Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave

PROGRAMSKO INŽENJERSTVO

ak. god. 2021./2022.

Uvod u programsko inženjerstvo

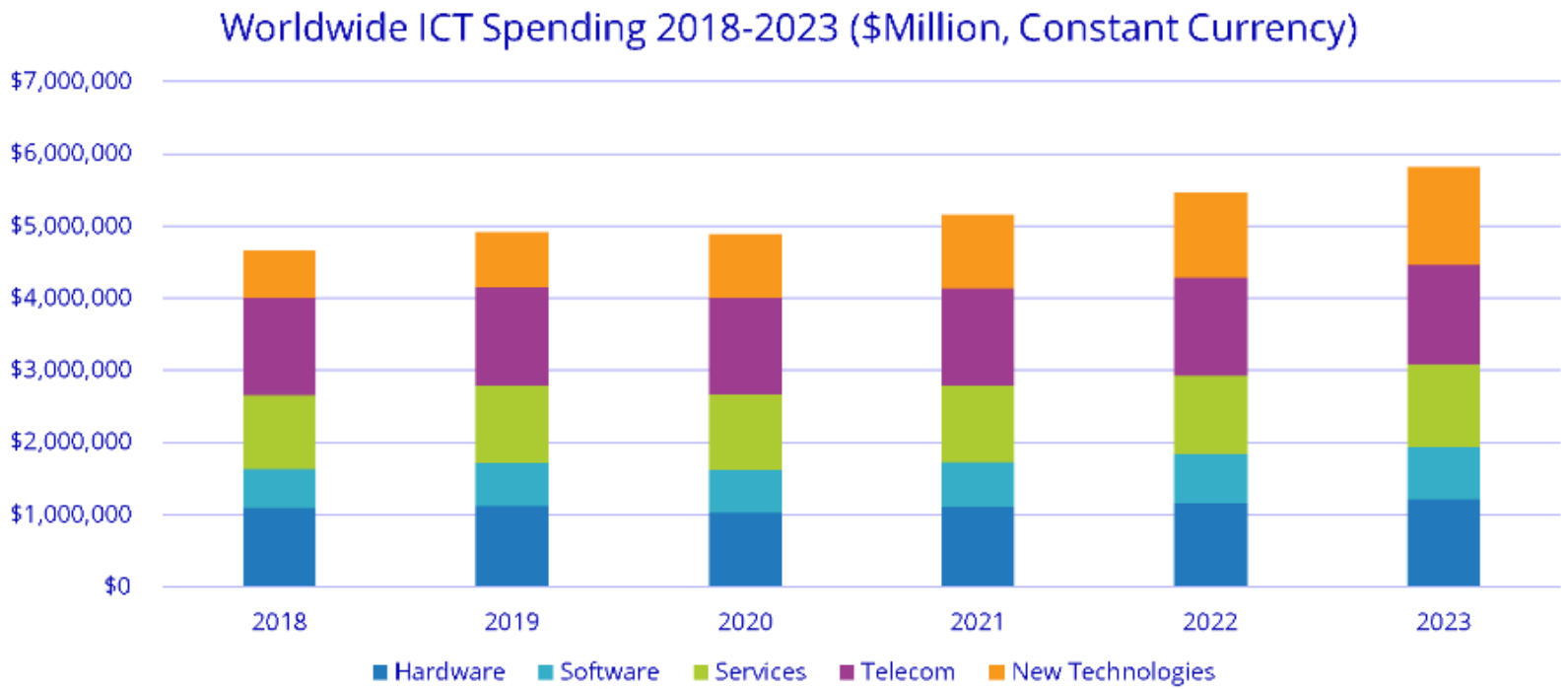
Pregled tema

- O predmetu
- Administracija
  - organizacija nastave
  - sadržaj kolegija
  - literatura
  - ocjenjivanje
- Razvoj programske potpore
- Programsko inženjerstvo

Računarstvo

- *“Studijski program Računarstvo omogućava stjecanje kompetencija za analizu i rješavanje srednje složenih inženjerskih problema, za rad u timu te za doprinos oblikovanju sustava i procesa s područja računarstva, uz korištenje temeljnih znanja iz matematike, fizike, elektrotehnike i računarstva te suvremenih računarskih alata.”*
- *obuhvaća teoriju, metode analize i sinteze, projektiranje i konstrukciju, primjenu i djelovanje računalskih sustava.*

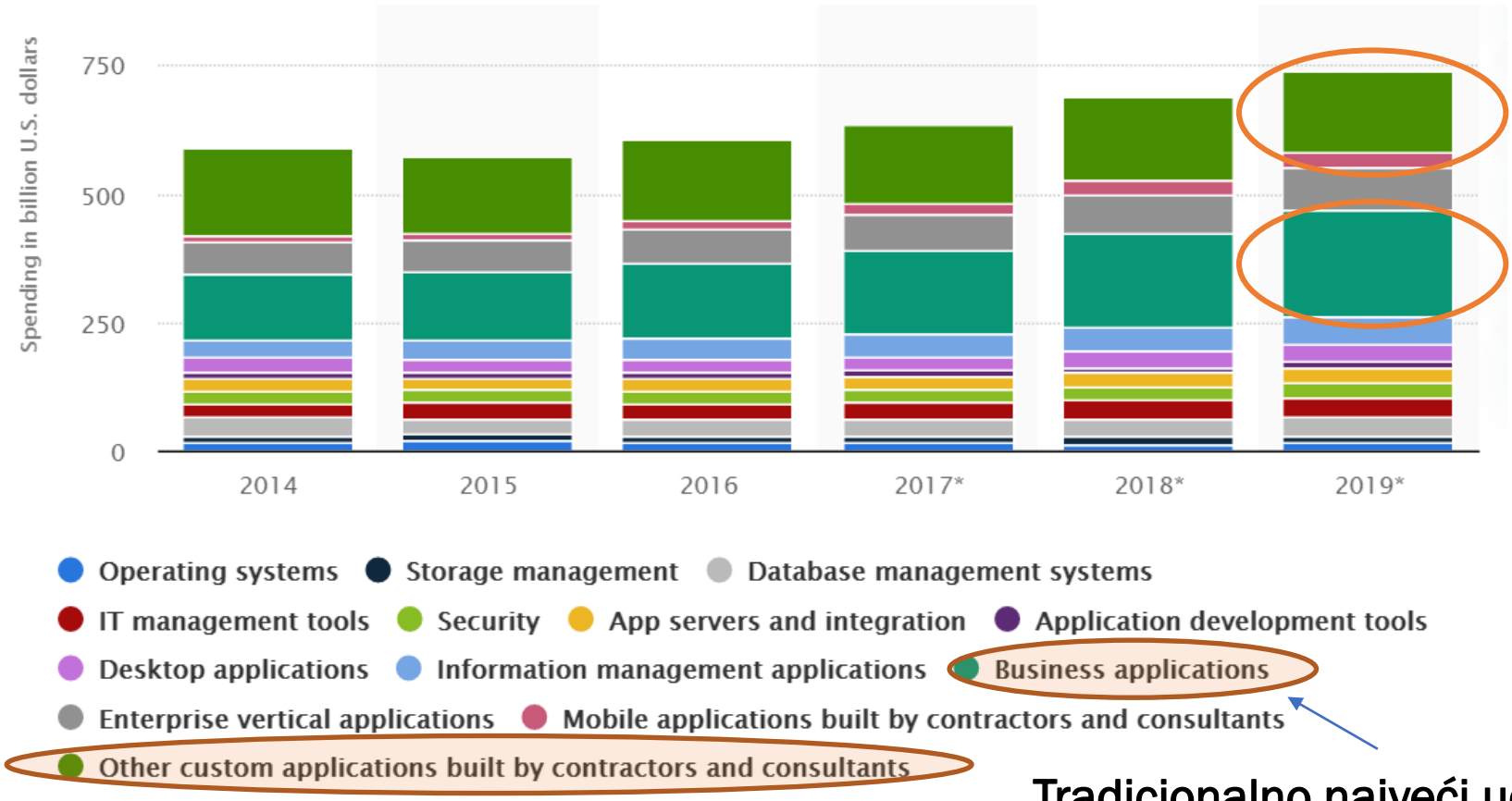
Trend informacijsko komunikacijskih tehnologija



Stabilan rast ~ 10%

Izvor: IDC - Global ICT Spending Forecast 2020 – 2023

Vrste programa

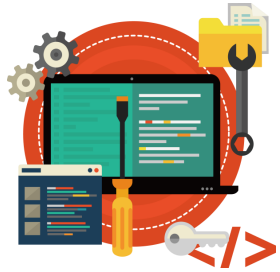


Tradicionalno najveći udio

Izvor: Statista, Global spending on IT software products

O predmetu

- Predmet *Programsko inženjerstvo* daje osnovna znanja i vještine nužne za postizanje kompetencija u programskom inženjerstvu, a posebice razumijevanje, oblikovanje i vrednovanje programskih sustava.
  - generički modeli procesa programskog inženjerstva;
  - inženjerstvo analize zahtjeva;
  - koncepti programskih arhitektura i paradigme specifikacije;
  - jezici za modeliranje objektno usmjerenih sustava;
  - modeliranje ulazno/izlaznih i reaktivnih programskih sustava;
  - ispitivanje programskih sustava;
  - formalna specifikacija i verifikacija željenih obilježja programskih sustava;
  - automatizirani pomoćni postupci i alati u oblikovanju programskih sustava.



