MODELI ŽIVOTNOG CIKLUSA SOFTVERA

Uloga testiranja • Agile • Scrum • Kanban • DevOps • CI/CD • TDD

Cilj lekcije: da razumeš kako se razvoj softvera organizuje od ideje do održavanja, kako se testiranje uklapa u različite modele razvoja i zašto savremeni timovi koriste kratke cikluse, automatizaciju i stalnu povratnu informaciju.

1. Životni ciklus razvoja softvera

Softver ne nastaje samo pisanjem koda. Od početne ideje do proizvoda koji korisnici upotrebljavaju postoji niz aktivnosti koje zajedno čine životni ciklus razvoja softvera (Software Development Life Cycle - SDLC).

- Analiza zahteva - utvrđuje se šta korisniku treba.
- Planiranje i projektovanje - određuje se kako će sistem biti organizovan i realizovan.
- Implementacija - programeri razvijaju softver.
- Testiranje - proverava se da li softver radi prema zahtevima i otkrivaju se problemi.
- Isporuka - verzija softvera se stavlja na raspolaganje korisnicima.
- Održavanje - ispravljaju se problemi, uvode izmene i nove funkcionalnosti.

Važno: faze ne moraju u svakom projektu da se odvijaju jednom i strogo ovim redom. Način na koji su faze organizovane zavisi od izabranog modela razvoja.

2. Modeli procesa razvoja softvera

Model procesa razvoja opisuje koje aktivnosti postoje, kojim redosledom se izvode i kako se tim vraća na prethodne aktivnosti. Izbor modela utiče i na testiranje: kada se testira, koliko često se testira i koliko brzo se reaguje na pronađene probleme.

Najčešći modeli

Model	Osnovna ideja
Waterfall (vodopad)	Razvoj prolazi kroz unapred određene faze. Naglasak je na završavanju jedne faze pre prelaska na sledeću. Promene koje se pojave veoma kasno mogu zahtevati vraćanje na ranije aktivnosti.
V-model	Sličan je sekvencijalnom razvoju, ali posebno naglašava vezu razvoja i testiranja. Za aktivnosti razvoja planiraju se odgovarajuće aktivnosti provere i testiranja.
Inkrementalni model	Proizvod se gradi u delovima - inkrementima. Svaki novi inkrement dodaje nove funkcionalnosti.
Iterativni model	Rešenje se razvija kroz ponavljanja. Postojeća verzija se proverava, menja i poboljšava u sledećoj iteraciji.
Prototip model	Najpre se pravi pojednostavljena verzija ili prikaz rešenja. Korisnik daje povratnu informaciju, čime se zahtevi bolje razumeju pre razvoja konačnog proizvoda.
RAD	Rapid Application Development naglašava brz razvoj, prototipove, intenzivnu saradnju sa korisnikom i kratko vreme izrade.
Spiralni model	Razvoj se odvija kroz cikluse, uz posebno naglašenu analizu i smanjivanje rizika u svakom ciklusu.
Agilni pristup	Razvoj se organizuje u kratkim ciklusima, funkcionalnost se isporučuje postepeno, a povratne informacije i promene zahteva očekivani su deo rada.

3. Tri modela kroz isti primer - Školska kantina

Zamisli isti proizvod: aplikaciju Školska kantina. Učenik vidi dnevni meni, broj dostupnih porcija, rezerviše obrok i po potrebi otkazuje rezervaciju. Zaposleni u kantini imaju pregled rezervacija. Isti proizvod možemo razvijati na različite načine.

Waterfall

Klijent na početku daje zahteve. Tim prolazi kroz zahteve, dizajn, programiranje i testiranje, a klijent dobija završenu verziju. Ako tek tada zatraži mogućnost otkazivanja rezervacije, promena može zahtevati izmenu već završenih delova i ponovno testiranje.

> Pamti: zahtevi → dizajn → razvoj → testiranje → isporuka.

