Verifikacija softvera

— Briga o kvalitetu softvera —

Milena Vujošević Janičić

Matematički fakultet, Univerzitet u Beogradu

Sadržaj

Pitanja

Pitanja

- Na koji sve način neispravan softver utiče na svet oko nas?
- Šta obuhvata verifikacija softvera?
- Koji su osnovni pristupi i problemi koji se javljaju u okviru verifikacije softvera?
- Koje su specifičnosti automatske statičke verifikacije softvera?
- Kakvi su svetski trendovi u ovoj oblasti?

1 Razvoj softvera i greške

1.1 Razvoj softvera

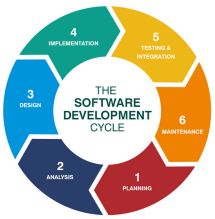
Softver je sastvani deo svih aspekata naših života

Razvoj softvera

Razvoj softvera obuhvata ...
... metode, principe i procedure potrebne da se procesom izrade softvera dođe do efikasnog i pouzdanog proizvoda

Razvoj softvera je ...
... složen proces izrade softvera koji obuhvata veliki broj različitih aktivnosti vezanih za:

- analizu sistema, specifikaciju zahteva,
- projektovanje i implementaciju softvera,
- brigu o kvalitetu softvera (verifikacija i validacija softvera, skraćeno V & V),
- održavanje softvera



Studentski i industrijski projekti

V&V softvera u okviru razvoja studentskog softvera
Proces V&V kod studenata sprovodi se pri kraju procesa razvoja softvera korišćenjem malog broja test primera. Obično čini od 2% do 5% ukupnog napora razvoja.

V&V softvera u okviru razvoja industrijskog softvera
Proces V&V u industriji sprovodi se tokom celokupnog razvoja softvera i čini 30% do 50% ukupnog napora.

Zašto postoji ovako velika razlika?
U čemu su osnovne razlike između studentskih i industrijskih projekata?

Studentski i industrijski projekti

Osnovne razlike

- Dužina upotrebe softvera
- Način upotrebe softvera
- Broj korisnika
- Očekivanja
- Cena pada
- ...

1.2 Briga o kvalitetu softvera

Quality Assurance - QA - briga o kvalitetu softvera

Briga o kvalitetu softvera ...
... je od suštinske važnosti za sve aspekte softvera.

Šta obuhvata briga o kvalitetu softvera?
Šta je kvalitetan softver? Kako se meri kvalitet softvera?

Kvalitet softvera

Šta je kvalitetan softver?
Atributi kojima se meri kvalitet softvera su:

- Statički atributi kvaliteta — strukturiranost koda, cena održavanja koda, mogućnost testiranja koda, prisutnost korektne i kompletne dokumentacije
- Dinamički atributi kvaliteta: pouzdanost (reliability), ispravnost (correctness), kompletnost (completeness), konzistentnost (consistency), lakoća korišćenja (usability), performanse (performance)

Kvalitet softvera

Značenja nekih atributa

- Kompletnost se odnosi na raspoloživost svih osobina koje su tražene u zahtevima ili u korisničkim uputsvima. Nekompletan softver je onaj koji nema implementirane sve zahtevane osobine
- Konzistentnost se odnosi na pridržavanje opšteg skupa pravila i konvencija koje se podrazumevaju (na primer, boje dugmića u intefejsu treba da prate očekivanu konvenciju boja)
- Lakoća korišćenja se odnosi na način korišćenja aplikacije i oslanja se na psihološke karakteristike korisnika
- Performanse se odnose na vreme koje aplikacija koristi da obavi neki traženi zadatak
- ...