ADMINISTRACIJA

Programsko inženjerstvo

- engl. *Software Engineering*
- Studij : Računarstvo

- Semestar: 5
- ECTS : 5

- Nositelji i izvođači:
  - Vlado Sruk – nositelj grupa PR1
  - Alan Jović – nositelj grupa PR2
  - Nikolina Frid – izvođač grupa PR3

Vježbe, konzultacije

- Asistenti:
  - Nikolina Frid
  - Hrvoje Nuić
  - Igor Stančin
  - Eugen Vušak
  - Daria Primorac
  - Vanjski suradnici
- Informacije:
  - Sve obavijesti u vezi predmeta bit će objavljene na web stranici:

    https://www.fer.unizg.hr/predmet/proinz
  - Predavanja: Moodle

    https://moodle.fer.hr/course/view.php?id=527
  - **Svi upiti na: progi@fer.hr**

Predavanja i konzultacije

- Predavanja
  - prema satnici u studentskim kalendarima
- Konzultacije
  - Pauze predavanja
  - Uz najavu e-poštom:
    - Predavaču e-poštom: Naslov/Subject: [PROGI] Konzultacije
    - Online (MS Teams)

Organizacija predmeta

- **Predavanja** u dva ciklusa (7+6 tjedana)
  - temeljne cjeline:
    - uvod i motivacija, procesi programskog inženjerstva
    - arhitekture programske potpore (modeli i izbor, UML ..)
    - validacija, verifikacija i ispitivanje programske potpore
- **Samostalni rad:** Projekt
  - 2 kontrolne točke

Preporučena literatura

- *Bilješke s predavanja*
- Nastavni sadržaji prezentacija s predavanja
- Knjige:
  - Sommerville, I., *Software engineering*, Addison Wesley, 2020.
  - R. Leach, *Introduction to software engineering*, 2nd ed. Boca Raton: CRC Press, 2016.
  - B. Unhelkar, Software Engineering With UML.: Auerbach Publications, 2020.
  - F. Zammetti, *Modern Full-Stack Development : Using TypeScript, React, Node.js, Webpack, and Docker*, 1st ed. Pottstown: Apress, 2020.
  - *Guide to the Software Engineering Body of Knowledge: Version 3.0 (SWEBOK Guide) version 3,* IEEE Computer Society, 2014.
- Dodatni sadržaji
  - A. Jović, N. Frid, D. Ivošević : "Procesi programskog inženjerstva"
  - A. Jović, M. Horvat, I. Grudenić: "UML dijagrami – Zbirka zadataka i riješenih primjera" (sveučilišni priručnik)
  - članci, tehnička dokumentacija (FER web, Moodle, web)

Dodatna preporučena literatura

- Maršić,I., *Software engineering*, Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/
- Pressman R.S.: *Software Engineering – A Practitioner's Approach*, McGraw-Hill, 2009
- Lethbridge, T. C., Laganiere, R., *Object-Oriented Software Engineering*, 2nd ed., McGraw-Hill, 2005.
- J. Dooley, *Software development, design and coding*, 2nd ed. Apress. 2017
- E. Freeman, *Head First Design Patterns : Building Extensible and Maintainable Object-Oriented Software*. O'Reilly Media, Incorporated, 2020.
- Gamma E. et al: *Design patterns: elements of reusable object-oriented software*, Addison-Wesley, 1995
- Rambaugh J., Jacobson I. and Booch G.: *The Unified Modeling Language Reference Manual,* Addison-Wesley, 2005
- Fowler M.: *UML Distilled*, Addison-Wesley, 3rd edition, 2004

Kontinuirana nastava

Kratke provjere	max. 9
Projekt	max. 30
Međuispit	max. 25
Završni ispit	max. 36

- Uvjet za izlazak na završni ispit
  - ≥ *15* bodova iz projekta
- Minimalni uvjeti za polaganje u kontinuiranoj nastavi
  - na završnom ispitu postignuto ≥ *12 bodova*
  - prag za prolaz: *50 bodova*

Ispitni rokovi

Projekt	max. 30
Pismeni ispit	max. 70

prenose se bodovi projekta iz kontinuirane nastave !

- Uvjet za izlazak na ispitni rok
  - *≥ 15/30* bodova iz projekta

- Polaganje predmeta
  - na pismenom ispitu postignuto *≥ 35 bodova*
  - prag za prolaz: *50 bodova*

Bodovni pragovi i ocjene

- Jednaki pragovi za kontinuiranu nastavu i ispitne rokove

Ocjena	Bodovi
Izvrstan (5)	≥86
Vrlo dobar (4)	≥72
Dobar (3)	≥60
Dovoljan (2)	≥50

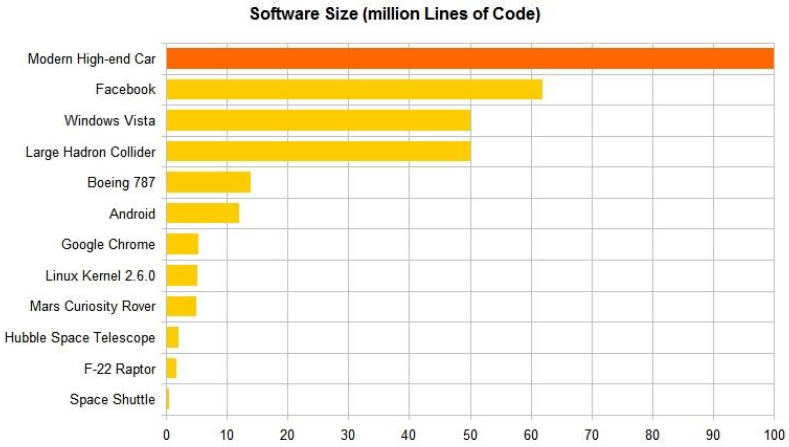
RAZVOJ PROGRAMSKE POTPORE

Razvoj programa

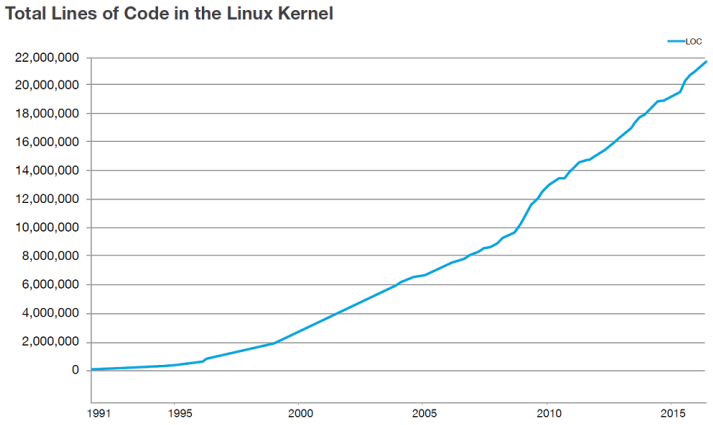
- 1965. Gordon Moore
  - broj tranzistora po in^2 udvostručuje se svake dvije godine
  - kasnija revizija: udvostručenje svakih godinu i pol
  - danas: Intelove prognoze govore o udvostručenju svakih 2.5 godine
- Programi – glavna industrija današnjice
  - primjena u poslovnom, znanstveno-istraživačkom, društvenom, ... životu
  - IT-sustavi, prikupljanje, obrada, pohranjivanje, dohvaćanje, upravljanje informacijama
  - sve razvijene ekonomije ovisne o programskoj industriji
- Svojstva programa
  - *veličina*: broj linija koda - KLOC
  - *funkcionalnost*: funkcionalnost koja je na raspolaganju krajnjem korisniku
  - *složenost*: složenost problema, algoritamska složenost, strukturna složenost, spoznajna složenost

