PROGRAMSKO INŽENJERSTVO

ak. god. 2020./2021.

Životni ciklus programske potpore i inženjerstvo zahtjeva

.. Do sada

- Izrada i organizacija projekta
- Problem razvoja programske potpore
- Inženjerski postupci
- Programsko inženjerstvo
- Svojstva programske potpore
- Izazovi u uloga modela u razvoju programa

Pregled tema

- Životni ciklus programske potpore
- Dionici na projektu i timski razvoj
- Inženjerstvo zahtjeva
  - Klasifikacija zahtjeva
  - Procesi inženjerstva zahtjeva
    - Generičke aktivnosti inženjerstva zahtjeva
    - Metode izlučivanja zahtjeva

Pripremio i prilagodio: Vlado Sruk, Alan Jović, Nikolina Frid

Životni ciklus programske potpore

Proces razvoja programske potpore

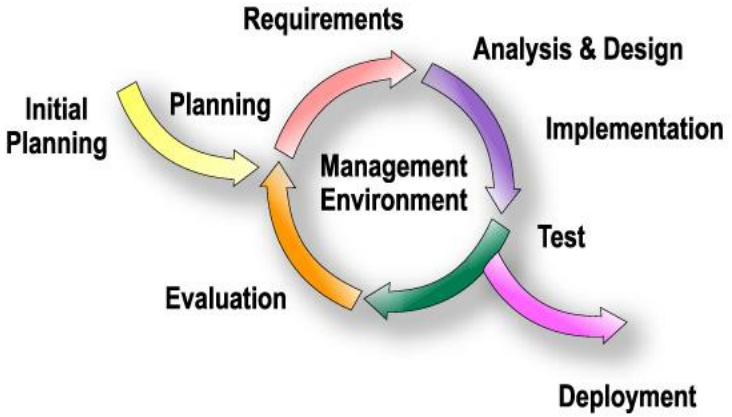
- Sinonimi: životni ciklus razvoja, životni ciklus, proces (engl. *software development life-cycle SDLC, software development process, software life cycle, software methodology, software process*)
- Obzirom na ranije diskutirane probleme razvoja složene programske potpore potreban je inženjerski način pretvorbe ideje projekta u funkcionalnu i potpuno operativnu strukturu
- Cilj SDLC je poboljšanje procesa oblikovanja proizvoda (programa), upravljanja proizvodom i upravljanja projektima
  - potrebno je proces podjele rada na razvoju programa na manje, paralelne i/ili uzastopne korake ili podprocese
- Model životnog ciklusa programske potpore engl. *SDLC model*
  - okvir koji opisuje aktivnosti provedene u svakoj fazi projekta razvoja programske potpore

Model životnog ciklusa

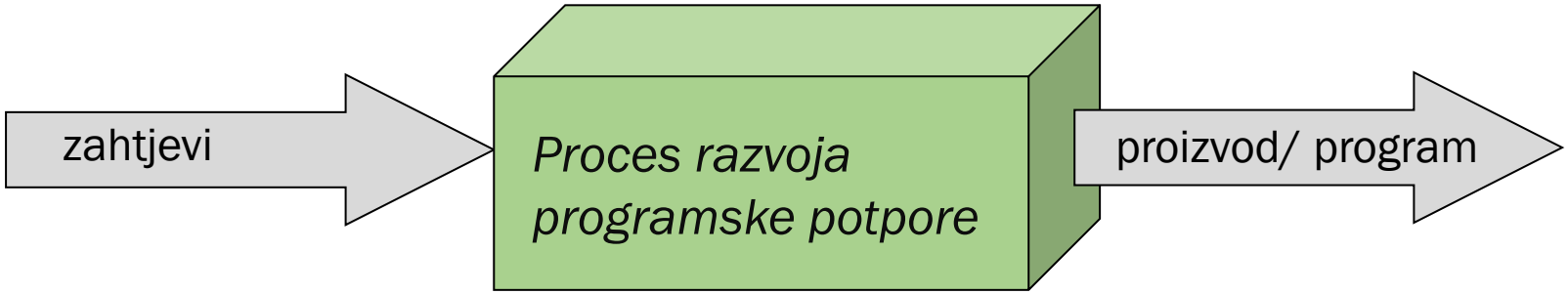
- Cjelokupni proces planiranja, analize, izrade, ispitivanja, puštanja u rad i evolucije programske potpore
- Model životnog ciklusa (engl. *lifecycle model*)
  - opisuje način kako se ostvaruju faze od začetka projekta do kraja životnog vijeka programske potpore
  - sastoji se od **faza** – niza povezanih aktivnosti koje se provode u svrhu ostvarenja cilja
  - opisuje odnose glavnih točaka procesa i rezultata
  - predstavlja normirani format procesa programske potpore
    - Npr. **ISO/IEC/IEEE 12207:2017** – *Systems and software engineering — Software life cycle processes*
  - postoji mnogo modela životnog ciklusa, u praksi se kombiniraju

Uloga procesa

- Proces razvoja programske potpore osigurava:
  - potrebne informacije
  - u trenutku kada su potrebne
  - u upotrebljivom obliku
    - ni previše ni premalo
    - jednostavno pronalaženje



- Proces definira TKO radi ŠTO, KADA i kako postići željeni cilj.

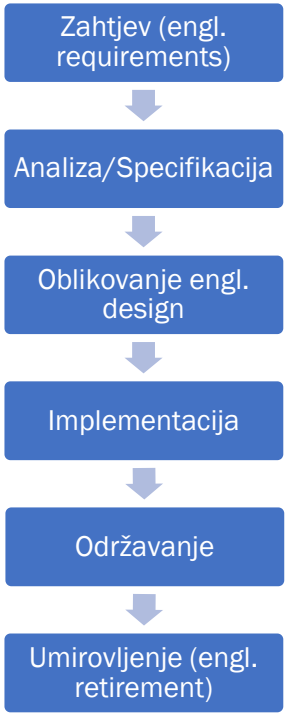


Proces programskog inženjerstva

- Skup aktivnosti čiji cilj je razvoj ili evolucija programskog proizvoda
  - u primjeni velik broj procesa programskog inženjerstva
- Generičke aktivnosti procesa programskog inženjerstva

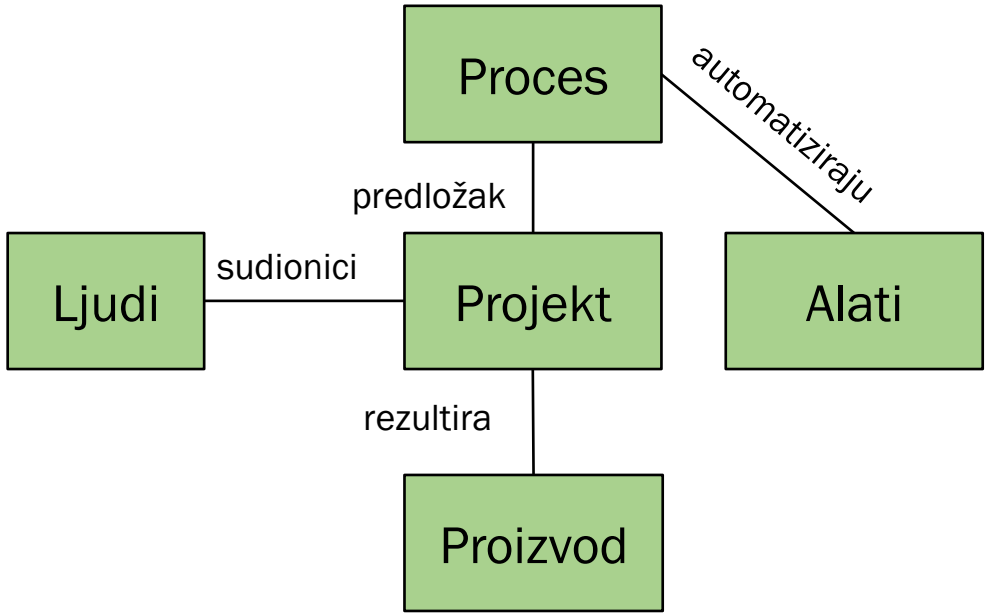
1.	Specifikacija	Temeljem analize zahtjeva određuje što sustav treba činiti i koja su ograničenja u razvoju
2.	Oblikovanje i implementacija	Izbor arhitekture i ostvarenje programskog sustava
3.	Validacija i verifikacija	Provjera da li sustav čini ono što se od njega zahtijeva
4.	Evolucija	Promjene sukladno novim zahtjevima