Kvalitet softvera

ISO 9126: Evaluation of Software Quality

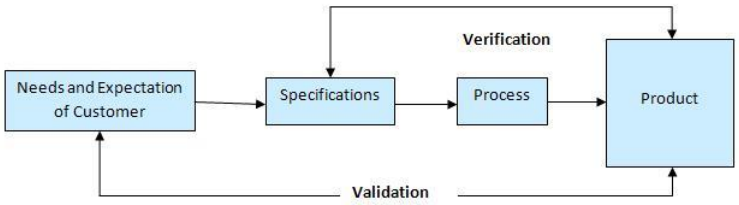
- **Functionality** - *A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.*
  - Suitability
  - Accuracy
  - Interoperability
  - Compliance
  - Security
- **Reliability** - *A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.*
  - Maturity
  - Recoverability
  - Fault Tolerance
- **Usability** - *A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.*
  - Learnability
  - Understandability
  - Operability
- **Efficiency** - *A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.*
  - Time Behaviour
  - Resource Behaviour
- **Maintainability** - *A set of attributes that bear on the effort needed to make specified modifications.* Stability
  - Analyzability
  - Changeability
  - Testability
- **Portability** - *A set of attributes that bear on the ability of software to be transferred from one environment to another.*
  - Installability
  - Replaceability
  - Adaptability
  - Conformance (similar to compliance, above, but here related specifically to portability, e.g. conformance to a particular database standard)

Validacija vs verifikacija

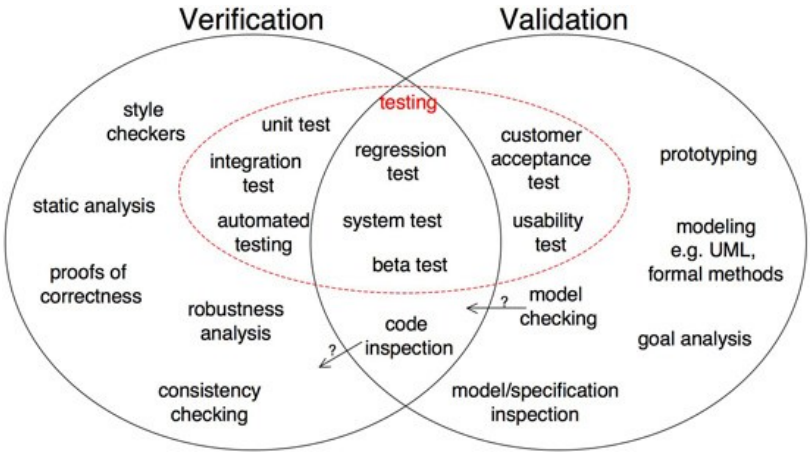
QA obuhvata ...

Validaciju — Da li specifikacija zadovoljava korisničke potrebe? Verifikaciju — Da li softver zadovoljava specifikaciju? (Da li softver sadrži nedostatke? Da li je softver ispravan?)

Validacija vs verifikacija



Validacija vs verifikacija



Validacija vs verifikacija

Šta je važnije?
Validacija ili verifikacija?

http://www.leptonica.com/cachedpages/perlis-epigrams.html

Alan Džej Perlis (engl. Alan Jay Perlis)...
... (Pitsburg, 1. april 1922 — Nju Hejven, 7. februar 1990) je bio američki naučnik koji se bavio računarstvom, poznat po svom pionirskom bavljenju razvojem programskih jezika i kao prvi dobitnik Tjuringove nagrade.

"**It is easier to change the specification to fit the program than vice versa**."

1.3 (Ne)ispravanost softvera

Zašto uopšte postoje greške?

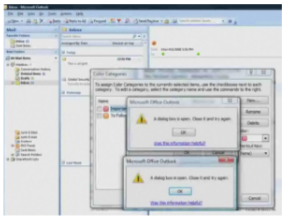
Da li su greške u softveru neminovnost?
Zašto postoje greške u softveru?

Zašto uopšte postoje greške?

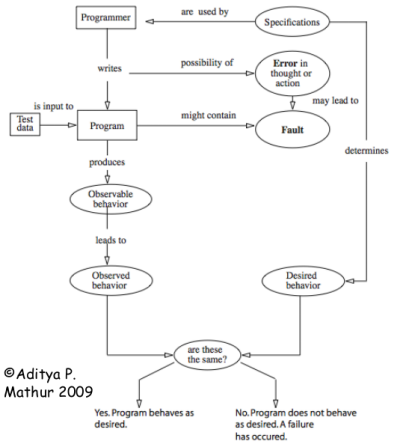
Softver razvijaju ljudi

- Greške su sastavni deo naših života
- Ljudi prave greške u razmišljanju, akcijama i proizvodima
- Greške se pojavljuju svuda gde ljudi sprovode akcije i prave odluke