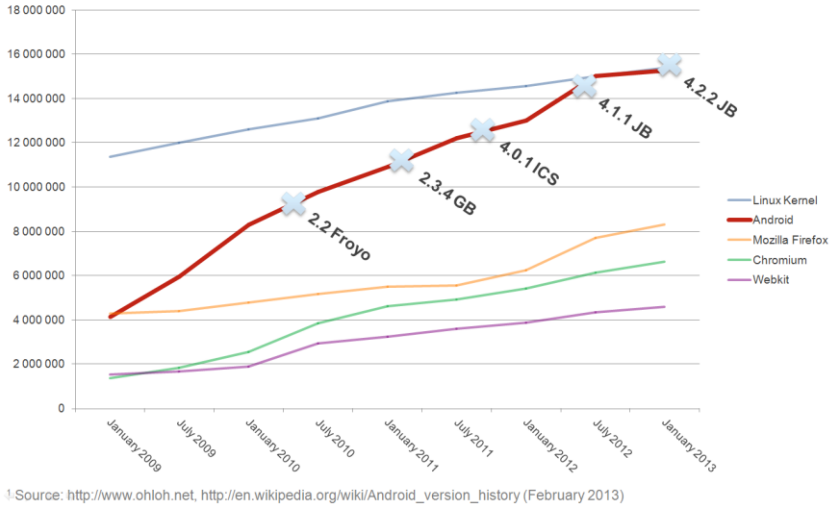
Primjer složenosti programa

- Konstantan rast
  - Nova područja
  - Novije inačice



Izvor: informationisbeautiful.net/visualizations/million-lines-of-code/





Izvor: D.Rosen: Evolution of the Android Open Source Project

Povijest

- 1968 – priznavanje problema realizacije složenih programskih projekata – kriza razvoja programske potpore
  - engl. *the software crisis*
- pokušaj rješavanja problema razvoja kvalitetnih programskih projekata na vrijeme i uz zadana sredstva
- 1968 - NATO Conference in Garmisch Partenkirchen
  - termin programsko inženjerstvo engl. *software engineering*
  - Primjena inženjerskih principa u području programiranja
- Programski sustavi
  - veliki, složeni
  - neophodan timski rad
  - postoje u mnogim verzijama
  - traju dulje vrijeme
  - podložni promjenama

Problemi razvoja programa

- *"The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."*
  - – 1972 Edsger Dijkstra, The Humble Programmer
- 1986. *F. Brookes, No Silver Bullet: Essence and Accidents of Software Engineering, Information Processing*
- Inherentni problemi razvoja programske potpore uzrokovani prirodom samih programa
  - suštinski, nezaobilazni problemi
  - neočekivani i nenamjerni problemi
    - možemo ih riješiti

Suštinski problem razvoja

- Svojstva programa
  - složenost (engl. *complexity*)
  - usklađenost (engl. *conformity*)
  - promjenljivost (engl. *changeability*)
  - nevidljivost. (engl. *invisibility*)
    - kako ga vizualizirati?
    - Kompletni prikazi mogu biti nerazumljivi, a djelomičan zavaravajući

Jedinstvenost programa i razvoja

- Nema fizička svojstva;
- Proizvod intelektualnog timskog rada;
- Produktivnost programera znatno odstupa od produktivnosti ostalih inženjerskih disciplina;
- Procjenu i planiranje projekata programske potpore karakterizira visok stupanj nesigurnosti, a u najboljem slučaju može djelomično ublažiti uporabom najbolje prakse;
- Upravljanje rizicima za projekte programske potpore pretežno je orijentirano na procese;
- Česte promjene.

Karakteristike programske potpore

- Programska potpore je neopipljiva
  - teško je razumjeti napore procesa oblikovanja
- Jednostavna za reproducirati
  - bitno različita od svih drugih artefakata (do sada)
- Industrija programske potpore je izrazito radno intenzivna
  - teško je automatizirati u ključnim segmentima
- Lagano je unijeti izmjene
  - često i bez dubljeg razumijevanja posljedica
- Programska potpora se ne troši
  - ali se kvari ako se ne koristi kako je zamišljena ili ako postoje ugrađene pogreške

Uspješnost projekata

- Različite mjere
- Osnovni atributi uspjeha prema *The Standish Group International*:
  - *Pravovremeno – engl. OnTime*
  - *U proračunu - engl. OnBudget*
  - *Prema cilju - engl. OnTarget*
  - *Poslovni cilj – engl. OnGoal*
  - *Vrijednost – engl. Value*
  - *Zadovoljstvo – engl. Satisfaction*

✓ **On time, Not on time**
✓ **On budget, Not on budget**
✓ **On Target, Not on target**
✓ **Precise, Close, Loose, Vague, Distant**
✓ ***Very high, High, Average, Low, Very Low***
✓ **Very satisfied, satisfied, somewhat ,not satisfied, disappointed**

- *Ontime*, OnBudget i OnTarget
  - projekt završen u razumnom procijenjenom roku i ostao unutar proračuna te sadržava dobar broj ciljanih značajki i funkcija.
- OnTime, OnBudget, Satisfaction
  - projekt završen u razumnom procijenjenom roku i ostao u proračunu te je zadovoljio kupaca i korisnike bez obzira na izvorni opseg zahtjeva.
- 1994. *godine 16% projekata uspješno*
- 2000. *godine 28% projekta uspješno*

Statistika uspješnosti projekata

Uspješan: na vrijeme, u okviru proračuna, potpuno prema specifikaciji
Izazovan: završen, izvan proračuna, kasni, manje funkcionalnosti
Neuspješan: prekinut prije kraja

	2011	2012	2013	2014	2015
SUCCESSFUL	39%	37%	41%	36%	36%
CHALLENGED	39%	46%	40%	47%	45%
FAILED	22%	17%	19%	17%	19%

The Traditional resolution of all software projects from FY2011–2015 within the new CHAOS database.

The total number of software projects is 25,000-plus, with an average of 5,000 per yearly period.

Software “HALL OF SHAME”

- 1993: London Stock Exchange ($ 600 M, canceled)
- 1995: American Airlines, (159 death)
- 1996: Arianespace (Ariane 5 rocket explosion, $350 M)
- 1997: Korean Jet crash, (225 death)
- 1999: State of Mississippi (Tax system, $185 M loss)
- 2000: National Cancer Institute (pogrešan izračun doze-8 mrtvih/20 ozlijeđenih)
- 2001: Panama radiation overdose, (5 death)
- 2001: Nike Inc. (Supply chain management, $100 M loss)
- 2002: McDonald’s (Purchasing system canceled, $170 M)
- 2003: North-eastern U.S. Power loss (3 death)
- 2004: Hewlett-Packard (Management system, $160 M loss)
- 2004: Sainsbury food chain, UK ($527 M, canceled)
- 2004: Ford Motor Co. (purchasing, $400 M, canceled)
- 2004: Mars Spirit, NASA: "a serious software anomaly“
- 2005: UK Tax (soft errors resulted in $3.5 G (billion) overpay)
- 2005: FBI Trilogy ($105 M, canceled)
- 2009: SQL injection errors - steal data on approximately 130 million credit and debit cards over several months
- 2010: German Bank Cards (30 M korisnika,...)
- 2010: Alarm Clock Bug in Mobile phones
- 2012: Royal Bank of Scotland (RBS) – transferi između računa
- 2014: UK Border Agency and the immigration system – sustav za imigrante ne funkcionira
- 2017: Cloudflare – problem sigurnosti
- 2018: NHS-IT problem pristupa podacima pacijenata
- 2018: O2 mobilna podatkovna mreža – 30 mil. korisnika
- 2019: Facebok Instagram – problem postavljanja slika
- 2020: Heatrow airport Check-in – otkazani letovi
- 2020: *Uzrok?*

Problem pisanja ispravnih programa

- SLOŽENOST (engl. *Complexity*)

- Nivo složenosti modernih programa dosegao je razinu gdje se može mjeriti s biološkim sustavima!
  - sustavi sustava s više desetaka milijuna linija koda

- Klasifikacija složenosti:
  - osnovna složenost
    - ovisi o problemu
    - ne može se eliminirati tehnologijom ili tehnikama
  - nenamjerna (engl. *Accidental complexity*)
    - uvedena uporabom alata ili tehnika
- Problem razvoja programske potpore je prekomjerna nenamjerna složenost

Proces izrade programske potpore

- Pretpostavke:
  - dobar proces -> dobra programska potpora
  - dobar proces -> manji rizik neuspjeha
- Upravljanje rizikom (*engl. Risk Management*)
  - koji su problemi izrade programske potpore?
  - kako smanjiti rizik?