Idealan razvoj:



Upravljanje projektima

- Upravljanje projektima programske potpore (engl. *software project management*) je krovna aktivnost u području programskog inženjerstva
- Efikasno programsko inženjerstvo fokusira se na:
  - Ljude
    - najvažniji
  - Projekt
    - rezultira proizvodom
  - Proizvod
    - nije samo kod
  - Proces
    - upravlja projektom
  - Alate



Primjer ISO/IEC/IEEE 12207:2017

- Dogovaranje engl. *Agreement*
  - Sve aktivnosti pokretanja projekta (dogovor isporučitelja i naručitelja)
- Organizacija projekta – engl. *organizational project-enabling*
  - Potporni procesi i organizacija omogućuju, kontroliraju i podržavaju životni ciklus sustava i srodne projekte.
- Upravljanje - engl. *technical management*
  - planiranje, procjena i kontrola životnog ciklusa, osiguravajući kvalitetu
- Tehnički procesi - engl. *technical processes*
  - obuhvaća 14 tehničkih aktivnosti i osoblja (informacijska tehnologija, alati za rješavanje problema, stručnjaci za softver itd.) tijekom rada prije, poslije i tijekom rada

Software Life Cycle Processes

Agreement Processes

- Acquisition Process (6.1.1)
- Supply Process (6.1.2)

Organizational Project-Enabling Processes

- Life Cycle Model Management Process (6.2.1)
- Infrastructure Management Process (6.2.2)
- Portfolio Management Process (6.2.3)
- Human Resource Management Process (6.2.4)
- Quality Management Process (6.2.5)
- Knowledge Management Process (6.2.6)

Technical Management Processes

- Project Planning Process (6.3.1)
- Project Assessment and Control Process (6.3.2)
- Decision Management Process (6.3.3)
- Risk Management Process (6.3.4)
- Configuration Management Process (6.3.5)
- Information Management Process (6.3.6)
- Measurement Process (6.3.7)
- Quality Assurance Process (6.3.8)

Technical Processes

- Business or Mission Analysis Process (6.4.1)
- Stakeholder Needs and Requirements Definition Process (6.4.2)
- Systems/Software Requirements Definition Process (6.4.3)
- Architecture Definition Process (6.4.4)
- Design Definition Process (6.4.5)
- System Analysis Process (6.4.6)
- Implementation Process (6.4.7)
- Integration Process (6.4.8)
- Verification Process (6.4.9)
- Transition Process (6.4.10)
- Validation Process (6.4.11)
- Operation Process (6.4.12)
- Maintenance Process (6.4.13)
- Disposal Process (6.4.14)

Dionici na projektu i timski razvoj

Dionici na projektu

- **Dionik** (engl. *stakeholder*) – svatko tko sudjeluje u projektu ili na koga projekt utječe, u tvrtki koja razvija projekt i izvan nje

- Primarni dionici (engl. *primary stakeholder*) – projekt na njih utječe izravno
  - rukovoditelji razvoja, razvojni tim, strateški klijenti i partneri
- Sekundarni dionici (engl. *secondary stakeholder*) – projekt na njih utječe sporedno ili neizravno
  - klijenti, prodavači, konkurencija, kreditori
- Ključni dionici (engl. *key stakeholders*) – oni donose odluke na projektu ili odlučuju o tome je li projekt uspio ili nije, mogu biti primarni ili sekundarni
- Dionici na projektu unutar tvrtke
  - Razvojni tim
  - Svi drugi koji nisu razvojni tim
- Dionici na projektu izvan tvrtke

Dionici na projektu unutar tvrtke

- Organizacija i podjela dionika na projektu unutar tvrtke ovisi o:
  - korištenom modelu životnog ciklusa
  - vrsti i ciljevima projekta
  - veličini i organizaciji tvrtke
- Primjer česte organizacije dionika:
  - rukovoditelj projekta (engl. *project manager*) – vodi projekt razvoja PP
  - poslovni analitičar (engl. *business analyst*) – traži poslovnu priliku
  - vlasnik proizvoda (engl. *product owner*) – ima viziju ciljnog proizvoda i zadužen je za njegov uspjeh
  - razvojni tim (engl. *development team*) – rade na ostvarenju proizvoda
  - inženjer osiguranja kvalitete (engl. *quality assurance engineer*) – provjerava i osigurava ispunjenje klijentskih zahtjeva

Organizacija razvojnog tima

- Organizacija razvojnog tima ovisi o prirodi projekta
- Specijalizacija za pojedini dio projekta programske potpore je često prisutna i najčešće doprinosi uspješnosti projekta
- Česte specijalizacije:
  - razvojni inženjer poslužiteljske strane (engl. *back-end developer*)
  - razvojni inženjer klijentske strane (engl. *front-end developer*)
  - razvojni inženjer baze podataka (engl. *database developer*)
  - dizajner korisničkog sučelja / korisničkog iskustva (engl. *UI/UX designer*)
  - podatkovni analitičar / podatkovni inženjer (engl. *data analyst/data engineer*)
  - ispitivač programske potpore (engl. *software tester*)
  - integrator sustava (engl. *systems integrator*)

Značajke razvojnog tima

- Veličina tima
  - više nije uvijek bolje, ovisno o količini posla, idealno 4 – 7 članova
  - veći tim bolje podijeliti u specijalizirane podtimove
- Kompetencije članova tima
  - široko opće znanje – poznaju veći broj tehnologija, ali sve površno (engl. *full stack developer*)
  - T-članovi – poznaju veći broj tehnologija površno, ali u jednoj ili par njih su stručnjaci
    - npr. razvojni inženjer poslužiteljske strane i stručnjak u PHP-u, poznaje i tehnologije baze podataka i Javascript
- Vrsta organizacije tima
  - Striktno rukovodstvo – bolje u početku formiranje tima
  - Samoorganizirajući tim – kasnije, s većim iskustvom članova
- Podjela uloga unutar tima
  - Arhitekt programske potpore (engl. *software architect*)
  - Inženjer koji nadgleda razvoj i puštanje u pogon (engl. *devops engineer*)
  - Voditelj tehničkog razvoja (engl. *tech lead*)
  - Glavni ispitivač (engl. *lead tester*)
  - ...

Stupnjevi razvoja razvojnog tima

- Svaki razvojni tim tipično prolazi pet stupnjeva razvoja
  - poznati iz psihologije kao Tuckmanov model razvoja tima
- Formiranje (engl. *forming*)
  - Počinje kada se tim prvi put nađe. Članovi su obično pristojni i prijateljski, nema predvidljivih sukoba. Važnu ulogu igra voditelj tima, budući da su odgovornosti članova nejasno definirane i ljudi se međusobno slabo poznaju
- Uspostava komunikacije/ Orijentiranje (engl. *storming*)
  - Tim počinje zajedno raditi, inicijalni entuzijazam splasne i počinju mogući konflikti. Voditelj tima treba prepoznati i razriješiti krizne situacije po mogućnosti čim prije kako ne bi utjecale na uspjeh projekta
- Definiranje uloga i normi/Izrastanje (engl. *norming*)
  - Tim počinje raditi kao jedna jedinica. Interakcija između članova tima se sve više poboljšava i ima sve manje konflikata. Počinje prevladavati međusobno uvažavanje i cijenjenje tuđih znanja i sposobnosti
- Ostvarivanje ciljeva/Razvijeni tim (engl. *performing*)
  - Tim postaje samouvjeren, motiviran i samoorganiziran. Voditelj tima sada može delegirati većinu posla članovima i raditi na daljnjim poboljšanjima
- Raspuštanje (engl. *adjourning*)
  - Kad su svi projektni ciljevi ostvareni, projektni tim se raspušta.

Programska potpora timskom razvoju

- Veliki broj dostupnih CASE alata za podršku suradnje i timskog razvoja
- Sustavi za komunikaciju između članova tima
  - Alati za upravljanje projektom
    - SharePoint, Confluence, Jira
  - Alati za brzu komunikaciju
    - Slack, MS Teams
- Sustavi za upravljanje verzijama programske potpore (engl. *version control system*)
  - Tehnologija Git
    - Github, Gitlab, BitBucket
  - Tehnologija Subversion (SVN)
    - Apache Subversion + TortoiseSVN,
  - Tehnologija Mercurial
    - Mercurial + TortoiseHG

Inženjerstvo zahtjeva

Inženjerstvo zahtjeva

- *engl. requirements engineering, requirements gathering, requirements capture, requirements specification*
- To je postupak pronalaženja, analiziranja, strukturiranja, dokumentiranja i provjere **korisnički zahtijevanih usluga** sustava, te **ograničenja u uporabi**
  - fokus: utvrđivanje ciljeva, funkcija i ograničenja sklopovlja i programske podrške
- **Zahtjevi** (engl. *requirements*) opisuju što programski sustav treba raditi kao i ograničenja u njegovom radu.
  - oblikuju dokument koji specificiraju usluge sustava i ograničenja u uporabi
  - Otvoren za tumačenje i dovoljno detaljan za razumijevanje
- Sve što oblikujemo započinje sa zahtjevima!

