# UPRAVLJANJE ZAHTJEVIMA

Programsko inženjerstvo

Zimski semestar 2025/26

# Što je inženjerstvo zahtjeva?

**Inženjerstvo zahtjeva (engl. *Requirements Engineering*)**

Proces definiranja, dokumentiranja i održavanja zahtjeva za softverski sustav

**Tri ključne aktivnosti:**

Prikupljanje i analiza zahtjeva

Specificiranje zahtjeva u dokumentu

Validacija i verifikacija zahtjeva

*Real-world primjer: Kada je Instagram razvijao Stories funkciju, morali su prvo razumjeti što korisnici žele - kako Snapchat funkcionalnost prilagoditi Instagram ekosustavu?*

https://www.youtube.com/watch?v=xopxod2F8so

# Zašto su zahtjevi KRITIČNO važni?

**Primjeri propalih projekata zbog loših zahtjeva:**

**Snapchat redesign (2018)**

Korisnici nisu bili uključeni - izgubili 3M korisnika u kvartalu (*https://www.youtube.com/watch?v=ShkJ8cWN-sw*)

Kylie Jenner tweet uzrokovao pad od 1.3 milijarde dolara

(*https://www.youtube.com/watch?v=-5xHHnks7IU*)

**Cyberpunk 2077 launch (2020)**

Obećane funkcionalnosti nisu isporučene

Katastrofalne performanse - povučen iz PlayStation Store-a

(*https://www.youtube.com/watch?v=R3R6NFCxb3k* i *https://www.youtube.com/watch?v=qLd2nw0mlpI*)

*Primjer uspješnog projekta:* ***Spotify Wrapped*** *- razumjeli što korisnici vole dijeliti, postao viralni godišnji event (https://www.youtube.com/watch?v=hpwv5tiQGKk)*

# Vrste zahtjeva - Osnove

- Prema sadržaju
  - Funkcijski (*functional requirements*)
  - Nefunkcijski (*non-functional requirements*)
- Prema razini detalja
  - Korisnički (*user requirements*)
    - Na razini s manje detalja (posebno tehničkih)
  - Sistemski (*system requirements*)
    - Na razini s više detalja
- I kombinacije gore navedenih

# Vrste zahtjeva – mjerljive veličine nefunkcijskih zahtjeva

| | |
|---|---|
| Brzina (speed) | Procesiranje u transakcija/sekuda<br>Vrijeme odgovora<br>Vrijeme osvježavanja (ekrana ili podataka) |
| Veličina (size) | Mbyte, GByte, TByte |
| Jednostavnost korištenja (ease of use) | Vrijeme edukacije |
| Pouzdanost (reliability) | Srednje vrijeme do kvara<br>Vjerojatnost neraspoloživosti,nedostupnosti (unavailability)<br>Učestalost pogrešaka |
| Otpornost na pogreške (robustness) | Vrijeme restartanja nakon pogreške<br>Postotak događaja koji uzrokuju grešku |
| Prenostivost (portability) | Broj (popis) odredišnih platformi |

# Vrste zahtjeva – nefunkcijski zahtjevi

- **Nefunkcijski zahtjevi klasificiraju se na:**
  - **Zahtjeve proizvoda**
    - Osiguravaju doprinos u korištenju sustava temeljem neke mjerljive veličine (brzine i sl.)
  - **Zahtjeve organizacije**
    - Osiguravaju doprinos u korištenju sustava koji je usklađen sa odredbama (policies) organizacije
  - **Vanjske zahtjeve**
    - Ostali zahtjevi koji dolaze od nekih vanjskih izvora
    - Npr. pravih regulativa (privatnost čuvanja zdravstvenih podataka, računi firme, FINA )

# Nefunkcijski zahtjevi – TikTok primjer

## Funkcijski zahtjevi za TikTok aplikaciju:

**Autentikacija korisnika i profili**: omogućavanje registriranje i prijavu te održavanje profila

**Video *feed***: omogućavanje neprestane reprodukcije videa

**Kreiranje i *editing* videa**: kreiranje i editiranje videa unutar aplikacije

**Socijalna interakcija**: engagement sa sadržajem ostalih korisnika

**Pretraga i istraživanje**: opcije pretrage specifičnog sadržaja prema zadanim kriterijima

**Notifikacije**: omogućavanje *real-time push* notifikacija

**Live streaming**: podrška *live video broadcasting*

# Nefunkcijski zahtjevi - Uber primjer

## Nefunkcijski zahtjevi za Uber aplikaciju:

**Performanse**: Vrijeme odgovora < 2 sek (95% API poziva)
**Dostupnost**: Uptime 99.9% - max 8.76h downtime/godišnje
**Skalabilnost**: Podrška za 10M+ istovremenih korisnika
**Sigurnost**: End-to-end enkripcija svih transakcija
**Upotrebljivost**: Narudžba vožnje u maksimalno 3 klika