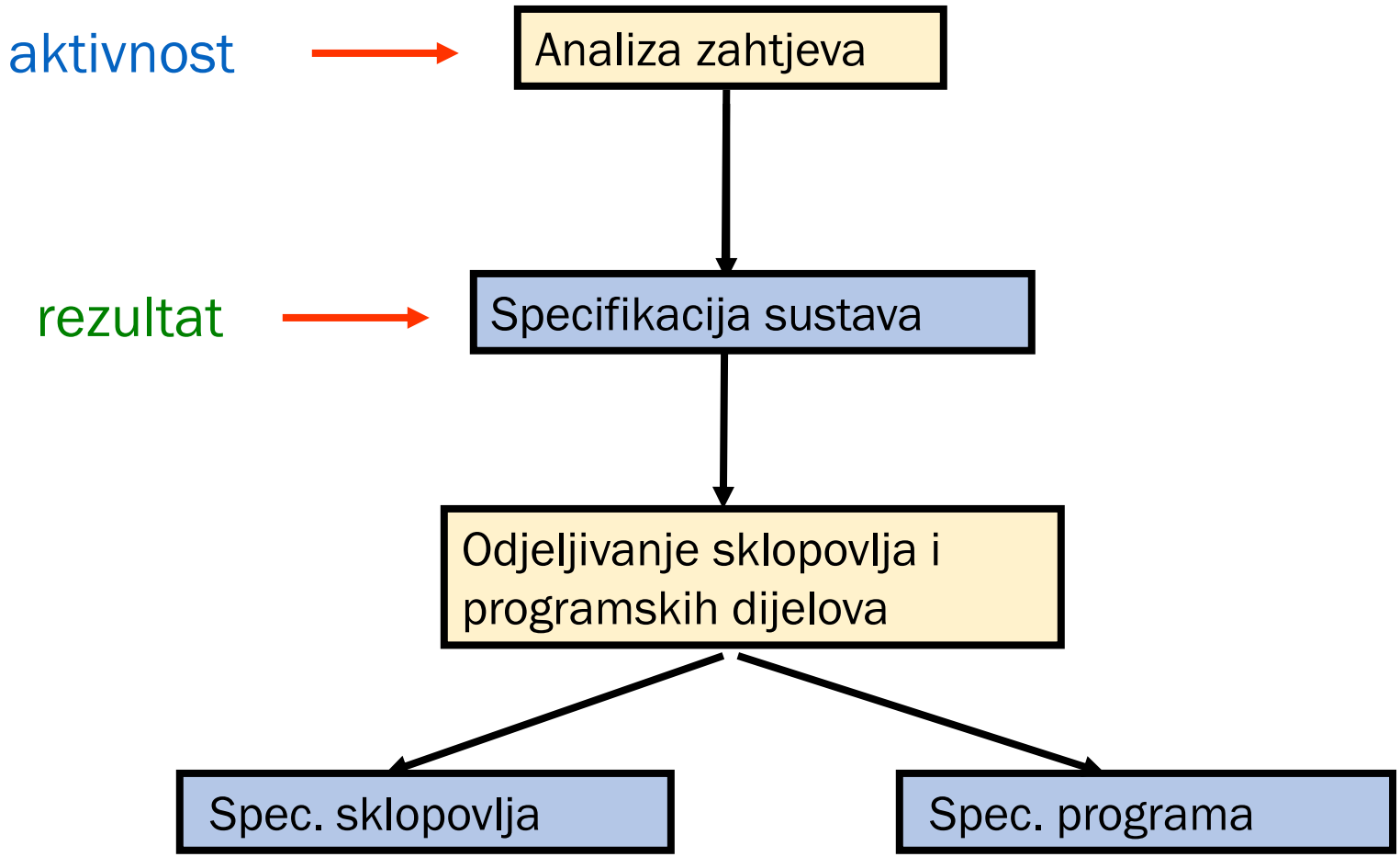
Uloga programskog inženjera

- Samo vještina programiranja nije dovoljna
- Programski inženjer
  - razumije zahtjeve i piše specifikacije
    - izgrađuje modele
  - izvrsno poznavanje programiranja
  - upotrebljava znanstvene metode i interpretaciju pri analizi i rješavanju inženjerskih problema.
  - član tima
    - vještine komunikacije
    - vještine upravljanja

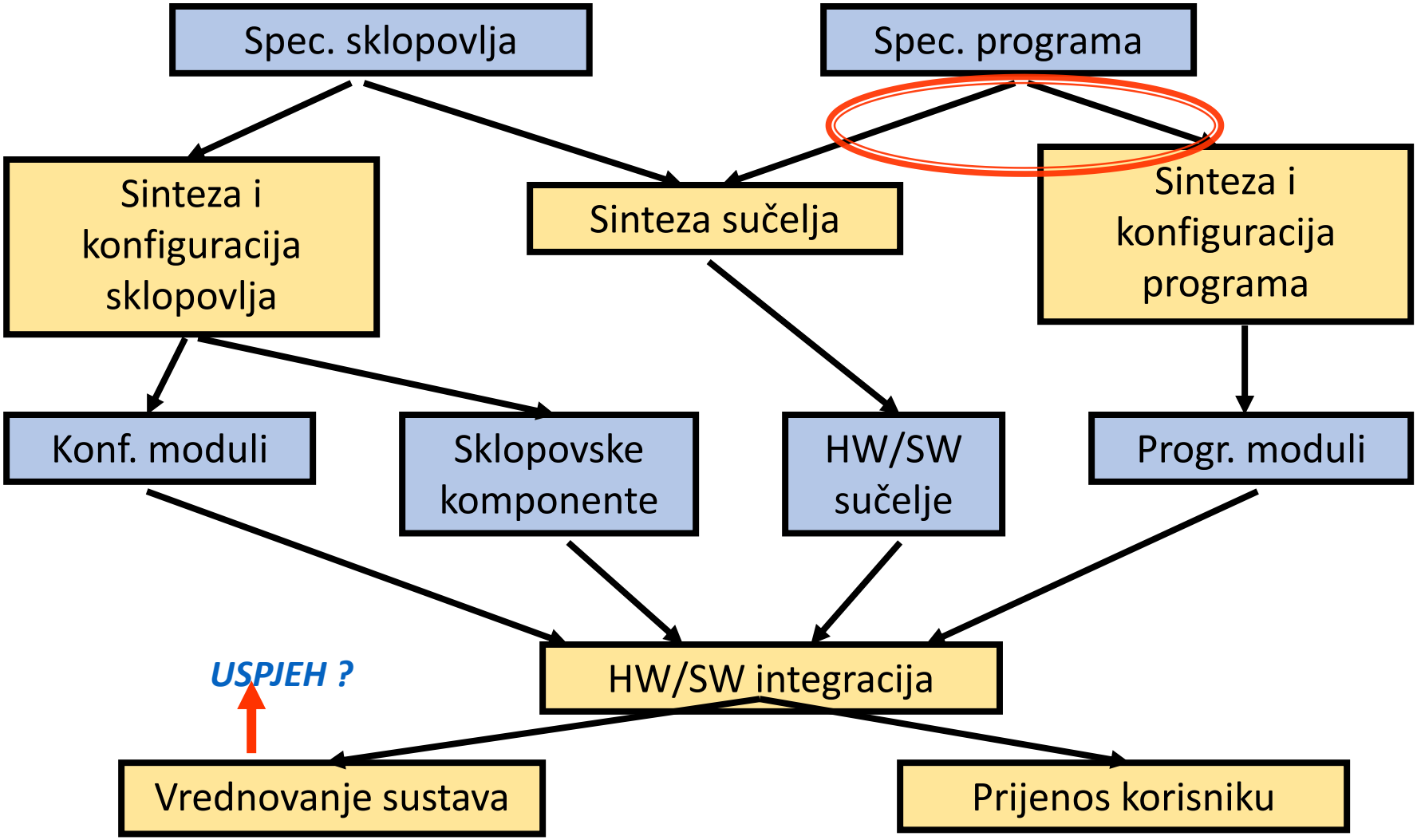
Tradicijsko oblikovanje programske potpore

- Formulacija zahtjeva (*engl. requirements*)
  - govori što bi sustav trebao raditi.
- Specifikacija i analiza.
- Oblikovanje
  - kako će sustav ispuniti ciljeve
- Kodiranje
  - stvarno programiranje
- Ispitivanje modula.
- Integracija i ispitivanje sustava.