Greške su svuda oko nas

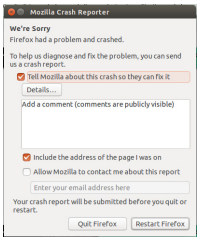


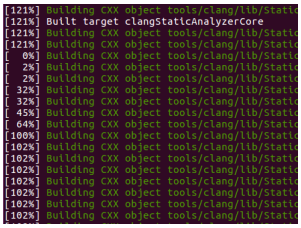
Greška $\overset{proizvodi}{\rightarrow}$ **Nedostatak** $\overset{uzrokuje}{\rightarrow}$ **Pad** $\overset{pravi}{\rightarrow}$ **Incident** $\overset{posledice}{\rightarrow}$ **...**



Na koji sve način neispravan softver utiče na svet oko nas?

Neprijatnosti
Mobilni telefoni, Internet pregledači, muzički uređaji...



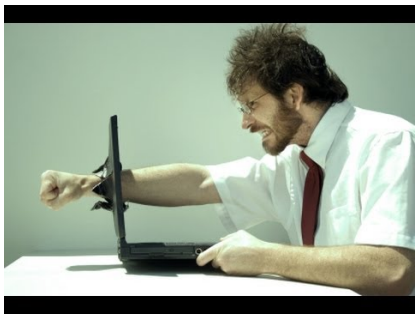


Neprijatnosti
Apple Maps gives us directions to nowhere (2012)

Koliko košta neprijatnost?

Postoji veliki izbor, vreme tolerancije grešaka je prošlo:

- 50% korisnika obriše mobilnu aplikaciju zbog samo jedne greške
- 50% aplikacija se skine i koristi samo jednom



Na koji sve način neispravan softver utiče na svet oko nas?

Materijalni gubici
Poslovni softver, banke, gubici podataka (virusi)...

Love virus, 2000
Desetine miliona zaraženih računara, značajan gubitak podataka, šteta od oko 10 milijardi dolara

Knight Capital Group, 2012
Izgubili su $460 miliona dolara za samo 45 minuta usled greške u softveru kojom je pokrenuta pogrešna verzija softvera.

(Ne)ispravnost softvera

Cena neispravnosti softvera — 2002. godine
Neispravan softver košta američku ekonomiju 59.5 milijardi dolara godišnje.* Ranije otkrivanje grešaka moglo bi da uštedi 22 milijarde dolara godišnje.* **Testing*, US National Institute of Standards and Technology (NIST), 2002.*

(Ne)ispravnost softvera

Cena neispravnosti softvera — sada?
Još veća...

Broj linija koda se duplira svake dve godine
Broj defekata po liniji koda je isti u zadnjih 10 godina.

Na koji sve način neispravan softver utiče na svet oko nas?

Fatalne posledice
Avioni, automobili, vozovi, aparati u zdravstvu, svemirske letilice, nuklearne elektrane

Ariane 5, 1996
Ariane 5, 1996 raspala se 37s posle lansiranja usled greške u konverziji (64bit u 16bit), šteta 370 000 000 $

Koliko koštaju ovakve greške?

Dhahran raketa, 1991
Raketa je pogodila cilj i ubila 28 vojnika: zbog greške u softveru nije ispaljena protivodbrana

Therac 25, 1986
Više ljudi je umrlo kao posledica predoziranja radijacijom usled neispravnosti softvera uređaja za terapijsku radijaciju.

Greška $\overset{proizvodi}{\rightarrow}$ **Nedostatak** $\overset{uzrokuje}{\rightarrow}$ **Pad** $\overset{pravi}{\rightarrow}$ **Incident** $\overset{posledice}{\rightarrow}$ **...**

Therac 25

- Incident — Pacijent je umro
- Pad — Prekoračena je bezbedna doza zračenja
- Nedostatak — U softveru je nedostajala provera opsega frekvencije zračenja koja se primenjuje
- Greška — (1) Programer je zaboravio da stavi proveru opsega frekvencije zračenja, (2) tehničar je uneo vrednost van opsega

Odnos kvalitet-ispravnost

Da li postoji način da se napravi potpuno ispravan softver?
Da li je kvalitetan softver ispravan? Da li je ispravan softver kvalitetan? Koje tehnike za verifikaciju softvera postoje?