Klasifikacija zahtjeva

Klasifikacija prema razini detalja

- **Poslovni zahtjevi** (engl. *Bussines requirements*)
  - Opis na visokoj razini apstrakcije (ciljevi, potrebama)
  - Opisuju zašto se projekt započinje i koji je njegov opseg i cilj
    - što se želi ostvariti i/ili probleme koje žele riješiti. Dio dokumenta prikaza poslovne opravdanosti (engl. *business case*).
- **Korisnički zahtjevi** (engl. *User requirements*)
  - specifikacija visoke razine apstrakcije - u okviru ponude za izradu programskog proizvoda
  - pišu se prirodnim jezikom, tablicama i grafičkim dijagramima.
  - u definiranju sudjeluju: klijenti (rukovoditelji, inženjeri), krajnji korisnici sustava, rukovoditelji za pisanje ugovora, specijalisti za oblikovanje sustava (arhitekti)
- **Zahtjevi sustava** (engl. *System requirements*)
  - vrlo detaljna specifikacija - uobičajeno nakon prihvaćanja ponude, a prije sklapanja ugovora
  - pišu se strukturiranim prirodnim jezikom, posebnim jezicima za oblikovanje sustava, dijagramima i matematičkom notacijom.
  - u definiranju sudjeluju: klijenti (inženjeri), specijalisti za oblikovanje sustava (arhitekti), specijalisti za razvoj programske potpore

Klasifikacija s obzirom na sadržaj

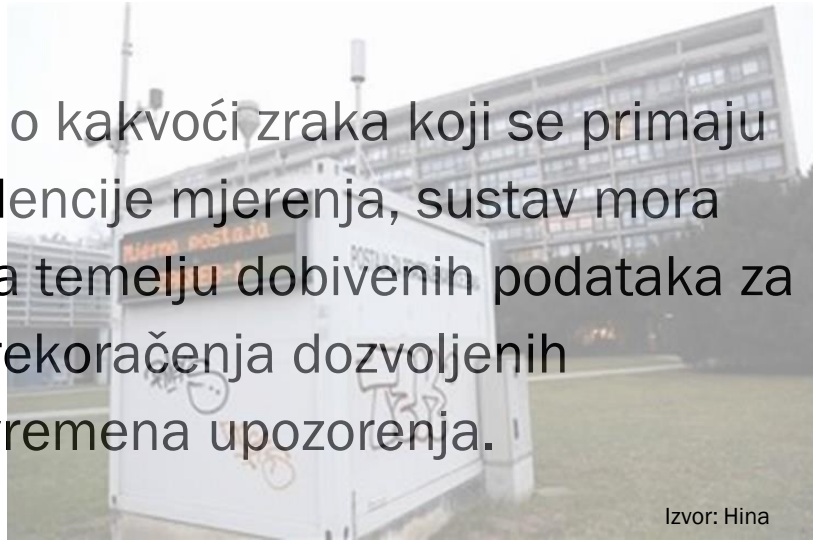
- Odnosi se i na korisničke zahtjeve i na zahtjeve sustava
- Funkcionalni zahtjevi (engl. *functional requirements*)
  - Opisuju ponašanje sustava
  - izjave o uslugama koje sustav mora pružati, kako će sustav reagirati na određeni ulazni poticaj te kako bi se sustav trebao ponašati u određenim situacijama.
  - specifikacija rezultata rada: engl. *"system shall do <requirement>"*
- Nefunkcionalni zahtjevi (engl. *non-functional requirements*)
  - Opisuju ostala poželjne (zahtjevane) atribute sustava
  - ograničenja u uslugama i funkcijama, kao što su vremenska ograničenja, ograničenja u procesu razvoja i oblikovanja i sl.
  - naglasak na karakteristikama: engl. *"system shall be <requirement>"*
- Zahtjevi domene primjene
  - zahtjevi (funkcionalni i nefunkcionalni) koji proizlaze iz domene primjene sustava kao i oni koji karakteriziraju tu domenu.

Kriteriji zahtjeva

- Osnovni kriteriji koje mora ispuniti pravilno formuliran zahtjev:
  - Potreba
    - što je točno potrebno da zadovolji korisnika
    - zahtjevi moraju odgovarati stvarnim potrebama
  - Provjerljivost
    - analiza, ispitivanje, promatranje ili demonstracija zahtjeva
    - cilj: napisati zahtjeve na objektivan, jasan i lako razumljiv način
  - Ostvarivost
    - ako nije moguće zadovoljiti potrebe zbog ograničenih resursa, onda ne bi trebalo dokumentirati i razrađivati takav zahtjev.
    - istražiti što je moguće ostvariti (obavezno dokumentirati)

Primjer funkcionalnih zahtjeva

- Sustav za praćenje kvalitete zraka
- Web aplikacija* koja povezuje podatke o kakvoći zraka koji se primaju sa svih mjernih postaja u RH. Osim evidencije mjerenja, sustav mora davati i statističku analizu te izvješća na temelju dobivenih podataka za zadani vremenski period te u slučaju prekoračenja dozvoljenih vrijednosti onečišćivača izdavati pravovremena upozorenja.


Izvor: Hina

Cjeloviti tekst na Moodleu

1. *Provesti evidenciju mjerenja. Za svako mjerenje bilježe se datum, vrijeme i vrijednost očitanja.*
2. *Potrebno je davati statistiku povijesnih podataka.*
3. *Ukoliko se tijekom mjerenja na pojedinoj postaji očita veća vrijednost od granične, sustav treba izdati upozorenje.*

Poteškoće: Prirodni jezik

- Zahtijeva socijalne i komunikacijske vještine
- Nedostatak jasnoće
  - preciznost nije lako postići bez detaljiziranog i teško čitljivog dokumenta
- Miješaju se funkcionalni i nefunkcionalni zahtjevi
- Nenamjerno objedinjavanje više zahtjeva u jednom
- Nejasno postavljeni zahtjevi mogu biti različito interpretirani od korisnika i razvojnih timova
  - uzrokuje probleme u procesu razvoja i kršenje ugovora
- Npr. u sustavu za praćenje kvalitete zraka: “davati statistiku”:
  - korisnik – grafički prikaz u sučelju s mogućnosti pregleda svakog mjeseca
  - razvojni tim – zbirna statistika, prikaz numeričkih podataka

Primjer: klasifikacija zahtjeva

- *„Mjerenja se provode u jednakim vremenskim razmacima (5 min) na način da web aplikacija pošalje upit svakoj mjernoj postaji koja zatim vrati očitanje sa svojih senzora. Za komunikaciju između web poslužitelja i mjernih postaja koristit će se protokol SSH. Prilikom slanja upita mjernoj postaji, web aplikacija će se pokušati spojiti najviše 3 puta uz maksimalno vrijeme isteka (engl. timeout) od 5 sekundi. Za svako mjerenje bilježe se datum, vrijeme i vrijednost očitanja.”*
- Miješaju se različiti tipovi zahtjeva:
  - Funkcionalni
    - slanje upita mjernoj postaji i dobavljanje očitanja
    - bilježenje datuma, vremena i vrijednosti očitanja u bazu podataka
  - Nefunkcionalni
    - vremenski razmak između mjerenja (5 min)
    - korištenje protokola SSH
    - broj pokušaja spajanja (3 puta) i maksimalno vrijeme isteka (5 sekundi)

Svojstva zahtjeva

- Zahtjev mora dati odgovor na pitanje što a ne kako.
- Potrebna (željena) svojstva
  - Korektnost (engl. *correct*)
  - Provjerljivost (engl. *Verifiable*)
  - Jednoznačnost (engl. *Unambiguous*)
  - Kompletnost (engl. *Complete*)
    - sadrže opise svih zahtijevanih mogućnosti
  - Konzistentnost (engl. *Consistent*)
    - ne smiju sadržavati konflikte ili kontradikcije u opisima zahtijevanih mogućnosti
  - Promjenjivost (engl. *Modifiable*)
- U praksi je nemoguće postići **kompletan i konzistentan** dokument o zahtjevima složenih sustava.