Tradicijski postupak oblikovanja sustava (1)



Tradicijski postupak oblikovanja sustava (2)



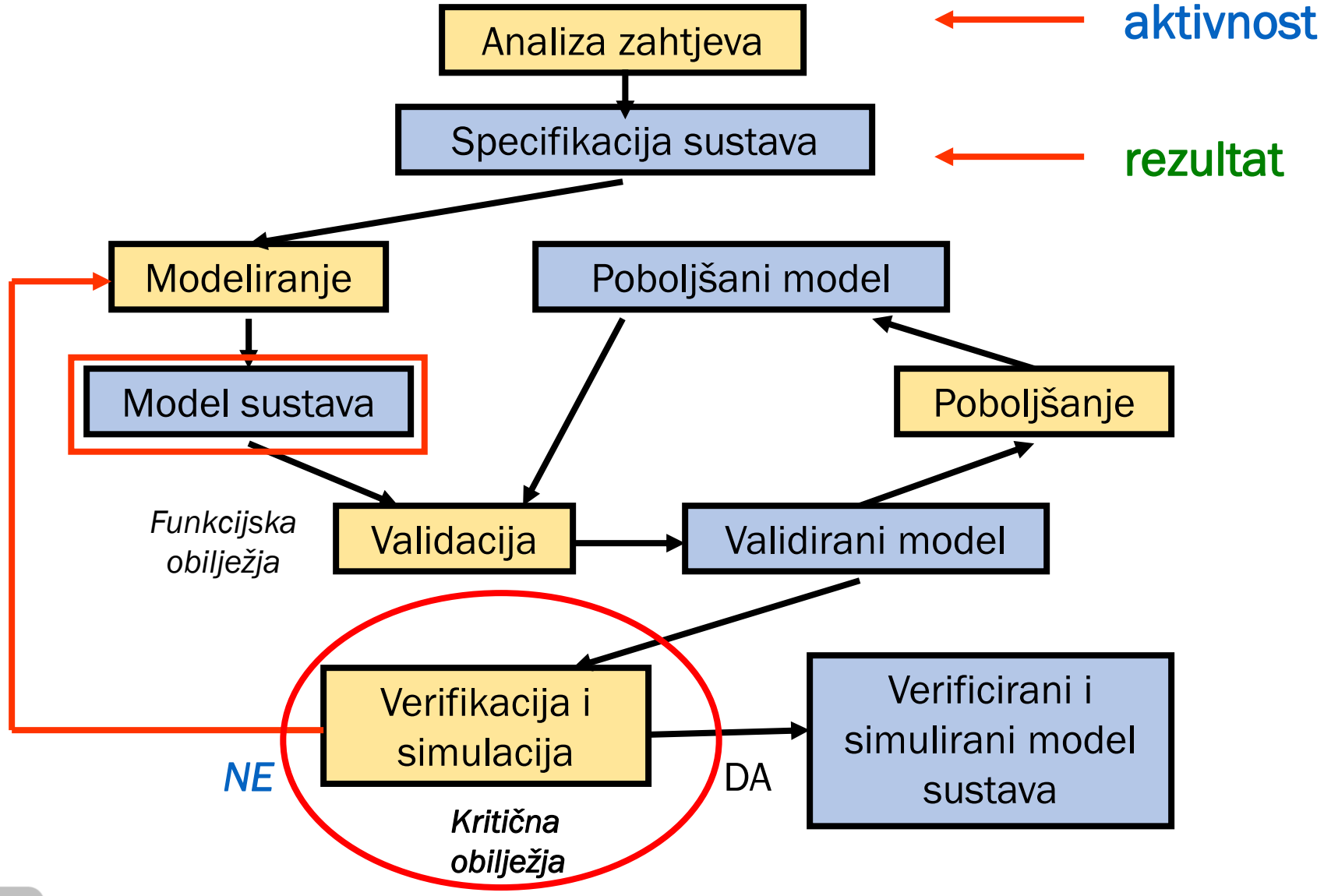
Nedostaci postupka oblikovanja sustava

- Analiza i oblikovanje:
  - zahtjevi nisu formalizirani
    - “Formalni” = temeljeni na matematičkim teorijama (logika, teorija automata, teorija grafova, ...).
    - nemoguće postići preciznost
    - neodređenost dovodi do nekompatibilnosti
  - izgradnja “ad hoc”prototipa
    - neformalni prototipovi ne omogućuju dublju analizu
  - procjena performansi
    - neformalni pristup daje pogrešne vrijednosti
  - dijeljenje na sklopovski i programski dio
    - nije poduprto formalnim transformacijama i postupcima donošenja odluka
  - dokumentacija se teško interpretira

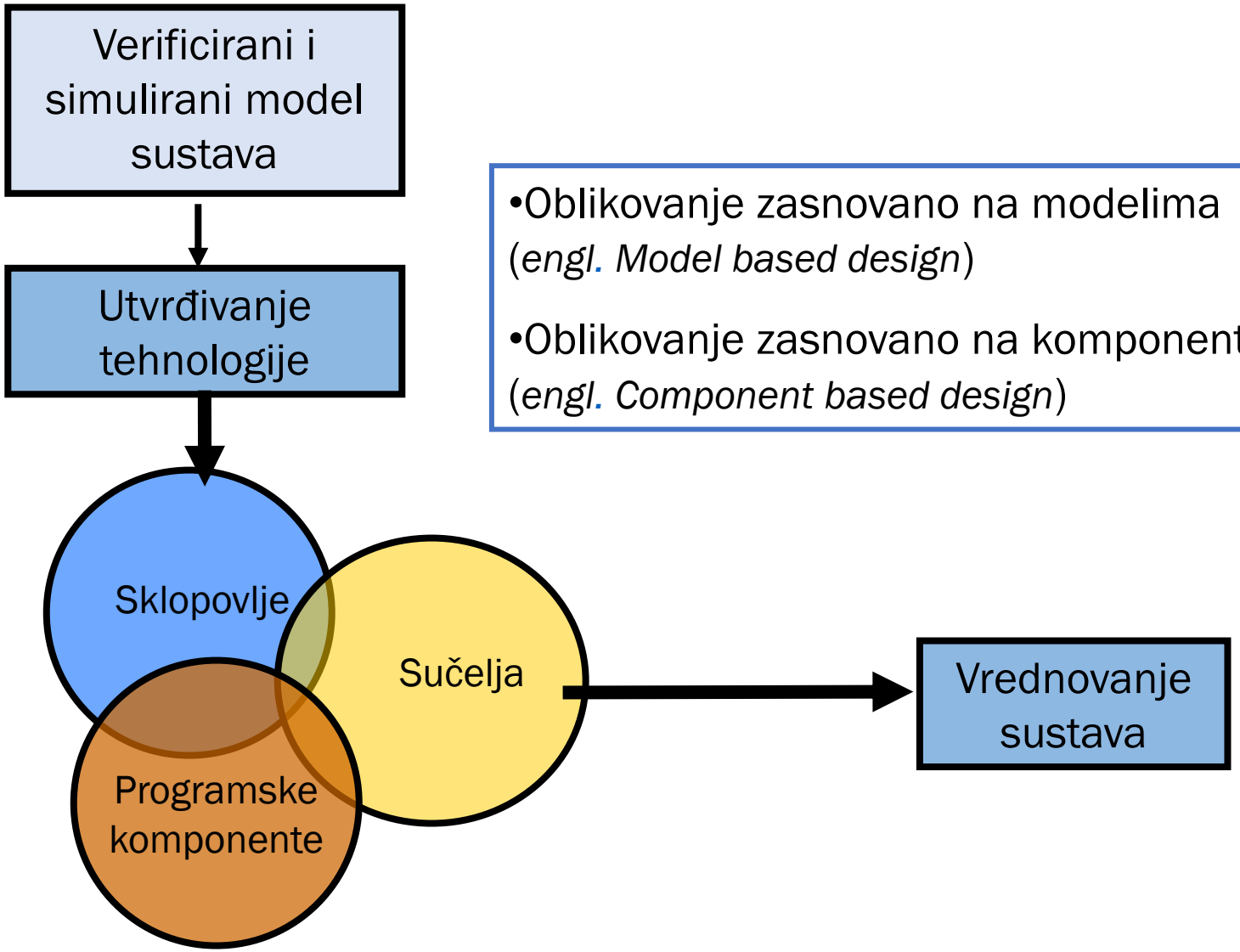
Analiza postojećeg stanja

- Temeljem analize nedostataka tradicijskog postupka oblikovanja sustava i izvedenih zaključaka slijedi predloženi moderan postupak.
- Elementi postupka:
  1. propisani strukturirani postupci
  2. dokumentiranje
  3. modeliranje (apstrakciju)
  4. višestruku uporabu komponenata
  5. formalna verifikacija modela + ispitivanje

Moderan postupak oblikovanja sustava (1)



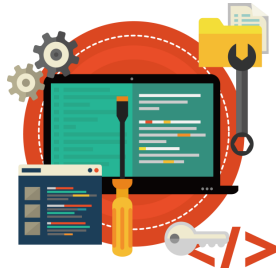
Moderan postupak oblikovanja sustava (2)