***Uber Black Friday 2023**: Nula downtime uz 300% povećanje prometa!*

# Korisnički vs Sistemski zahtjevi

## Primjer: Revolut instant plaćanja

**KORISNIČKI ZAHTJEV:**
*Želim poslati novac prijatelju i da on to primi u 5 sekundi*

**SISTEMSKI ZAHTJEVI:**
Validacija računa u 200ms
SEPA Instant protokol
256-bit AES enkripcija
Push notifikacija u 1 sekundi
Backup sustav preuzima u < 30 sekundi

# Svojstva DOBRIH zahtjeva

## Dobar zahtjev mora biti:

**JASAN - bez dvosmislenosti**
Loše: "Sustav mora biti brz" | Dobro: "API < 200ms (95%)"

**MJERLJIV - kvantificiran**
Loše: "Puno korisnika" | Dobro: "100,000 istovremeno"

**IZVEDIV - tehnički moguć**
Loše: "AI koja čita misli" | Dobro: "Preporuke iz povijesti"

**PRIORITIZIRAN (MoSCoW) i TESTABILAN**

# Loši primjeri zahtjeva - Što NE raditi

**Primjeri LOŠIH zahtjeva:**

Nedovoljno precizan: "Sustav mora biti koristan korisnicima"
Proturječan: "Jednostavno" + "Sve funkcije kao Excel"
Tehnički nemoguć: "Baterija traje 30 dana uz stalno korištenje"
Nejasan: "Korisnici mogu pretraživati po klinikama"

*Jedan nejasan zahtjev može stajati tjedne dodatnog rada!*

# Formati zapisivanja - User Stories

## USER STORIES - Agile pristup

*Format: Kao [korisnik], želim [akciju], da bih [benefit]*

## Primjeri iz Discorda:

Kao Discord korisnik, želim kreirati privatne kanale, da bih razgovarao samo s odabranim prijateljima

Kao admin servera, želim postaviti automatske odgovore bota, da bih smanjio repetitivne zadatke

*Acceptance Criteria: Max 50 ljudi, vidljiv samo njima, prava se mogu mijenjati*

# Dokumentiranje zahtjeva - alati

**Alati za upravljanje životnim ciklusom aplikacije (ALM) i agilno planiranje**

- *Jira i Confluence (Atlassian)*
- *Azure DevOps (Microsoft)*

**Specijalizirani alati za upravljanje zahtjevima (RM Tools)**

- *ReqView*
- *Jama Connect*
- *CodeBeamer*

**Alati za kolaboraciju i prototipiranje**

- *Figma i Sketch*
- *Miro i Mural*
- *Google Docs i SharePoint*

# Prikupljanje zahtjeva - Intervjui

- **Sastanci i intervjuiranje**
  - Vrste intervjua obzirom na sadržaj
    - Zatvoreni
      - Sadrži unaprijed definiranu listu pitanja
    - Otvoreni
      - Obuhvaća probleme koji nisu unaprijed isplanirani za sastanak
    - Najčešće su kombinacija
      - Obuhvaća i jedno i drugo

# Etnografija - Promatranje u akciji

## ETNOGRAFIJA - Promatranje u prirodnom okruženju

**Prednosti:**
Vidiš kako ljudi STVARNO rade (ne kako kažu)
Otkriješ nesvjesne navike i workaroundove

**Nedostaci: Vremenski zahtjevno, skupo**

**Amazon Go case study:**
*Proveli mjesece promatrajući kupce → otkrili: ljudi mrze čekati → Just Walk Out tehnologija → revolucija u retail industriji!*

# Scenariji i Use Case dijagrami

## SCENARIJI - WhatsApp slanje poruke

### Glavni tijek (Happy Path):

Otvara app → Odabere kontakt → Upiše poruku → Pošalji → Server → Notifikacija → Delivered ✓✓

### Alternativni tijekovi (What if):

Nema interneta? → Queue

Primatelj blokirao? → Error

Server down? → Retry

Poruka prevelika? → Kompresija

# Prototipiranje - Brzo učenje

## PROTOTIPIRANJE - Show, don't tell

**Vrste:**
Low-fidelity (Wireframes) - Figma, Sketch - brzo i jeftino
High-fidelity (Klikabilni) - izgleda kao pravi produkt
MVP - funkcionalni produkt s minimalnim featuresima

**Real-world primjeri:**
Twitter započeo kao SMS prototip unutar tvrtke
Dropbox validirao zahtjeve s YouTube videom prije koda!

