristup
Aktivnosti uglavnom slede jedna drugu kroz definisane faze.	Proizvod se razvija kroz manje celine i ponovljene iteracije.
Kompletna funkcionalnost se može proveravati kasnije u toku razvoja.	Delovi proizvoda se češće proveravaju i dobijaju se ranije povratne informacije.
Kasne promene mogu zahtevati vraćanje na prethodne faze.	Promene i povratne informacije očekuju se tokom razvoja.

Važno: u praksi postoji više modela i varijanti. Cilj ove lekcije nije da se modeli uče napamet, već da se razume kako organizacija razvoja utiče na trenutak i način testiranja.

4. Ko je odgovoran za kvalitet?

Na razvoju proizvoda mogu učestvovati različite uloge: korisnik ili naručilac, analitičar, projektni menadžer, dizajner, programer, tester/QA i stručnjaci za infrastrukturu i isporuku. Na manjim projektima jedna osoba može imati više uloga.

Tester ima važnu ulogu u proveri proizvoda, ali kvalitet nije odgovornost samo testera. Kvalitet je odgovornost celog tima.

5. DevOps, CI i CD

Kod savremenih proizvoda nove verzije mogu nastajati veoma često. Ako SchoolTrip dobije novu validaciju u ponedeljak, novu funkciju u utorak i izmenu prijave u sredu, nije praktično čekati mesecima da se sve zajedno prvi put proveri.

CI - Continuous Integration (kontinuirana integracija)

Promene programera često se integrišu u zajedničku verziju, uz automatizovane provere koje pomažu timu da brzo uoči probleme nastale novom izmenom.

CD - Continuous Delivery (kontinuirana isporuka)

Ispravna verzija proizvoda održava se u stanju u kome može pouzdano i relativno brzo da bude isporučena korisnicima ili odgovarajućem okruženju.

DevOps

DevOps povezuje razvoj softvera i njegov rad/isporuku kroz saradnju, automatizaciju, česte provere i povratne informacije. DevOps nije samo jedan alat koji se instalira, već način rada i skup praksi kojima razvoj i operacije bliže sarađuju.

6. TDD - Test Driven Development

TDD menja uobičajenu ideju „prvo napiši kod, pa ga testiraj“. Kod TDD-a se za sledeće ponašanje prvo piše test, zatim kod potreban da test prođe, a zatim se kod poboljšava.

Primer za SchoolTrip

Pravilo: učenik može na ekskurziju ako ima od 16 do 18 godina. Pre pisanja funkcije možemo definisati očekivanja:

Ulaz	Očekivani rezultat
15	NE
16	DA
17	DA
18	DA
19	NE

TDD ciklus: RED -> GREEN -> REFACTOR
RED: napišemo test za ponašanje koje još nije implementirano i test ne prolazi.
GREEN: napišemo dovoljno koda da test prođe.
REFACTOR: poboljšamo strukturu koda uz očuvanje ponašanja i prolaznih testova.

Martin Fowler TDD sažima kao ponavljanje tri koraka: napisati test za sledeću funkcionalnost, napisati funkcionalni kod dok test ne prođe i zatim refaktorisati kod.

7. Kako su pojmovi povezani?

Pojam	Suština
Životni ciklus	Softver prolazi kroz povezane procese od ideje do rada, održavanja i povlačenja.
Rano testiranje	Problemi u zahtevima i proizvodu pokušavaju se otkriti što ranije.
Iterativni/agilni razvoj	Razvoj i provere odvijaju se kroz manje celine i česte povratne informacije.
DevOps	Razvoj i rad/isporuka softvera bliže sarađuju uz automatizaciju i česte provere.
CI/CD	Promene se često integrišu, proveravaju i pripremaju za pouzdanu isporuku.
TDD	Test usmerava razvoj sledećeg dela funkcionalnosti: test -> kod -> refaktorisanje.

8. Pitanja za proveru znanja

1. Šta podrazumevamo pod životnim ciklusom softvera?
2. Zašto testiranje nije dobro ostaviti samo za kraj razvoja?
3. Objasni osnovnu razliku između sekvencijalnog i iterativnog/agilnog pristupa.
4. Ko je u razvojnom timu odgovoran za kvalitet softvera?
5. Šta je osnovna ideja DevOps pristupa?

6. Objasni svojim rečima CI i CD.

7. Po čemu se TDD razlikuje od pristupa „prvo kod, pa test“?

8. Objasni ciklus RED - GREEN - REFACTOR.

9. Šta treba zapamtiti

• Softver prolazi kroz životni ciklus - od zahteva do razvoja, rada i održavanja.
• Testiranje treba uključiti rano i sprovoditi tokom razvoja.
• Model razvoja utiče na način organizovanja razvoja i testiranja.
• Kvalitet nije posao samo testera - odgovornost je celog tima.
• DevOps naglašava saradnju razvoja i operacija, automatizaciju i česte povratne informacije.
• CI/CD omogućavaju češće integrisanje, proveravanje i pripremu verzija za isporuku.
• Kod TDD-a test prethodi implementaciji sledeće funkcionalnosti.

Upotrebljena literatura i izvori

1. Živković, M. (2018). Testiranje softvera. Univerzitet Singidunum, Beograd.
Osnovna domaća literatura za predmet; korišćena za opšti okvir testiranja, značaj testiranja i vezu testiranja sa razvojem softvera.

2. ISO/IEC/IEEE 12207:2026. Systems and software engineering - Software life cycle processes.
Aktuelni međunarodni standard koji definiše zajednički okvir procesa životnog ciklusa softvera. Korišćen kao stručna osnova za deo o životnom ciklusu i različitim načinima organizovanja procesa.

3. ISO/IEC/IEEE 29119-1:2022. Software and systems engineering - Software testing - Part 1: General concepts.
Korišćen za savremeni okvir i terminologiju testiranja, uključujući primenu testiranja u različitim metodologijama i životnim ciklusima.

4. ISO/IEC/IEEE 29119-2:2021. Software and systems engineering - Software testing - Part 2: Test processes.
Standard definiše generičke procese testiranja i navodi da se mogu primenjivati u svim modelima životnog ciklusa razvoja softvera.

5. Manifesto for Agile Software Development (2001).
Korišćen kao osnova za ideju iterativnog/agilnog razvoja, česte povratne informacije, funkcionalan softver i reagovanje na promene.

6. Fowler, M. (2023). Test Driven Development.
Korišćen za objašnjenje TDD pristupa i ciklusa Red - Green - Refactor.

Napomena: Lekcija je didaktički prilagođena učenicima III razreda srednje stručne škole. Standardi su korišćeni kao stručna osnova, dok su objašnjenja, primer SchoolTrip i pitanja pojednostavljeni za nastavu.