- Oblikovanje zasnovano na modelima (*engl. Model based design*)
- Oblikovanje zasnovano na komponentama (*engl. Component based design*)

Sažetak

- Programsko inženjerstvo
  - upravljanje složenosti na raznim nivoima
    - mikro nivo
      - pogreške uzrokovane teškom razumljivošću npr. protokola
    - veliki programi
      - problem interakcija komponenti
      - složeni društveno-tehnički sustavi
      - problem predvidjeti utjecaj svih funkcionalnosti
- Najčešći uzrok zatajenja velikih sustava:
  - pogrešna specifikacija
  - promjena zahtjeva
  - suprotstavljeni zahtjevi
- Veliki programski sustavi su složeni
  - koja je razlika u odnosu na tradicionalne sustave?
  - zahtjev prilagodljivosti i čestih promjena (podrazumijeva se?!)
    - dovodi do akumuliranja funkcionalnosti
  - otežana preglednost strukture
  - problem složenosti ispitivanja



Oblikovanje programske potpore

Principi savladavanja problema složenosti

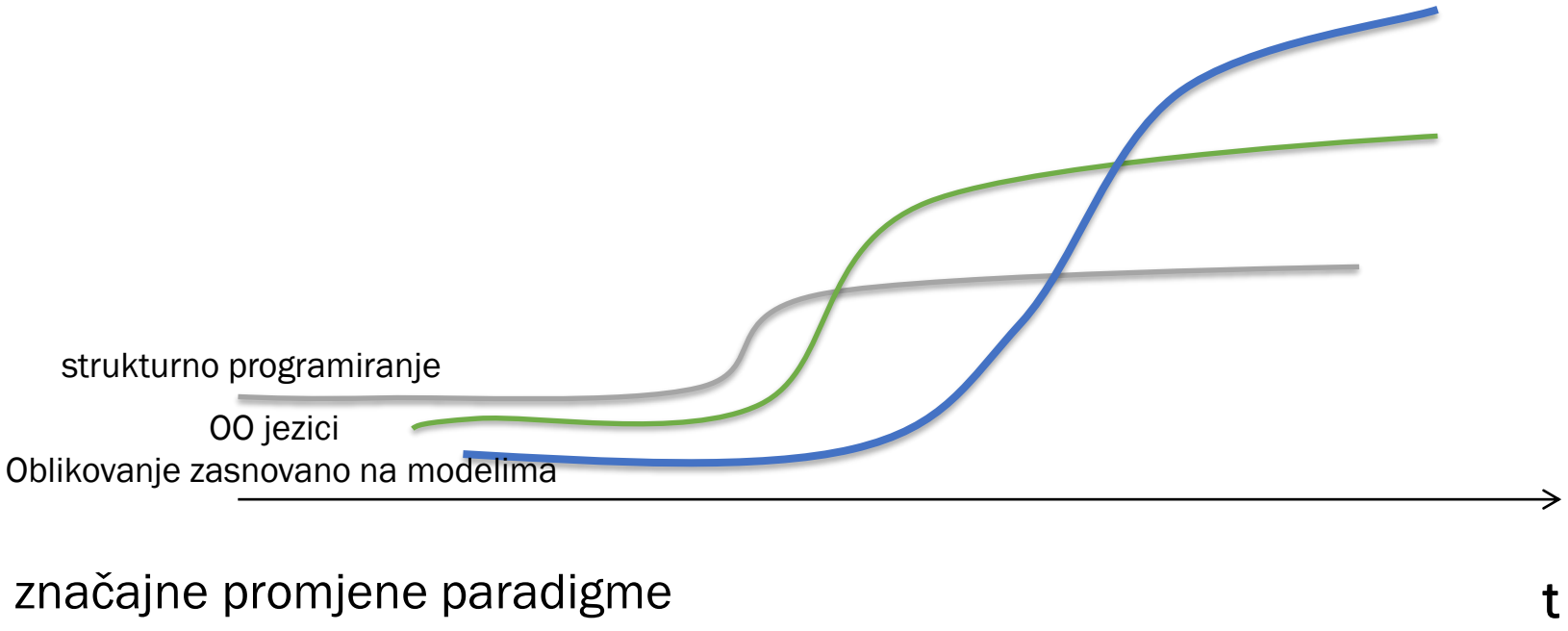
- Razdor problem – implementacija

- Dekompozicija - “podijeli pa vladaj”
- Inkrementalno poboljšanje
- Ponovno korištenje dijelova (engl. *design reuse*)
- Odvojiti podprobleme
- Apstrakcija - eliminirati nepotrebne detalje
  - uvođenje modela
- Uvođenje formalizama - matematički određena semantika!

Kada se javlja?

Tehnološki val

- engl. *Technology waves*
  - široka primjena novih koncepata tehnologije



  - značajne promjene paradigme

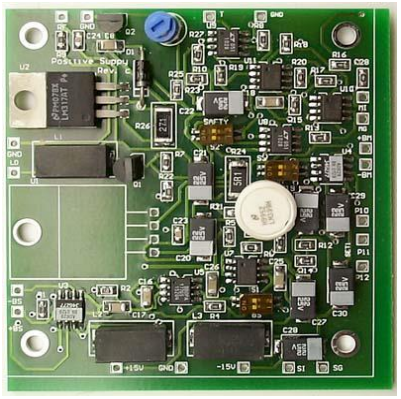
Metodologije oblikovanja zasnovane na modelima

- engl. *Model-driven Development Methods*
- Modeliranje ima dugu tradiciju u programiranju
- Akronimi metodologija oblikovanja zasnovanih na modelima
  - MDSD - Model-Driven Software Development
  - MBASE – Model-Based (System) Architecting and Software Engineering
    - Product, Process, Property, and Success models
  - MBE – Model-Based Engineering
  - MDE – Model-Driven Engineering
  - MDD – Model-Driven Development
  - MDA – Model Driven Architecture
  - DDD – Domain Driven Design
  - ….

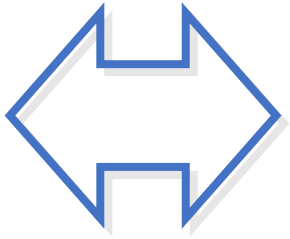
Koji model/e poznajete?

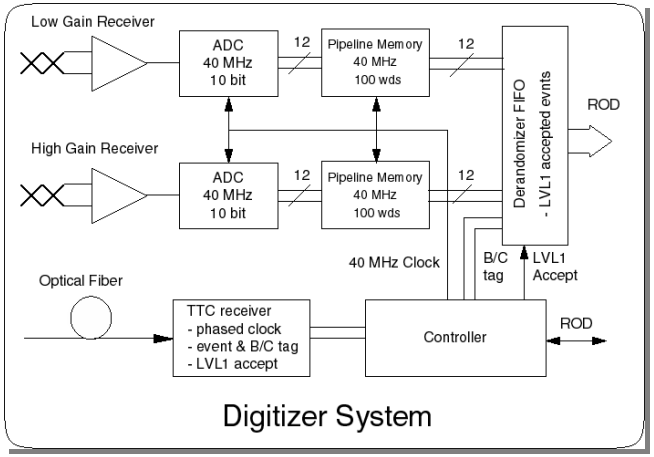
Model

- Inženjerski model: Sažeta reprezentacija sustava koja naglašava značajna svojstva iz nekog pogleda na sustav



modelirani sustav





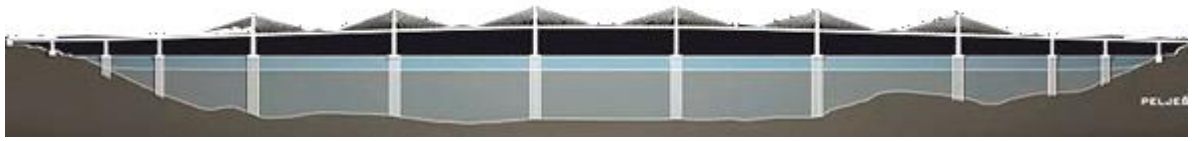
funkcionalni model

Izvor: Bran Selic: "Model-Driven Development", IBM Software Group

- Ne prikazuje sve odjednom
- Upotrebljava oblik prezentacije jednostavno razumljiv u području primjene

Zašto upotrebljavamo modele?

Primjer modela



Što nedostaje?

Problem?