Klasifikacija nefunkcionalnih zahtjeva

- **Zahtjevi programskog proizvoda**
  - zahtjevi koji specificiraju da se isporučeni proizvod mora ponašati na osobit način (npr. vrijeme odziva).
- **Organizacijski zahtjevi**
  - zahtjevi koji su rezultat organizacijskih pravila i procedura (npr. uporaba propisanog normiranog procesa razvoja, DoD ADA)
- **Vanjski zahtjevi**
  - zahtjevi koji proizlaze izvan sustava i razvojnog procesa (međusobna operabilnost, legislativni zahtjevi i sl.).

Nefunkcionalni zahtjevi moraju biti mjerljivi!

Primjer nefunkcionalnih zahtjeva

- Primjer sustava za praćenje kvalitete zraka
  - Zahtjevi programskog proizvoda
    - *npr. vremenski razmak između mjerenja (5 min)*
  - Organizacijski zahtjevi
    - *provjera kvalitete mjerenja i podataka treba slijediti normu HRN EN ISO/IEC 17025*
  - Vanjski zahtjevi
    - *sustav neće otkriti vanjskim sudionicima mjerene podatke, što će se ostvariti korištenjem protokola SSH*

Zahtjevi domene primjene

- Zahtjevi domene primjene mogu biti **novi funkcionalni zahtjevi** ili **ograničenja na postojeće zahtjeve**
- Problemi zahtjeva domene:
  - **razumljivost** – programeri ne razumiju domenu primjene i traže detaljan opis zahtjeva
  - **implicitnost** – specijalisti domene poznaju primjenu tako dobro da podrazumijevaju zahtjeve (koje tada eksplicitno ne određuju)

- Npr. zahtjevi domene primjene za sustav za praćenje kvalitete zraka:
  - *Ukoliko se aplikacija ne uspije povezati s mjernom postajom ili ne uspije očitati stanje jednog ili više senzora na mjernoj postaji, dežurnom djelatniku se šalje SMS poruka svakih 60 min sve dok ne isključi (deaktivira) pokvareni senzor.*
  - *Vrijednosti koje prelaze graničnu vrijednost definiranu u sustavu za svaki pojedini onečišćivač trebaju biti dodatno označene/naglašene.*

Zahtjevi sustava

- **Detaljnija specifikacija** funkcija sustava, njegovih usluga i ograničenja nego zahtjevi korisnika
  - uloga tih specifikacija je definiranje oblikovanja sustava
- Zahtjevi sustava mogu se definirati ili prikazati nekim od modela sustava
- Odnos zahtjeva sustava i oblikovanja
  - zahtjevi određuju **ŠTO** sustav mora raditi, a oblikovanje (dizajn) određuje **KAKO** će se to ostvariti
  - u praksi su zahtjevi i oblikovanje neodvojivi
    - arhitektura sustava strukturira zahtjeve
    - sustav često mora raditi u sinergiji s drugim sustavima koji generiraju zahtjeve na oblikovanje

Izražavanje zahtjeva sustava

- **Strukturirani prirodni jezik**
  - definiranje normiranih formulara i obrazaca u kojima se izražavaju zahtjevi (definicije, ulazni podaci i izvori, prethodni i posljedični uvjeti, popratni učinci)
  - prednost ovakve specifikacije je u zadržavanju izražajnosti prirodnog jezika, ali uz nametnutu izvjesnu uniformnost, a nedostatak je ograničena terminologija
  - u praksi nema usvojene globalne norme za strukturirani prirodni jezik zahtjeva
- **Jezik za opis oblikovanja (npr. SDL)**
  - poput programskog jezika, ali s više apstraktnih obilježja, definira se operacijski model sustava
- **Grafička notacija (npr. UML)**
  - grafički jezik proširen tekstom
- **Matematička specifikacija (FSM, teorija skupova i sl.)**
  - notacija zasnovana na matematičkom konceptu. Najstrože definirana specifikacija. Korisnici je ne vole jer je ne razumiju

Normiranje zahtjeva

- **29148-2018** - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes – Requirements engineering

  https://standards.ieee.org/standard/29148-2018.html

  - ova norma specificira sve potrebne procese koji se provode tijekom inženjerskih aktivnosti, a koji rezultiraju u zahtjevima na sustave i programske proizvode
  - zamjenjuje više ranijih normi
- ISO/IEC 25010, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models
  - Zamjenjuje ranije norme ISO/IEC 9126 i FURPS+

Zaključci o zahtjevima

- Zahtjevi postavljaju što sustav treba raditi i definiraju ograničenja u implementaciji i radu sustava
- Korisnički zahtjevi su izjave na višoj apstraktnoj razini što bi sustav trebao raditi, a pišu se prirodnim jezikom, tablicama, dijagramima
- Zahtjevi sustava su detaljne specifikacije o funkcijama sustava, a pišu se strukturiranim prirodnim jezikom, specifičnim jezicima oblikovanja, grafičkom notacijom, matematičkom specifikacijom
- Funkcionalni zahtjevi definiraju usluge koje sustav osigurava
- Nefunkcionalni zahtjevi postavljaju ograničenja na sustav ili na proces oblikovanja sustava i moraju biti mjerljivi
- Zahtjevi domene primjene su specifični (funkcionalni i nefunkcionalni) zahtjevi za određenu domenu koja se razmatra

Procesi inženjerstva zahtjeva

Procesi inženjerstva zahtjeva

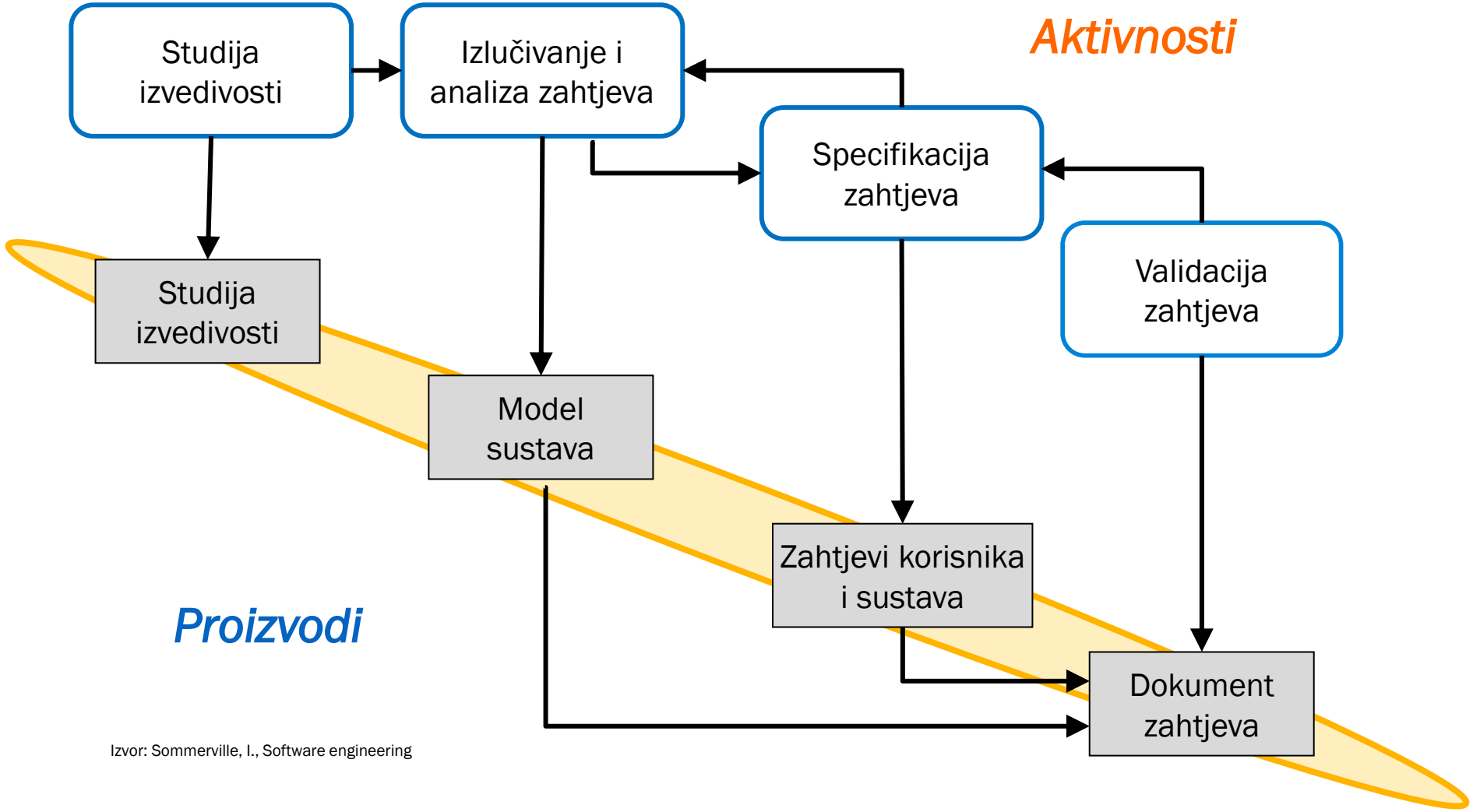
- Proces predstavlja strukturiran (organiziran) skup aktivnosti koji vodi nekom cilju
- **Proces inženjerstva zahtjeva** je skup aktivnosti koje generiraju i dokumentiraju zahtjeve
- Ciljevi:
  - opisati temeljne inženjerske aktivnosti u generiranju i dokumentaciji zahtjeva
  - upoznati se s tehnikama za izlučivanje i analizu zahtjeva
  - opisati validaciju zahtjeva i ulogu recenzenta
  - analizirati upravljanje zahtjevima (*engl. requirements management*) kao potporu procesu inženjerstva zahtjeva