2 Tehnike verifikacije softvera

Tehnike verifikacije softvera

Pristupi verifikaciji softvera
Dinamička verifikacija softvera Statička verifikacija softvera

2.1 Dinamička verifikacija softvera

Dinamička verifikacija softvera

Dinamička verifikacija softvera obuhvata ...
... tehnika ispitivanja ispravnosti koda u toku njegovog izvršavanja.

Najčešći vid verifikacije softvera je ...
... **testiranje**. Testiranje se često koristi kao sinonim za verifikaciju softvera. Testiranje se često koristi i kao sinonim za validaciju i verifikaciju softvera.

Dinamička verifikacija softvera

Testiranje je lako...
... samo pustiš program i proveriš da li radi ili ne radi...

Dinamički pristupi verfikaciji softvera

Testiranje
Testiranje je tehnika izvršavanja programa sa namerom da se pronađe što više mogućih defekata ili da se stekne dovoljno poverenja u sistem koji se testira.

Kada se testira i šta se testira?
Testiranje je aktivnost koja je prisutna u svakoj fazi razvoja softvera. Plan testiranja zavisi od primenjene metodologije razvoja softvera i prilagođava se svakom konkretnom projektu.

Kako i koliko testirati?

Šta garantuje bolji kvalitet koda?

- Veći broj test primera?
- Duže vreme trajanja testiranja?
- ... ?

Strategije testiranja
Metod određivanja reprezentativnog skupa podataka nad kojima će se vršiti testiranje:

- Visok potencijal otkrivanja grešaka
- Relativno mala veličina
- Visok stepen poverenja u pouzdanost softvera

Gde pronaći dobre test primere?

Specifikacija programa
Testiranje crne kutije — funkcionalno testiranje, generisanje test primera bez razmatranja interne strukture koda

Kôd programa
Testiranje bele kutije — strukturno testiranje, generisanje test primera na osnovu interne strukture koda, npr jedinični testovi. Kriterijum pokrivenosti koda: broj izvršenih putanja, broj izvršenih instrukcija, broj izvršenih grana...

Specifikacija i kôd programa
Testiranje sive kutije — mešovita strategija

Ručno i automatsko testiranje

Kako ubrzati testiranje?
Kako automatizovati testiranje? Šta se može i treba automatizovati?

Ručno testiranje
Nekada je potrebno ručno pokrenuti i proveriti neki test primer

Automatizacija testiranja
Tehnike automatizacije procesa testiranja — sastavni deo alata za razvoj softvera, alati za kontinuiranu integraciju softvera, alati za testiranje specifičnih vrsta softvera...

Automatizacija generisanja test primera

Kako ubrzati testiranje?
Da li se test primeri mogu automatski generisati?

Automatizacija generisanja test primera
Tehnike automatskog generisanja test primera — olakšavanje generisanja test primera, npr *fuzz testing*.

Dinamička analiza koda

U okviru testiranja, koriste se i alati za ...
... debagovanje i razne vrste profajliranja.

Ali stalno treba imati u vidu...

Edsger Wybe Dijkstra (Tjuringova nagrada 1972)
"Program testing can show the presence of bugs, never their absence"

Testiranje ne može da dokaže ispravnost softvera...
Pravilnim i sistematičnim testiranjem podižemo nivo pouzdanosti i smanjujemo verovatnoću da greške promaknu. **Testiranje se ne radi nasumično, već je važno poznavati metodologiju, procese i principe testiranja.**

Verifikacija softvera

Da li postoji način da se napravi potpuno ispravan softver?
Avion? Automobil? Uređaji u zdravstvu? Svemirske letilice? Nuklearne elektrane?

2.2 Statička verifikacija softvera

Statička verifikacija softvera

Statička verifikacija softvera
Analiza ispravnosti programa bez njegovog izvršavanja — analiza izvornog koda

Vrste statičke verifikacije

- Ručne provere i pregledi koda
- Formalne metode verifikacije softvera — uslovi ispravnosti softvera iskazuju se u terminima matematičkih tvrđenja na striktno definisanom formalnom jeziku izabrane matematičke teorije.

Formalne metode verifikacije softvera

Idealno rešenje...
Alat koji automatski analizira kôd i daje precizne informacije o njegovoj ispravnosti...

Formalne metode verifikacije softvera

Fundamentalno ograničenje
Halting problem je neodlučiv* — Ne postoji opšti automatizovan način za proveravanje da li je neka naredba programa dostižna, pa sami tim ni da li je ispravna, odnosno da li je sam program ispravan.