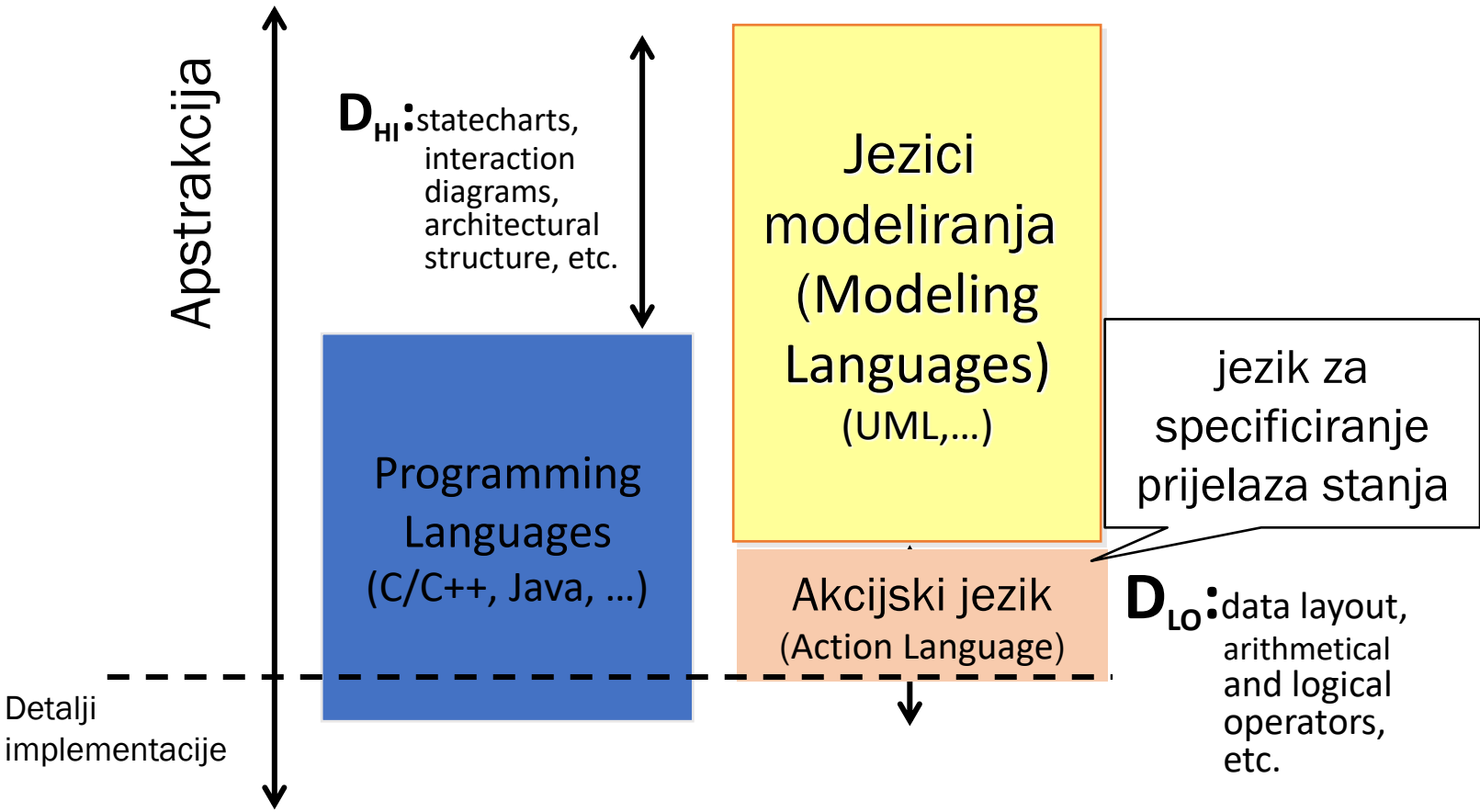
- Razlike modeli i ostvarenja:
  - posebnosti upotrjebljenih materijala
  - posebnosti metoda izgradnje
  - problem stvarne veličine
  - vještine i tehnologija
- Razlike mogu dovesti do ozbiljnih pogrešaka i odstupanja u realizaciji!

Uporaba modela

- Razumijevanje složenih sustava
  - specifikacija zahtjeva
  - projektiranje
  - rano otkrivanje pogrešaka
    - analiza, simulacija
  - olakšano komuniciranje
- Osnova za implementaciju

- Svojstva modela
  - Apstrakcija
  - Razumljivost
  - Točnost
  - Predvidljivost
    - daje odgovor na pitanja o promatranom sustavu
  - Jednostavnost

Programiranje i modeliranje



Osobito svojstvo programa

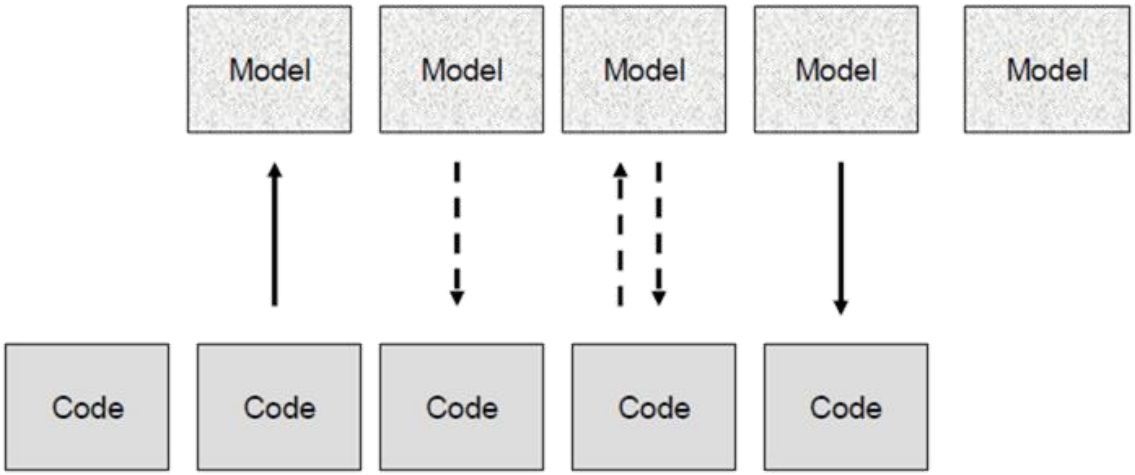
- Programi imaju jedinstveno svojstvo koje omogućava **razvoj apstraktnih modela u potpunu implementaciju** bez promjene inženjerske okoline, metoda ili alata.
  - modeli se mogu nadograđivati sve do potpune specifikacije
  - model postaje sustav kojeg je u početku samo modelirao!
- Moguće generiranje apstraktnih modela izravno i automatizirano iz implementacije
  - to osigurava potpunu točnost modela programa
  - model ≡ sustav

Posljedice

- Programska potpora je manje ovisna o okolini
  - ne potpuno neovisna
- Individualne mogućnosti dolaze do izražaja
  - produktivnost pojedinca značajno različita
  - nije nužno mjera kvalitete...
  - ...niti inteligencije
- Brz razvojni ciklus
  - uzrokuje nestrpljivost
    - ... nesistematičnost, “hack”
    - velika obećanja

engl. hack: sjeći, udarati, probiti put.

Oblikovanje PP zasnovano na modelima

- *engl. Model-Driven Software Development*
- Postavljaju model u centar razvojnog procesa PP
  - metode: apstrakcije i automatizacija
- Povijest uporabe modela:
- 

- Cilj: povećanje razine apstrakcije, poboljšanje komunikacije, produktivnost, održavanje ....

Poboljšanja

- Kako oblikovati programsku potporu da se smanji vjerojatnost neuspjeha?

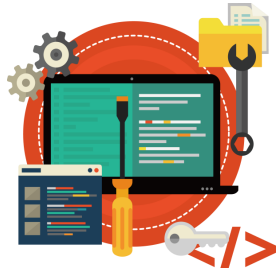
1. Uvesti inženjerski propisane ***postupke*** (procedure) u proces oblikovanja programske potpore.
   - precizno definirati faze procesa oblikovanja – tko radi što i kada.
2. Svaku fazu procesa ***dokumentirati*** (po mogućnosti na standardan i formalan način).
3. U oblikovanje uvesti ***analizu i izbor stila arhitekture*** programske potpore te pripadne modele pogodne za formalnu analizu.
4. Programsku potporu oblikovati u manjim cjelinama (***komponentama***).
5. Uz tradicijsko ispitivanje uvesti ***formalne metode provjere*** (verifikacije) modela programske potpore.

Proces oblikovanja programske potpore

- Ovaj dio problema oblikovanja programske potpore analizira i predlaže rješenja disciplina koju nazivamo ***programsko inženjerstvo*** (*engl. Software engineering*).
- Nema jedinstvenog i standardnog prijedloga
  - mogu se izlučiti neke generičke aktivnosti u svim prijedlozima.
  - osnove najpopularnijih pristupa razmotrit će se u nastavku predavanja.
- Posebice je bitno razmotriti postupke precizne specifikacije zahtjeva koje korisnik postavlja na sustav.
  - ***inženjerstvo zahtjeva*** (*engl. Requirements engineering*).