Generičke aktivnosti inženjerstva zahtjeva

Procesi inženjerstva zahtjeva

- Procesi u inženjerstvu zahtjeva razlikuju se ovisno o domeni primjene, ljudskim resursima i organizaciji koja oblikuje zahtjeve
  - nema jedinstvenog procesa inženjerstva zahtjeva!
- Dva uobičajena **modela** procesa inženjerstva zahtjeva:
  - klasični i spiralni
- Generičke aktivnosti inženjerstva zahtjeva:
  - studija izvedivosti (*engl. feasibility study*)
  - izlučivanje i analiza zahtjeva (*engl. requirements elicitation*)
  - specifikacija zahtjeva
  - validacija zahtjeva
  - upravljanje zahtjevima

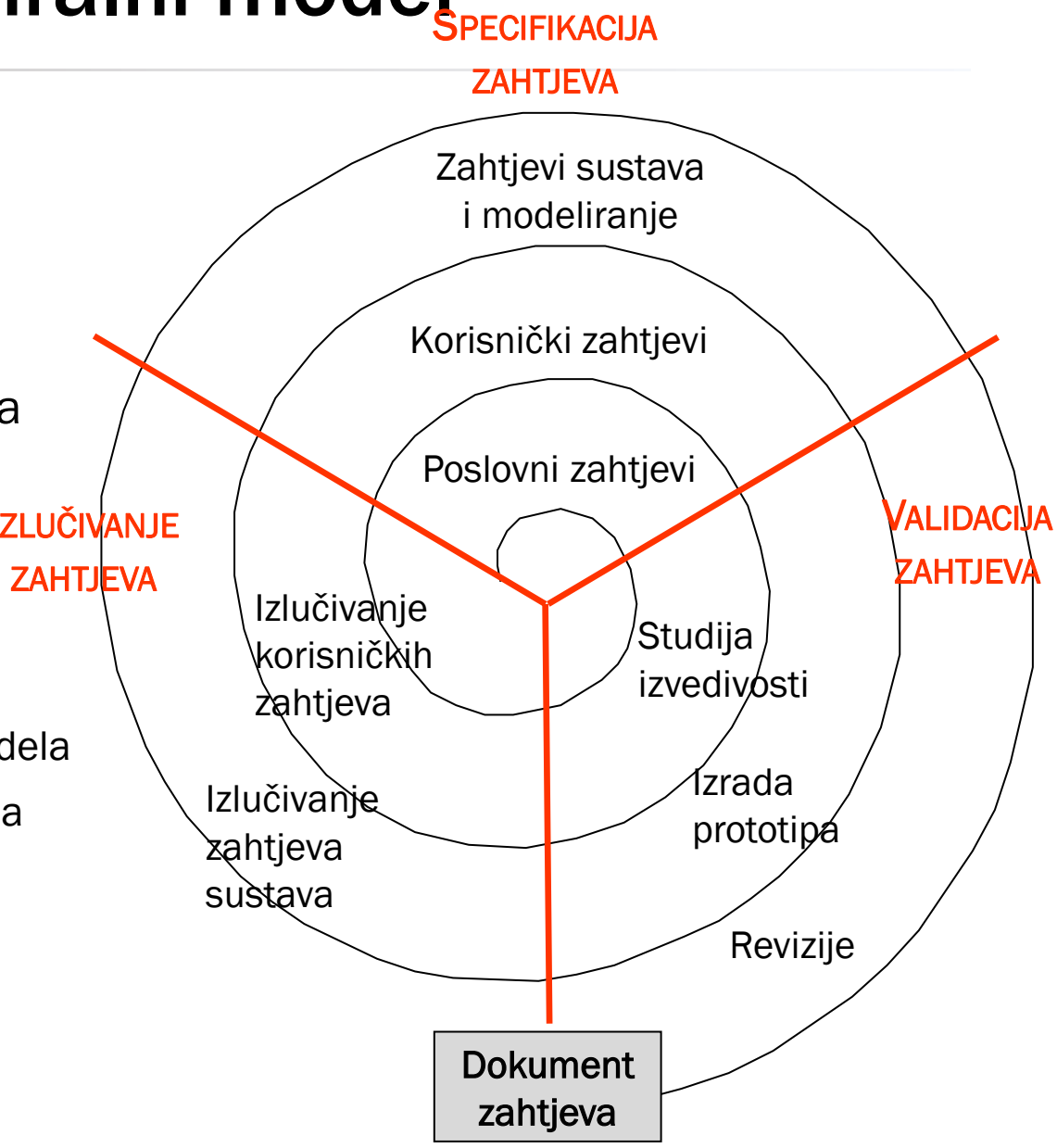
Klasični model



Izvor: Sommerville, I., Software engineering

Spiralni model

- Trostupanjska aktivnost
  - specifikacija
  - validacija
  - izlučivanje zahtjeva
- Promatra proces inženjerstva zahtjeva kroz iteracije
- U svakoj iteraciji je različit intenzitet aktivnosti
  - u ranim iteracijama fokus na razumijevanju poslovnog modela
  - u kasnijim modeliranje detalja sustava
- Zahtjevi se u pojedinim iteracijama specificiraju s različitom razinom detalja



Studija izvedivosti

- *engl. feasibility study*
- Na početku procesa inženjerstva zahtjeva, određuje isplati li se predloženi sustav
  - ulaz predstavljaju preliminarni zahtjevi
  - traži se odgovor na pitanje je li novi sustav vrijedan uloženih sredstava
- Kratka fokusirana studija koja provjerava:
  - doprinose sustava ciljevima organizacije u koju se uvodi
  - mogućnosti ostvarenja postojećom tehnologijom i predviđenim sredstvima
  - mogućnosti integracije predloženog sustava s postojećim sustavima organizacije u koju se uvodi

Provedba studije izvedivosti

- Temelji se na određivanju koje informacije su potrebne za studiju, prikupljanje informacija i pisanju izvješća
- Pitanja za korisnike:
  - što ako se sustav ne implementira?
  - koji su trenutni problemi procesa organizacije?
  - kako bi predloženi sustav pomogao u poboljšanju procesa?
  - koji se problemi mogu očekivati pri integraciji novoga sustava?
  - je li potrebna nova tehnologija ili nove vještine?
  - koje dodatne resurse organizacije traži implementacija novoga sustava?

Izlučivanje i analiza zahtjeva

- *engl. requirements elicitation*
  - najznačajnija aktivnost u procesu inženjerstva zahtjeva
  - poznata je i pod terminom otkrivanje zahtjeva (*engl. requirements discovery*)
- Uključuje stručno tehnički obrazovano osoblje koje u zajedničkom radu s kupcima i korisnicima:
  - razjašnjava domenu primjene
  - definira usluge koje sustav treba pružiti
  - određuje ograničenja u radu sustava
- Može uključivati različite dionike: krajnje korisnike sustava, rukovoditelje, inženjere uključene u održavanje sustava, eksperte domene primjene, predstavnike sindikata i sl.