Prototip

Klijent zna da želi rezervaciju obroka, ali nije siguran kako aplikacija treba da izgleda. Tim prvo pravi jednostavan prototip. Klijent ga pregleda i traži izmene: prikaz broja porcija, detalje obroka, pregled sopstvene rezervacije. Tek kada se ideja razjasni, pristupa se izradi proizvoda.

> Pamti: napravi → pokaži → prikupi povratnu informaciju → izmeni.

Agilni pristup

Klijent ne čeka celu aplikaciju. Najpre dobija mali funkcionalni deo, na primer prijavu i dnevni meni. Zatim se dodaje rezervacija, pa otkazivanje, pa pregled za kantinu. Posle svakog manjeg dela tim dobija povratnu informaciju, testira rezultat i prilagođava prioritete.

> Pamti: mali deo → razvoj → testiranje → povratna informacija → prilagođavanje → sledeći deo.

Osobina	Waterfall	Prototip	Agile
Kada korisnik vidi rezultat?	Kasnije / nakon većeg dela razvoja	Vrlo rano kao prototip	Često, kroz manje funkcionalne celine
Promene zahteva	Kasne promene mogu biti skuplje	Pomažu razjašnjavanju zahteva	Očekivane su i prioritetizuju se
Testiranje	Naglašena posebna faza	Proverava se ideja/prototip, zatim proizvod	U svakom ciklusu razvoja

4. Agile, Scrum i Kanban

U školskoj literaturi može se sresti izraz agilni model. U savremenoj praksi preciznije je govoriti o agilnom pristupu i porodici agilnih načina rada. Osnovna ideja je da tim često isporučuje vredan rezultat, sarađuje sa korisnikom i prilagođava se promenama.

Scrum

Scrum je okvir za organizovanje rada na složenim proizvodima. Rad se odvija u vremenski ograničenim ciklusima - Sprintovima. Za učenike su najvažniji pojmovi: Product Backlog (uređena lista onoga što je potrebno proizvodu), Sprint Backlog (izabrani posao za Sprint) i Increment (upotrebljiv rezultat nastao tokom Sprinta).

- Product Owner - odgovoran za maksimizovanje vrednosti proizvoda i upravljanje Product Backlogom.
- Scrum Master - pomaže timu da razume i primenjuje Scrum.
- Developers - članovi Scrum tima koji stvaraju upotrebljiv Increment.

Napomena: u realnoj organizaciji članovi mogu imati stručnosti kao što su programiranje, dizajn ili testiranje, ali Scrum Guide Scrum Team formalno opisuje kroz tri odgovornosti: Product Owner, Scrum Master i Developers.

Kanban i Trello

Kanban je način vizuelizacije i upravljanja tokom rada. Trello može da posluži kao digitalna Kanban tabla. Kartica predstavlja posao, a kolona stanje kroz koje posao prolazi.

> Primer: PRODUCT BACKLOG → SPRINT BACKLOG → IN PROGRESS → TESTING → DONE

Ako se u koloni TESTING nalaze tri kartice, tim odmah vidi da tri posla čekaju proveru. Vizuelizacija olakšava uočavanje zastoja i dogovor o sledećem poslu.

5. Uloga testiranja u životnom ciklusu

Testiranje nije aktivnost koja mora da počne tek kada programeri završe ceo proizvod. Testne aktivnosti mogu da se planiraju rano, a testiranje se prilagođava modelu razvoja.

- Waterfall: testiranje je tradicionalno jasnije izdvojeno kao kasnija faza nakon implementacije.
- V-model: razvojne aktivnosti se povezuju sa odgovarajućim aktivnostima provere i testiranja.
- Inkrementalni/iterativni razvoj: proverava se svaki novi deo ili nova verzija.
- Agile: testiranje je deo svakog ciklusa i rezultat se često proverava pre nego što se posao smatra završenim.

> Cilj nije samo „pronaći grešku na kraju“, već doprineti kvalitetu tokom celog procesa razvoja.

6. DevOps i CI/CD

DevOps povezuje razvoj (Development) i operacije/isporuku sistema (Operations). Ideja je da razvoj, testiranje, isporuka i praćenje sistema budu bolje povezani, uz što više pouzdane automatizacije.