Sažetak

- Metodologija oblikovanja programske potpore kontinuirano se razvija.
- Postoji stalna ozbiljna kriza razvoja programske potpore.
- Osnovni cilj programskog inženjerstva je oblikovanje programske potpore sa smanjenom vjerojatnošću neuspjeha.
- Moderno oblikovanje programske potpore zasnovano na modelima standardno u praksi i podržano alatima.
  - Ima mjesta za poboljšanja ☺



PROGRAMSKO INŽENJERSTVO

Programsko inženjerstvo

- Programsko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima izgradnje programske potpore.
  - metodama i alatima za profesionalno oblikovanje i proizvodnju programske potpore uzimajući u obzir cjenovnu efikasnost
- Programsko inženjerstvo
  - ***sistematski i organiziran*** pristup procesu izrade;
  - upotrebljavati ***prikladne alate i tehnike*** ovisno o ***problemu*** koji treba riješiti, ***ograničenjima*** u procesu izrade i postojećim ***resursima***.
  - proces rješavanja problema kupaca i korisnika
    - ponekad je rješenje: kupi, ne razvijaj!

IEEE definicija:

the application of a *systematic, disciplined, quantifiable* approach to the development, operation, maintenance of software; that is, the application of engineering to software.

PROGRAMSKO INŽENJERSTVO ↔ RAČUNARSKA ZNANOST

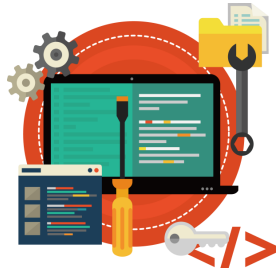
- Računarska znanost se bavi *teorijom i temeljima*; programsko inženjerstvo se bavi *praktičnim problemima razvoja, oblikovanja i isporuke korisnog računalnog programa*.
- Teorije u okviru računarske znanosti još uvijek su nedostatne za ukupnu podlogu programskom inženjerstvu
  - npr. razlika od fizike u elektrotehnici i elektronici.

PROGRAMSKO INŽENJERSTVO ↔ INŽENJERSKI SUSTAVI

- Inženjerstvo sustava – engl. *system engineering*
  - Interdisciplinarno - sagledavanje cjeline, primjenjivo na sve složene sustave
- Inženjerstvo računalnih sustava (engl. *computer system engineering*) se bavi svim aspektima sustava zasnovanim na računalima (sklopovlje, programska potpora, inženjerstvo procesa).
  - Programsko inženjerstvo je dio procesa usredotočeno na razvoj programske infrastrukture, upravljanje, primjenu i baze podataka u sustavu.
- Inženjeri sustava (računalnog) su uključeni u specificiranje sustava, oblikovanju arhitekture, integraciju i postavljanju u korisničko okruženje.

Osnovni izazovi programskog inženjerstva

- Heterogenost (engl. *Heterogeneity*)
  - tehnike razvoja programske potpore za različite platforme okoline izvođenje.
- Vrijeme isporuke (engl. *Delivery*)
  - tehnike koje osiguravaju kratak interval od zamisli do stavljanja na tržište (engl. *Time-to-market*) uz prihvatljivu kakvoću
- Povjerenje (engl. *Trust*)
  - tehnike koje pokazuju da korisnici mogu imati povjerenja u programski sustav (npr. uporaba matematičkih, tj. formalnih metoda).
- Promjena zahtjeva (engl. *Requirements changes*)
  - Postoji česta i iznenadna promjena zahtjeva kupaca.
- Složenost (engl. *Complexity*)
  - Programska potpora je složen (kompliciran) sustav s velikim brojem detalja.



ETIKA

Profesionalna i etička odgovornost

- Programski inženjer se mora ponašati profesionalno korektno i etički odgovorno. Etičko ponašanje je više nego puko pridržavanje zakona.
  - mora biti svjestan na čemu rade, kao i utjecaja toga na okoliš.
  - moralno, inženjeri trebaju poštivati privatnost javnosti.
  - engl. Codes of Ethics (CoEs) / Codes of Conduct (CoCs)
- Izazov razvoja program koji su sigurni i zadovoljavaju standard
  - ne smije umanjivati kvalitetu života i privatnost
- Povjerljivost prema poslodavcu i kupcu
  - Formalni ugovori
    - CONFIDENTIALITY AND NON-DISCLOUSURE AGREEMENT
- Kompetencije
  - prihvaćanje posla u okviru svojih kompetencija.
  - Razumijevanje zahtjeva u potpunosti primjenim odovarajućih metodologija
- Poštenje i profesionalnost
  - Čuvajte integritet i neovisnost
- Kolegijalnost
- Poštivanje prava intelektualnog vlasništva.
- Ne zlouporabiti računalne sustave
  - npr. igranje za vrijeme rada, širenje virusa, spama ...
- IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices
  - https://ethics.acm.org/code-of-ethics/software-engineering-code/, 2016
  - https://www.ieee.org/about/corporate/governance/p7-8.html June 2020

Načela etike i profesionalnog rada PI

- Programski inženjeri obvezuju se da će analizu, specifikaciju, oblikovanje, razvoj, provjeru i održavanje programskih proizvoda učiniti korisnom i poštovanom strukom. U skladu sa svojom obvezom prema zdravlju, sigurnosti i dobrobiti javnosti, programski inženjeri se obvezuju pridržavati slijedećih osam načela:
  1. JAVNI INTERES – Programski inženjeri djeluju u skladu s javnim interesom.
  2. KLIJENT I POSLODAVAC – Programski inženjeri djeluju u skladu s interesima njihovog klijenta ili poslodavca, a koji nisu u suprotnosti s javnim interesom.
  3. PROIZVOD – Programski inženjeri obvezuju se osigurati da njihovi proizvodi i prateće izmjene zadovoljavaju najviše moguće standarde struke.
  4. PROSUDBA – Programski inženjeri održavaju integritet i neovisnost u svojim stručnim prosudbama
  5. UPRAVLJANJE – Menedžeri i voditelji programskog inženjeringa potiču i promiču etički pristup upravljanju programskim razvojem i održavanjem.
  6. STRUKA – Programski inženjeri unapređuju čestitost i ugled struke u skladu s javnim interesom.
  7. KOLEGIJALNOST – Programski inženjeri međusobno se podržavaju i ophode pošteno.
  8. ODNOS PREMA SEBI – Programski inženjeri se kontinuirano stručno unapređuju i promiču etički pristup radu u svojoj struci.

IEEE kod etičnosti

- IEEE Policies, Section 7 - Professional Activities

Mi, članovi udruge IEEE, prepoznavajući *važnost tehnologija* i njihov utjecaj na kvalitetu života u cijelom svijetu, te prihvaćajući *osobnu obvezu prema vlastitom zanimanju, kolegama i zajednicama kojima služimo*, ovime se obvezujemo na najviše etičko i profesionalno ponašanje te smo odlučili:

- I .To uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities.
  1. to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment;
  2. to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;
  3. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;
  4. to avoid unlawful conduct in professional activities, and to reject bribery in all its forms;
  5. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others;
  6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
- II. To treat all persons fairly and with respect, to not engage in harassment or discrimination, and to avoid injuring others.
  7. to treat all persons fairly and with respect, and to not engage in discrimination based on characteristics such as race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;
  8. to not engage in harassment of any kind, including sexual harassment or bullying behavior;
  9. to avoid injuring others, their property, reputation, or employment by false or malicious actions, rumors or any other verbal or physical abuses;
- III. To strive to ensure this code is upheld by colleagues and co-workers.
  10. to support colleagues and co-workers in following this code of ethics, to strive to ensure the code is upheld, and to not retaliate against individuals reporting a violation.

Izvor: IEEE Code of Ethics, 2020

Zaključak

- Postoji stalna kriza razvoja programske potpore.
  - Složenost
  - Promjene
  - Timski rad
- Programsko inženjerstvo je inženjerska disciplina usredotočena na izgradnju i održavanje visokokvalitetnih programskih sustava i usluga koji pružaju vrijednost korisnicima i društvu u cjelini.
- Moderno oblikovanje programske potpore zasnovano na modelima standardno u praksi i podržano alatima.
  - Ne postoji jedan standardni postupak!
  - Ne postoji najbolji jezik ili razvojno okruženje!
- Metodologija razvoja malih projekata znatno se razlikuje od razvoja složenih projekata programske potpore
- Kvaliteta programa koju postaje ključna za napredak društva u cjelini
  - Osim tehničkih mogućnosti, kvaliteta proizvoda ovisi o etici i profesionalnom ponašanju inženjera.