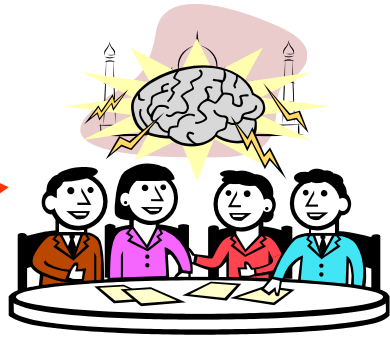
Koraci uspješnog izlučivanja zahtjeva

- Koje informacije trebam?
  - s kojim dionicima razgovarati?
- Gdje doći do potrebnih informacija?
- Prikupljanje informacija
  - ankete, intervjui, promatranje ...
- Organiziranje/analiza prikupljenih zahtjeva
  - funkcijski/nefunkcijski ...
  - grupiranje prema korisnicima, složenosti .....
- Provjera zahtjeva
- REZULTAT: navedeni zahtjevi (engl. *stated requirements*)
  - ono što su nam dionici prezentirali kao zahtjeve
  - u kasnijim iteracijama procesa dolazi se do ispravnih zahtjeva

Rezultati procesa izlučivanja

Opis problema
engl. problem statement

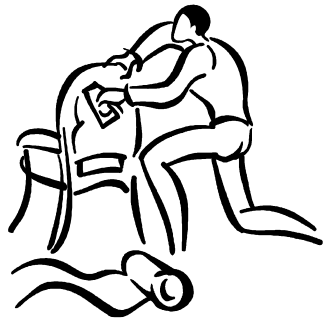
Izlučivanje zahtjeva



Specifikacija zahtjeva
engl. Requirements specification
- definiranje sustava **razumljivo korisniku**
- **komunikacija** s klijentima i korisnicima

Analiza zahtjeva

Analitički model zahtjeva
engl. Analysis model
- **tehnička** specifikacija razumljiva razvojnom timu
- **formalna** i poluformalna sintaksa
- komunikacija s razvojnim timom



Spiralni model izlučivanja i analize zahtjeva

4 osnovne aktivnosti:

- **Izlučivanje/otkrivanje zahtjeva**
  - interakcija s dionicima s ciljem otkrivanja njihovih zahtjeva. *Zahtjevi domene primjene se također definiraju na ovom stupnju. Izvori informacija su dokumenti, dionici, slični sustavi*
- **Klasifikacija i organizacija zahtjeva**
  - grupiraju se srodni zahtjevi i organiziraju u koherentne grozdove (klastere)
- **Ustanovljavanje prioriteta i pregovaranje**
  - zahtjevi se razvrstavaju po prioritetima i razrješuju konflikti
- **Dokumentiranje zahtjeva**
  - zahtjevi se dokumentiraju i ubacuju u sljedeći ciklus spirale

Pogledi

- *engl. viewpoints*
- Način strukturiranja zahtjeva tako da oslikava perspektivu i fokus različitih dionika
  - dionici se mogu razvrstati po različitim pogledima
- Ova **više-perspektivna analiza** je značajna jer ne postoji jedan jedinstveni ispravan način u analizi zahtjeva sustava i omogućava razrješavanje konflikata
- Tipovi pogleda:
  - **pogledi interakcije**
    - ljudi i drugi sustavi koji izravno komuniciraju sa sustavom
  - **indirektni pogledi**
    - dionici koji ne koriste sustav izravno, ali utječu na zahtjeve
  - **pogledi domene primjene**
    - karakteristike domene i ograničenja koja utječu na zahtjeve

Primjer: Bankomat

- Bankomat
  - čitač *kartica*
  - tastatura, zaslon
  - utor za umetanje omotnica
  - spremnik novčanica
  - pisač za ispis potvrda
  - ključ za uključivanje/isključivanje
  - komunikacija s bankom

Bankomat: dionici i pogledi

- Dionici sustava bankomata:
  - bankovni klijenti
  - predstavnici banaka

  } pogledi interakcije

  - bankovni rukovoditelji
  - šalterski službenici
  - administratori baza podataka
  - rukovoditelji sigurnosti
  - marketing odjel
  - inženjeri održavanja sustava
    - sklopovlja i programske potpore

  } indirektni pogledi

  - regulatorna tijela za bankarstvo
  - norme u komunikaciji između banaka

  } pogledi domene primjene

Metode izlučivanja zahtjeva

Metode izlučivanja zahtjeva

- Intervjuiranje kao metoda izlučivanja
- Scenarij kao metoda izlučivanja
- Izlučivanje i specificiranje zahtjeva obrascima uporabe (UML “use cases”)
- Specificiranje dinamičkih interakcija u sustavu (UML sekvencijski dijagrami)
- Promatranje rada – etnografija
- Izrada prototipa
- …

Intervjuiranje za izlučivanje zahtjeva

- U formalnom i neformalnom intervjuiranju tim zadužen za inženjerstvo zahtjeva **ispituje** dionike o sustavu koji trenutno koriste te o novopredloženom sustavu
- Tipovi intervjua:
  - **Zatvoreni intervju** – odgovara se na skup prije definiranih pitanja
  - **Otvoreni intervju** – ne postoje definirana pitanja, već se niz pitanja otvara i raspravlja s dionicima
- U praksi intervjui često ne daju dobre rezultate za zahtjeve domene primjene
  - inženjeri zahtjeva često **ne razumiju specifičnu terminologiju** domene
  - eksperti domene toliko poznaju te zahtjeve da ih **ne artikuliraju dobro**.

Scenariji

- Primjeri iz stvarnog života o načinu korištenja sustava
- Sadržaj scenarija:
  - opis početne situacije
  - opis normalnog/standardnog tijeka događaja
  - opis što se eventualno može dogoditi krivo
  - informaciju o paralelnim aktivnostima
  - opis stanja gdje scenarij završava
- Dionici diskutiraju i kritiziraju scenarij

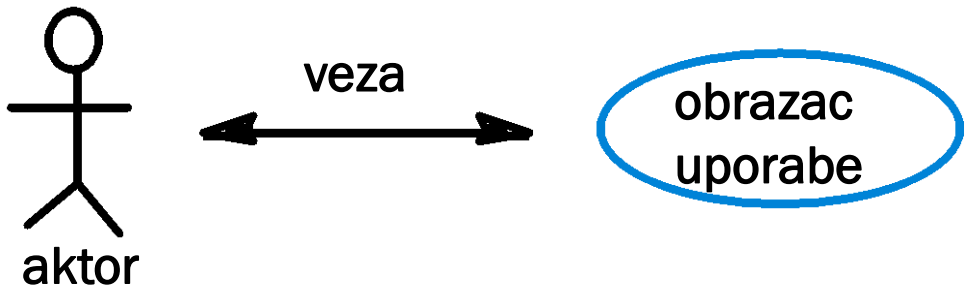
Primjer scenarija

- sustav za praćenje kvalitete zraka

Element	Opis
Redni broj	UC3
Namjena	Praćenje kvalitete zraka
Naziv	Mjerenje kvalitete zraka
Opis	Provodi se mjerenje koncentracije onečišćivača na mjernoj postaji.
Glavni aktor	Web aplikacija
Sudionik	Mjerna postaja
Preduvjeti	-
Početna situacija	Prošlo je potrebno vrijeme (5 min) od zadnjeg mjerenja.
Opis osnovnog tijeka	1. Web aplikacija pošalje upit mjernoj postaji. 2. Mjerna postaja kao odgovor vraća očitanje sa svojih senzora. 3. Aplikacija prima odgovor od mjerne postaje i bilježi datum, vrijeme i vrijednost očitanja. 4. Ako je prošlo 30 min od zadnjeg upisa u bazu podataka, mjerna postaja šalje zabilježene vrijednosti u bazu podataka (UC3.1).
Paralelne aktivnosti	Nema paralelnih aktivnosti
Opis mogućih odstupanja (greške u sustavu)	3.b) Izmjerena vrijednost prelazi graničnu i sustav izdaje upozorenje (UC3.2). 3.b) Aplikacija ne može očitati podatak sa nekog od senzora na mjernoj postaji i šalje upozorenje SMS porukom dežurnom djelatniku svakih 60 min do isključenja pokvarenog senzora (UC3.3).
Završetak scenarija	Podaci zabilježeni u bazi podataka

Obrasci uporabe

- *engl. use cases*
- Obrasci uporabe predstavljaju tehniku preuzetu iz UML norme. *Opisuju se aktori u interakciji s uslugama sustava*
- Skup obrazaca uporabe opisuje **sve moguće interakcije sustava**
- Uz obrasce uporabe, dodatno se mogu koristiti i drugi dijagrami sekvenci za detaljan opis tijeka događaja
- Tri temeljna elementa u modelima obrazaca uporabe su: **obrasci uporabe**, **aktori** i **odnosi** (relacije, *engl. relations*) među njima