*Alan Turing, *On Computable Numbers With an Application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society, **1936**.

Formalne metode verifikacije softvera

Idealno rešenje...
... ne postoji!

Posledica teorijskog ograničenja
Nije moguće napraviti program koji bi potpuno automatski, u konačnom vremenu, koristeći konačne resurse, mogao da utvrdi ispravnost proizvoljnog programa *potpuno precizno*.

Posledica teorijskog ograničenja
Moguće je napraviti program koji bi potpuno automatski, u konačnom vremenu, koristeći konačne resurse, mogao da utvrdi ispravnost proizvoljnog programa ~~potpuno precizno~~.

Preciznost

- Lažna upozorenja i propuštene greške
- Kompromis: Preciznost ⇄ Efikasnost

3 Automatske formalne metode verifikacije softvera

Automatske formalne metode statičke verfikacije softvera

Automatske metode

- Simboličko izvršavanje
- Proveravanje modela
- Apstraktna interpretacija

Oslanjaju se na ...

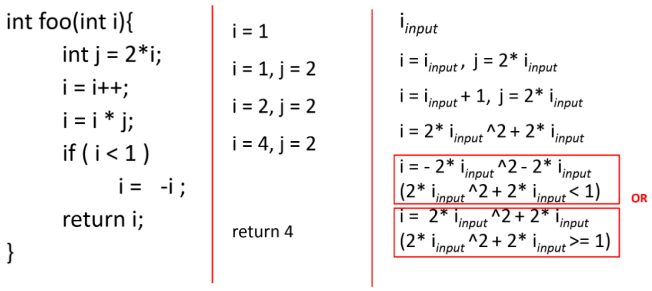
- Semantiku programskih jezika
- Automatsko dokazivanje teorema
- Često se koriste SMT rešavači

3.1 Simboličko izvršavanje

Simboličko izvršavanje

Simboličko izvršavanje
Praćenje simboličkih umesto konkretnih vrednosti promenljivih

Primer

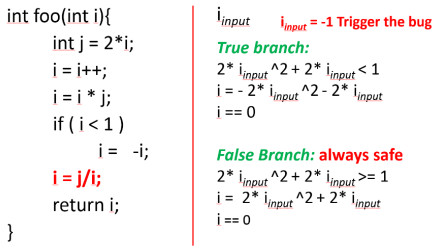


Simboličko izvršavanje

Primene

Generisanje test primera, otkrivanje mrtvog koda, pronalaženje grešaka, generisanje invarijanti koda...

Primer - pronalaženje grešaka



Simboličko izvršavanje

Izazovi

- Ogroman broj puteva — kako ih sve obići?
- Modelovanje naredbi i okruženja
- Rešavanje uslova

Ideje

Osnovne ideje datiraju iz 1976. godine: *Symbolic Execution and Program Testing. James C. King.* Međutim, potrebno je bilo da prođe vreme kako bi se ideje efikasno realizovale

Karakteristike

Precizna ali vremenski i memorijski veoma zahtevna tehnika

Simboličko izvršavanje

Alati

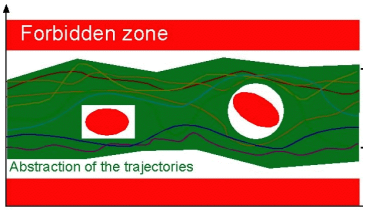
- KLEE — alat otvorenog koda, baziran na LLVM infrastrukturi, pronašao veliki broj problema u softveru otvorenog koda
- SAGE — alat koji se interno koristi u okviru Microsoft-a za pronalaženje grešaka
- PEX — .NET platforma, Microsoft
- jCUTE — Java, Open Systems Laboratory
- Java PathFinder — proveravanje modela sa podrškom za simboličko izvršavanje, NASA Ames Research Center
- Code Hunt — generisanje test primera, Microsoft
- ...

3.2 Apstraktna interpretacija

Apstraktna interpretacija

Apstraktna interpretacija

Teorijski okvir za formalizaciju apstrakcije. Osnovna ideja: konkretna semantika programa je previše kompleksna da bi se o njoj moglo rezonovati. Zbog toga je potrebno apstrahovati konkretnu semantiku programa u nekakav sveobuhvatni nadskup o kome je moguće efikasno rezonovati.



