Formalne metode

u softverskom inženjerstvu

00 Informacije o predmetu

ETFBL 24-25

Dunja Vrbaški

Dunja Vrbaški
predavanja online, uživo

Stefan Novaković
Bojan Bulatović
vežbe uživo

moodle

dunja.vrbaski@gmail.com // dunja.vrbaski@uns.ac.rs
(subject: ETFBL ili FMuSI)

Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Katedra za primenjene računarske nauke

PRAVILA POLAGANJA

Pogledati informacije na moodle stranici kursa

TEME

hardver → programski jezik → softver → SE

Računarske nauke:

matematičke ideje, koncepti, modeli, reprezentacije kao osnova računarstva
→
dizajn i razvoj rešenja za konkretne probleme

Logika
Formalni jezici
Teorija automata
Sistemi tipova
Semantika
Merenja
Verifikacija i validacija
...

Ne treba nam “teorija”?

Saznanja koja ne zastarevaju
Garancije zasnovane na naučnom metodu
Postoje i bez tehnologije
Postoje i bez domenskog znanja

Čovekova ideja, znanje, razum, intuicija, iskustvo // Formalne metode

brzo, jeftino // sporo, skupo
jednostavno // zahtevno
rigorozno → opravdano?

Različiti nivoi primene formalnih metoda

Na sredini: primena alata i nekih znanja

Primeri:
- testiranje softvera
- razvoj programskog jezika
- razvoj softvera za, na primer, NASA-u
- razvoj hardvera

Razvoj softvera:

- Specifikacija zahteva (funkcionalnosti, povezanost, opšti zahtevi...)
- Specifikacija rešenja (komponente, komunikacija, interfejsi,...)
- Specifikacija implementacije (različiti delovi sistema)
- Specifikacija testiranja
- Validacija i/ili verifikacija

Provera modela

Sistem se uvek nalazi u nekom stanju. Prelazi iz jednog u drugo stanje.

Model Sistema → pretraživanje prostora stanja → rezultat

Algoritamski verifikovana ispravnost
Ko dokazuje da je dokazivač ispravan?
Složeniji sistemi – uključivanje čoveka, poluautomatski

Primer: programski prevodioci

Softver kao i svaki drugi
Korisnici programeri
Greške u softveru?

Ručno implementiran ad hoc
Ručno impementiran i zasnovani na teoriji
Automatski implementirani delovi zasnovani na teoriji

Primena formalnih metoda u svim fazama
Teorijsko računarstvo “vidljivije” prilikom razvoja

Predmet:

Šta je računar?
Kako računar radi i šta može da uradi?
Koji problemi su za njega teški za rešavanje?

Metode za rešavanje nekih problema

Razvoj logičko-analitičkih veština na apstraktnom nivou.

Razvoj specifičnog softvera koji zahteva određeni formalni način razmišljanja i pristupa problemu.

“teorija” → inženjerski pristup

PRIMER

```
1110101101
10110110101
1101110111011101
```

Šta možemo da kažemo o ovim sekvencama?

```
ispravno:
1110101101
101101101001
1101110111011101

neispravno:
00000
1001
```

Zahtev:

Sekvenca nula i jedinica proizvoljne dužine pri čemu ne postoji podsekvenca od dve nule jedna za drugom.

Ko postavlja ovaj zahtev?

Ispravni i neispravni?

- Koji je misaoni postupak kad to utvrđujemo?
- Kako da automatizujemo? Kako da implementiramo?
- Kako smo sigurni da je dobra automatizacija?
- Šta znači dobra?

PRIMER

```
{{}{}}
{{}{}{}}{{}}
{{}{{}{}}

{
}{}
{}{}{{}
```

Da li su zagrade balansirane?

```
{{}{}}
{{}{}{}}{{}}
{{}{{}{}}

{
}{}
{}{}{{}
```

Kako bismo implementirali?

U čemu je razlika u odnosu na prethodni primer?
Da li nam treba neka memorija?

Opis reči i jezika

Primer: Jezik balansiranih zagrada.
Reči su sastavljenje od zagrada tako da svaka otvorena ima odgovarajuću zatvorenu.

Generisanje reči nekog jezika

Kako se generišu ispravne reči?

Prepoznavanje reči

Da li neka reč pripada jeziku?

Osobine

Da li nešto možemo da zaključimo o jeziku?