Modeliranje obrascima uporabe

- Model obrazaca uporabe je pogled koji ističe ponašanje sustava kako ga vide vanjski korisnici
- Model obrazaca uporabe razdjeljuje funkcionalnost sustava u
  - transakcije (“obrasce uporabe”)
  - razumljive korisnicima (”aktorima”)
- Pogodno za modeliranje:
  - korisničkih zahtjeva
  - scenarija ispitivanja sustava (*engl. test scenarios*)

Specificiranje dinamičkih interakcija

- Metoda izlučivanja zahtjeva modeliranjem dinamičkih interakcija u sustavu
- Modeliranje ponašanja sustava (engl. *behavior modeling*)
- Služi za detaljniji razvoj i prikaz scenarija u izlučivanju, analizi i dokumentiranju zahtjeva:
  - obrasci uporabe identificiraju individualne interakcije u sustavu
  - dijagrami interakcije pokazuju aktore/objekte sustava involvirane u interakciji
- Osnovni tipovi dijagrama interakcija:
  - **sekvencijski**
  - **komunikacijski** (kolaboracijski)

Interakcije

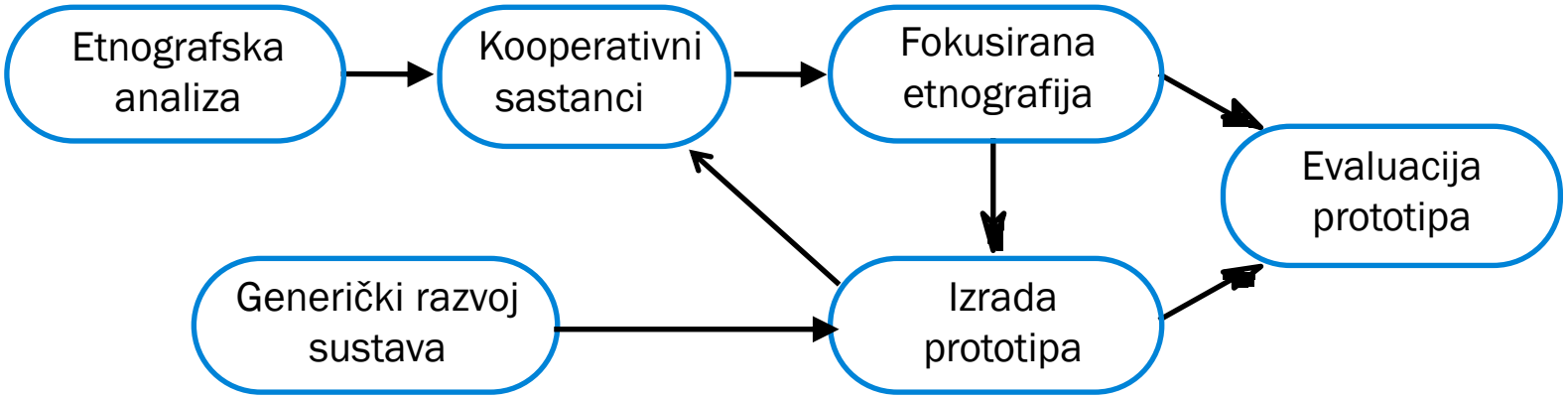
- Interakcija prikazuje komunikaciju elemenata sustava
  - npr. u objektno usmjerenoj arhitekturi:
    - pozivi metoda
    - stvaranje objekata
    - uništavanje objekata
  - komunikacije su djelomično uređene u vremenu
- Modeliranje interakcije omogućava
  - specificiranje međudjelovanje između elemenata sustava
  - olakšava identifikaciju sučelja
  - utvrđuje se potreba za raspodjelom zahtjeva

Društveni i organizacijski čimbenici

- Programski proizvodi se uvijek koriste u društvenom i organizacijskom kontekstu.
  - to uvelike utječe, a ponekad i dominira nad zahtjevima sustava
- Društveni i organizacijski čimbenici nisu jedinstven pogled, već utječu na sve poglede
  - oblikovanje programske potpore mora to uvažiti, ali trenutno ne postoji sistematski postupak kako se to može uključiti u analizu zahtjeva.
- Ne postoje jednostavni modeli za opis obavljanja nekog posla
  - ljudima je često teško precizno opisati što rade
  - postojeći modeli zasnovani su na prošlim, a ne na obrascima ponašanja na novom poslu
  - djelomično se tome može doskočiti izradom prototipa te praćenjem rada na njemu
- Za razumijevanje procesa korisna su znanja iz područja društvenih znanosti

Etnografija

- Tehnika opažanja koja podrazumijeva dolazak jednog ili više ljudi iz razvojnog tima u tvrtku gdje će se sustav primjenjivati i uključivanje tih inženjera (tzv. etnografa) u svakodnevne aktivnosti u tom okruženju kako bi ustanovili **kako ljudi stvarno rade** te tome prilagodili programski proizvod
- Fokusirana etnografija (etnografija + prototip)



Izvor: Sommerville, I., Software engineering

Analiza zahtjeva

- Provjere zahtjeva po kriterijima
  - neophodnosti zahtjeva
  - konzistentnosti i kompletnosti
  - mogućnosti ostvarenja
- Cilj analize zahtjeva je utvrđivanje problema, nekompletnosti, i nejednoznačnosti u izlučenim zahtjevima
  - dionici ih razrješavaju pregovorima
- Problemi:
  - dionici ne znaju što stvarno žele
  - dionici izražavaju zahtjeve na različite, njima specifične načine
  - različiti dionici mogu imati konfliktne zahtjeve
  - organizacijski i politički faktori mogu utjecati na zahtjeve
  - zahtjevi se mijenjaju za vrijeme procesa analize
  - pojavljuju se novi dionici uz promjenu poslovnog okruženja

Profiliranje dionika

- Za razumijevanje trenutnog stanja procesa potrebno odrediti važnost dionika u pojedinim dijelovima sustava
  - velika
  - srednja
  - mala
- Odrediti vještine i sposobnosti
- Dokumentirati ponašanje i stavove
- Olakšava razumijevanje sadašnjeg i željenog stanja
- Olakšava grupiranje i postavljanje prioriteta zahtjeva