Apstraktna interpretacija

Ideje

Osnovne ideje datiraju iz 1977. godine: *Patrick Cousot, Radhia Cousot: "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints"*

Karakteristike

- Skalira dobro na velikim programima
- Ne propušta greške, ali može imati lažna upozorenja
- Primena u avio-industriji, automobliskoj industriji, svemirske letilice — neki standardi zahtevaju upotrebu statičke analize i apstaktne interpretacije.

Apstraktna interpretacija

Alati

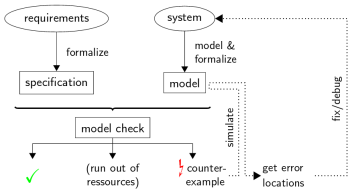
- Astrée — AbsInt
- Polyspace Bug Finder — MathWorks
- Coverity — Synopsys
- CPAchecker — Free software, Apache 2.0 License
- Frama-C value analysis — Open source software

3.3 Proveravanje modela

Proveravanje modela

Proveravanje modela
Sistematski prolazak kroz sva moguća stanja sistema Metodi matematičke logike Automatsko traženje grešaka, sa mogućnošću generisanja kontraprimera

Osnovni algoritam



Proveravanje modela

Ideje
Osnovne ideje datiraju iz 1980-tih godina, E. M. Clarke, E. A. Emerson, J. Sifakis (Tjuringova nagrada 2007).

Karakteristike
Najpre verifikacija hardvera, komunikacionih protokola i konkurentnih sistema Verifikacija softvera Precizna tehnika, ali ne skalira dobro na velikim sistemima.

Proveravanje modela

Alati

- Velike kompanije imaju svoje interne alate za proveravanje modela
- SLAM — Microsoft
- CBMC — Carnegie Mellon University
- SatAbs — University of Oxford
- LLBMC, LAV — Proveravanje modela na LLVM IR
- DSVerifier, ESBMC — Federal University of Amazonas
- BLAST — Berkeley
- Java Pathfinder — NASA Ames Research Center
- CPAchecker — Free software, Apache 2.0 License

4 Formalne metode

Formalne metode verifikacije softvera

Posledica teorijskog ograničenja
Nije moguće napraviti program koji bi potpuno automatski, u konačnom vremenu, koristeći konačne resurse, mogao da utvrdi ispravnost proizvoljnog programa *potpuno precizno*.

Posledica teorijskog ograničenja
Moguće je napraviti program koji bi interaktivno, u konačnom vremenu, koristeći konačne resurse, mogao da utvrdi ispravnost nekih programa potpuno precizno.

http://www.leptonica.com/cachedpages/perlis-epigrams.html

Alan Džej Perlis (engl. Alan Jay Perlis, 1922–1990)...

... poznat po svom pionirskom bavljenju razvojem programskih jezika, prvi dobitnik Tjuringove nagrade.

"**It is easier to write an incorrect program than understand a correct one.**"

Fromalne metode verifikacije softvera

Tehnike razvoja ispravnog softvera
Razvoj iz specifikacije, formalno dokazivanje ispravnosti softvera koji se razvija, najviši nivo sigurnosti

Najskuplji razvoj softvera
Zahteva visoko stručne ljude Zahteva upotrebu posebnih alata, npr Coq ili Isabelle Najsporiji ali najpouzdaniji razvoj Koristi se u industriji za razvoj kritičnih delova koda, sastavni deo različitih standarda

5 Umesto zaključka...

Umesto zaključka...

Uvod u oblast

- Uticaj neispravnog softvera: neprijatnosti, materijalni troškovi, materijalno nemerljive posledice...
- Verifikacija softvera: skup metoda, alata i procesa za utvrđivanje ispravnosti softvera
- Osnovni pristup: testiranje, ali testiranje ne može da garantuje ispravnost softvera

Umesto zaključka...

Uvod u oblast

- Automatske statička verifikacija: neophodni su kompromisi između preciznosti i efikasnosti
- Formalne metode: za razvoj kritičnih delova aplikacija
- Trendovi: razvoj algoritama i integracija alata automatske statičke verifikacije softvera, primena ovih tehnika u sintezi koda i bioinformatici, poboljšavanje i ubrzavanje metodama mašinskog učenja