CI - Continuous Integration

Programeri često integrišu izmene u zajednički kod. Nakon izmene mogu automatski da se pokrenu provere i testovi. Problem se tako može uočiti ranije, dok je promena još mala.

CD - Continuous Delivery / Deployment

CD se odnosi na automatizovanje koraka kojima proverena verzija dolazi do stanja spremnog za isporuku, a u nekim okruženjima i do automatskog postavljanja u produkciju.

> Pojednostavljen tok: izmena koda → integracija → automatske provere/testovi → build → isporuka/postavljanje → praćenje.

QA i DevOps nisu isto: QA je prvenstveno usmeren na kvalitet proizvoda i proveru ponašanja sistema, dok je DevOps usmeren na pouzdan tok izgradnje, isporuke, okruženja i rada sistema. U praksi blisko sarađuju.

7. TDD - Test Driven Development

TDD je praksa razvoja u kojoj se test piše pre koda koji treba da zadovolji taj test. Ideja se često opisuje kratkim ciklusom:

> RED → GREEN → REFACTOR

- RED: napiši test za željeno ponašanje; test pada jer funkcionalnost još ne postoji.
- GREEN: napiši najmanju količinu koda potrebnu da test prođe.
- REFACTOR: poboljšaj strukturu koda bez menjanja njegovog ponašanja; testovi treba i dalje da prolaze.

Primer - Školska kantina: pre implementacije pravila „učenik može rezervisati samo jedan obrok dnevno“ pišemo test koji proverava da druga rezervacija istog učenika mora biti odbijena. Tek zatim razvijamo kod koji ispunjava to pravilo.

8. Šta treba da zapamtiš

- Životni ciklus softvera obuhvata aktivnosti od zahteva do održavanja.
- Model razvoja određuje kako su te aktivnosti organizovane.
- Waterfall, V-model, inkrementalni, iterativni, prototip, RAD, spiralni i agilni pristup predstavljaju različite načine organizovanja razvoja.
- Ne postoji jedan model koji je najbolji za svaki projekat.
- Scrum je okvir za organizovanje agilnog rada; Kanban vizuelizuje i pomaže upravljanje tokom rada.
- Testiranje treba povezivati sa razvojem, a ne posmatrati samo kao poslednju aktivnost.
- DevOps i CI/CD povezuju razvoj, automatizovane provere i pouzdanu isporuku.
- TDD koristi ciklus RED - GREEN - REFACTOR.

9. Pitanja za proveru

1. Šta je životni ciklus razvoja softvera?
2. Zašto izbor modela razvoja utiče na testiranje?
3. Objasni glavnu razliku između Waterfall, prototip i agilnog pristupa.
4. Koja je razlika između inkrementalnog i iterativnog razvoja?
5. Zašto je spiralni model posebno povezan sa rizicima?
6. Šta su Product Backlog i Sprint?
7. Čemu služi Kanban tabla?
8. Šta znače CI i CD?
9. Objasni TDD ciklus RED - GREEN - REFACTOR.
10. Zašto testiranje u agilnom razvoju ne treba ostaviti samo za kraj?

Literatura i izvori

- Živković, M. (2018). Testiranje softvera. Univerzitet Singidunum. Posebno poglavlje 9: Modeli procesa razvoja softvera.
- ISO/IEC/IEEE 29119-1:2022. Software and systems engineering - Software testing - Part 1: General concepts.
- ISO/IEC/IEEE 29119-2:2021. Software and systems engineering - Software testing - Part 2: Test processes.
- Schwaber, K. & Sutherland, J. (2020). The Scrum Guide. Zvanični Scrum Guide; dostupna je i srpska verzija.
- Beck, K. et al. (2001). Manifesto for Agile Software Development.
- Fowler, M. Materijali o Test-Driven Development pristupu i ciklusu Red-Green-Refactor.

Napomena za učenike: literatura služi za razumevanje pojmova. Nije potrebno učiti napamet brojeve delova standarda; važnije je razumeti njihovu svrhu i primenu.