# Validacija i verifikacija zahtjeva

## PROVJERA ZAHTJEVA

**Validacija - Gradimo PRAVI produkt? (pregled sa stakeholderima)**

**Verifikacija - Gradimo produkt PRAVILNO? (testiranje i QA)**

**Cijena greške:**

U fazi zahtjeva: 1 EUR | Implementacija: 10 EUR | Nakon releasea: 100+ EUR

*NASA izgubila Mars Climate Orbiter (327M USD) zbog greške u zahtjevima!*

# Upravljanje promjenama zahtjeva

## PROMJENA ZAHTJEVA - Neizbježna realnost

### Zašto se mijenjaju:

Tržište (konkurencija), tehnologija (nove mogućnosti), regulativa (GDPR, AI Act), korisnici nauče više

*Proces: Change Request → Impact Analysis → Approval → Implementacija → Verifikacija*

*Instagram primjer: Dodavanje Reels-a kao odgovor na TikTok - morali promijeniti arhitekturu cijele aplikacije. Sada je Reels #1 feature!*

# Prioritizacija - MoSCoW metoda

- **MoSCoW PRIORITIZACIJA**

- **MUST have - Kritično (npr. Login)**
- **SHOULD have – Važno, ali ne blokira (npr. Reset password)**
- **COULD have - Lijepo za imati (npr. Dark mode)**
- **WON'T have - Ne sada (npr. AI features v2.0)**

- *Spotify: MUST-Play music | SHOULD-Offline | COULD-Lyrics | WON'T-Video podcasts*

- *Steve Jobs: "Innovation is saying no to 1,000 things"*

# Alati za upravljanje zahtjevima

# Zahtjevi u Agile vs Waterfall metodologijama

## DVA PRISTUPA - DVA SVIJETA

**WATERFALL: Svi zahtjevi unaprijed, detaljni SRS, promjene skupne (Banking, avioni)**

**AGILE: Zahtjevi evoluiraju, user stories, kontinuirane promjene (Web/mobile apps)**

**HYBRID: Core zahtjevi fiksirani + implementacija iterativna (Najčešći u praksi)**

*Trend: 71% tvrtki koristi Agile (2024 State of Agile Report)*

# Stvarni primjeri iz industrije

**SUCCESS STORIES:**
Spotify Discover Weekly - 40+ iteracija, 40M korisnika tjedno
Netflix Personalization - 80% *contenta* otkrije se kroz preporuke

**FAILURE STORIES:**
Healthcare.gov (2013) - loši zahtjevi, 55 contractora, 2.1B USD fijasko
Boeing 737 MAX – nekompletani zahtjevi za MCAS, 346 života izgubljeno

***LEKCIJA: Zahtjevi spašavaju ili uništavaju živote!***

# Best Practices - Sažetak

## TOP 10 PRAVILA:

1. Uključi sve stakeholdere rano
2. Dokumentiraj SVE (ako nije zapisano, ne postoji)
3. Budi precizan i mjerljiv
4. Prioritiziraj nemilosrdno (MoSCoW)
5. Validiraj s prototipovima
6. Očekuj promjene
7. Verzioniraj dokumentaciju
8. Testiraj zahtjeve (*acceptance criteria*)
9. Koristi industrijske alate (Jira)
10. Komuniciraj, komuniciraj, komuniciraj!

# Gdje naučiti više?

## RESURSI ZA DALJNJE UČENJE:

**Knjige:**
Software Requirements - Karl Wiegers (Biblija RE)
User Stories Applied - Mike Cohn (Agile)

**Online: Coursera, Udemy, Pluralsight**
**Certifikati: IREB, PMI-PBA, CSPO**

**Praksa: Open source projekti, freelance, vlastiti side projects**

*Najbolji način učenja: raditi na pravim projektima!*

# Pitanja za razmišljanje

## TEME ZA DISKUSIJU

Zašto većina projekata propada zbog loših zahtjeva, a ne tehnologije?

Kako balansirati između što korisnik kaže i što stvarno treba?

Je li bolje imati previše ili premalo dokumentiranih zahtjeva?

Koja je uloga AI-a u budućnosti requirements engineeringa?

# Zaključak - Key Takeaways

**ŠTO SMO NAUČILI:**

Zahtjevi su OSNOVA uspješnog projekta
Loši zahtjevi koštaju milijune i uništavaju reputacije
Funkcijski vs Nefunkcijski - različite ali oboje kritične
Korisnik nije uvijek u pravu - treba razumjeti pravu potrebu
Dokumentacija mora biti živa i evolvirati
Alati (Jira) su standard industrije
Validacija rano štedi novac kasnije
Agile pristup dominira u tech industriji

***Requirements engineering razlikuje dobre od izvrsnih inženjera!***

# Hvala na pažnji!