

Implementirano  $\neq$  testirano  $\neq$  ispravno

Nenad Ajvaz, Stefan Kapunac, Filip Jovanović, Aleksandra  
Radosavljević

Univerzitet u Beogradu, Matematički Fakultet

14. maj 2019

# Primena softvera u svakodnevnom životu

## Softver se danas koristi doslovno svuda

- Administracija
- Edukacija
- Komunikacija
- Industrija
- Saobraćaj
- Ekonomija
- Zdravstvo
- Nauka
- Inženjerstvo
- ...

# Primeri neispravnog softvera

- Therac-25, 1985.
- Marsov orbiter za proučavanje klime, 1999.
- Letovi u Los Angelesu, 2004.
- Boing 737 MAX, 2019.

# Testiranje u razvoju softvera



*„Program testing can be used to show the presence of bugs, but never to show their absence!“*

---

Edsger W. Dijkstra

# Verifikacija softvera

## Definicija

- *verification = Are we building the product right?*
- *validation = Are we building the right product?*

## Podela

- 1 Dinamička verifikacija
  - testiranje crne kutije (funktionalno testiranje)
  - testiranje bele kutije (strukturno testiranje)
- 2 Statička verifikacija
  - simboličko izvršavanje
  - aptraktna interpretacija

# Alati za verifikaciju softvera

- **Alati za automatsko testiranje**

*Obezbeđuju automatsko sprovođenje testova*

- Selenium
- ...

- **Formalni dokazivači ispravnosti**

*Omogućavaju interaktivnu formulaciju matematičkog dokaza korektnosti*

- Isabelle/HOL
- Coq
- ...

# Modeli i metrike pouzdanosti

- ① Deterministički modeli
  - Holstedova metrika
  - Mek-Kejbova ciklometrična složenost
  - ...
- ② Probabilistički modeli
  - Modeli stope neuspeha
  - Modeli rasta pouzdanosti
  - ...

# Holstedova metrika

Obeležje	Opis	Formula
$n_1$	broj jedinstvenih operatora	
$n_2$	broj jedinstvenih operanada	
$n$	vokabular programa	$n_1 + n_2$
$N_1$	ukupan broj operatora	$n_1 \cdot \log_2(n_1)$
$N_2$	ukupan broj operanada	$n_2 \cdot \log_2(n_2)$
$N$	dužina programa	$N_1 + N_2$
$V$	obim programa	$N \cdot \log_2(n_1 + n_2)$
$D$	težina razumevanja i debugovanja	$n_1/2 \cdot N_2/n_2$
$M$	vreme utrošeno na implementaciju	$V \cdot D$
$T$	vreme utrošeno na testiranje	$M/18$
$E$	broj grešaka i bagova	$V/3000$



# Mek-Kejbova ciklometrična složenost

$$C = E - V + 2P$$

E = broj grana grafa G

V = broj čvorova grafa G

P = broj povezanih komponenti grafa G

G = graf kontrole toka programa

# Probabilistički modeli

Pojava greške se posmatra kao verovatnosni događaj.  
Prave se modeli na osnovu ponašanja programa na koje se primenjuju statističke metode.

# Budućnost softvera

Razni alati već danas automatizuju mnoge faze razvoja

- Generisanje koda
- Optimizacija
- Debugovanje

# Veštačka inteligencija

- Sa napretkom veštačke inteligencije i mašinskog učenja, proces razvoja može da se ubrza eksponencijalno
- Već se radi na sistemima koji automatski generišu kod (Bayou)
- Verujemo da će u budućnosti kod da pišu mašine, a zadatak programera će biti samo da kontroliše i usmerava

# Zaključak

- 1 Greške su neizbežne
- 2 Testiranje je važno, ali ne uvek dovoljno
- 3 Formalna verifikacija je ponekad neophodna
- 4 Mašine će zavladatai svetom

01001000 01110110 01100001  
01101100 01100001 00100001  
(Hvala!)