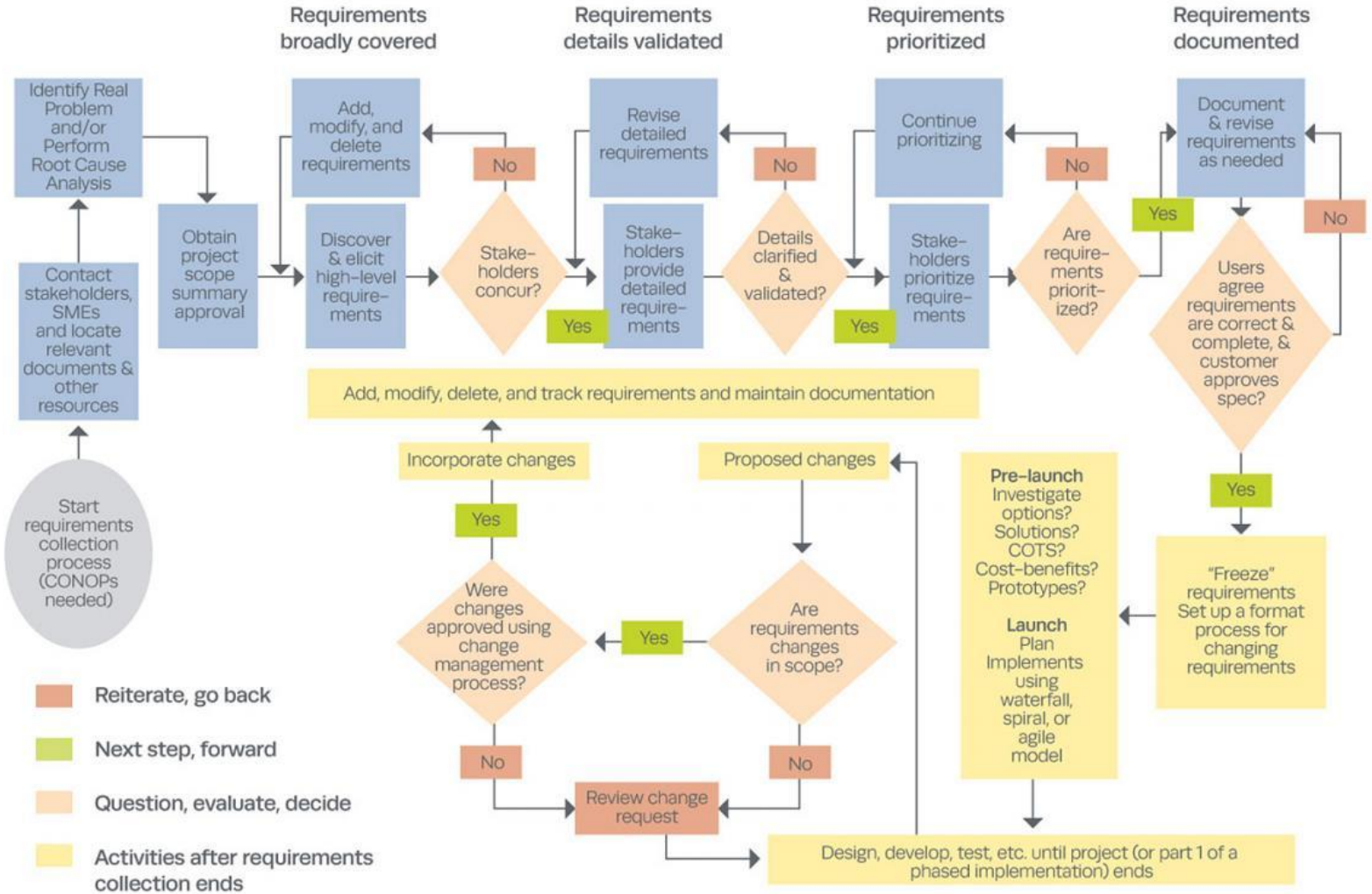
Konflikti i utvrđivanje prioriteta

- Utvrđivanje interakcije zahtjeva pomaže pronalaženju problema, obično se analiziraju tablicom
- **Zahtjevi visokog prioriteta** (*engl. core requirements*)
  - obavezno razmotriti pri analizi, oblikovanju i implementaciji
  - neophodno demonstrirati klijentu pri preuzimanju
- **Srednji prioritet** (*engl. optional requirements*)
  - obavezno razmotriti pri analizi i oblikovanju
  - uobičajeno se implementira u drugim iteracijama projekta
- **Nizak prioritet** (*engl. fancy requirements*)
  - analizira se u obliku naprednih mogućnosti
  - pogodno za prikaz mogućnosti budućeg razvoja

Kako doći do vrijednosti prioriteta?

Naputak za pisanje zahtjeva

- Jedna rečenica sa subjektom i radnjom/stanjem
  - rečenica koja uključuje subjekt, sposobnost (izražena pomoću glagola), a može uključivati kriterij koji je kvantitativni ili kvalitativni.
- Eksplicitna uporaba specifičnih riječi
  - npr. Sustav mora imati mogućnost ...
- Izbjegavanje nejednoznačnih riječi
  - nekoliko, brzo, često, pogodno, ...
- Ne kombinirati više zahtjeva
  - sustav očitava, prenosi i pohranjuje podatke na odgovarajuće stranice
- Ispravna gramatika i pravopis
- Biti jasan i sažet od vitalne je važnosti za uspjeh projekta
- Što jednostavnije na početku projekta, i održavajte jednostavnost jer će se gotovo sigurno mijenjati


Requirements broadly covered
Requirements details validated
Requirements prioritized
Requirements documented
Identify Real Problem and/or Perform Root Cause Analysis
Contact stakeholders, SMEs and locate relevant documents & other resources
Start requirements collection process (CONOPs needed)
Obtain project scope summary approval
Discover & elicit high-level require-ments
Add, modify, and delete requirements
No
Stake-holders concur?
Yes
Stake-holders provide detailed require-ments
Revise detailed requirements
No
Details clarified & validated?
Yes
Stake-holders prioritize require-ments
Continue prioritizing
No
Are require-ments priorit-ized?
Yes
Document & revise requirements as needed
No
Users agree requirements are correct & complete, & customer approves spec?
Yes
"Freeze" requirements Set up a format process for changing requirements
Add, modify, delete, and track requirements and maintain documentation
Incorporate changes
Yes
Were changes approved using change management process?
Yes
No
Proposed changes
Are requirements changes in scope?
No
Review change request
Pre-launch
Investigate options?
Solutions?
COTS?
Cost-benefits?
Prototypes?
Launch
Plan
Implements using waterfall, spiral, or agile model
Design, develop, test, etc. until project (or part 1 of a phased implementation) ends
Reiterate, go back
Next step, forward
Question, evaluate, decide
Activities after requirements collection ends

Validacija zahtjeva

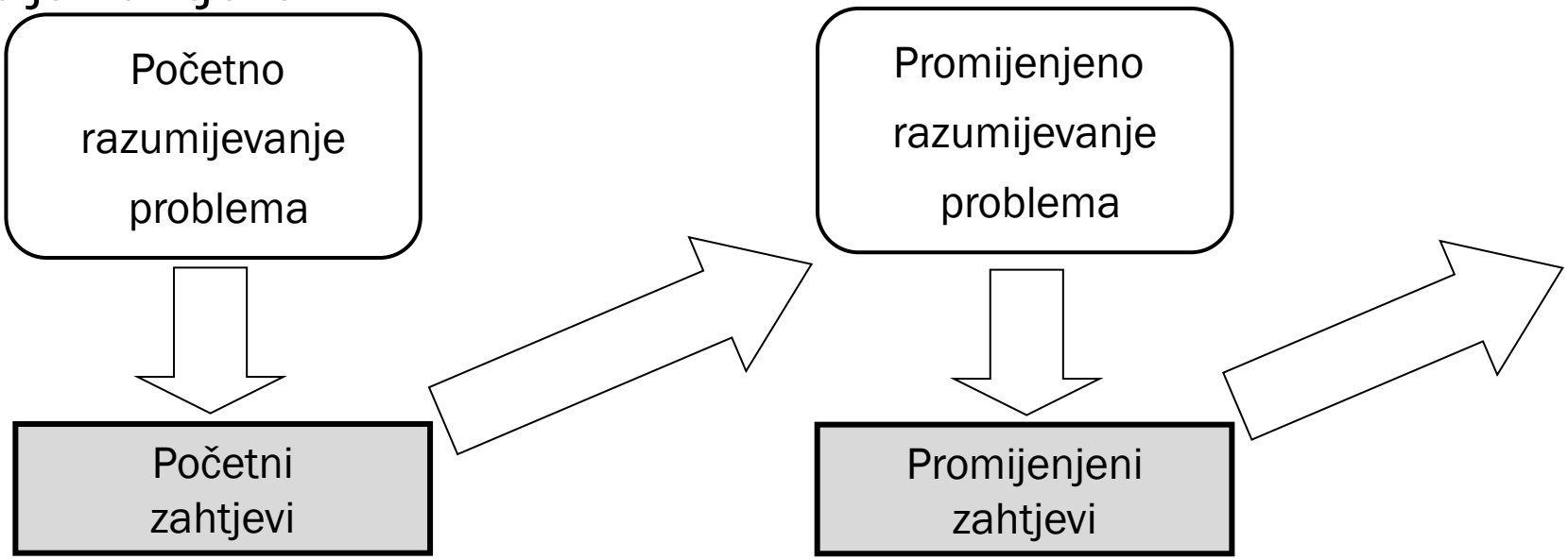
- Cilj validacije je pokazati da dokument zahtjeva predstavlja prihvatljiv opis sustava koji naručitelj doista želi
  - naknadno ispravljanje pogreške u zahtjevima može biti višestruko skuplje od ispravljanja pogrešaka u implementaciji
- Tehnike validacije
  - **recenzija zahtjeva**
    - sistematska ručna analiza
  - **izrada prototipa**
    - provjera na izvedenom sustavu
  - **generiranje ispitnih slučaja**
    - razvoj ispitnih sekvenci za provjeru zahtjeva
- Rezultati validacije
  - lista problema (pojašnjenja, nedostaci, konflikti, neostvarivost)
  - lista utvrđenih akcija za razrješavanje problema

Elementi provjere zahtjeva

- Razumljivost
  - Je li dokument jasno napisan?
- Kompletnost
  - uključuje li dokument sve funkcionalnosti koje je korisnik tražio?
- Konzistencija
  - postoji li konflikt u zahtjevima?
- Valjanost
  - osigurava li sustav funkcije koje podupiru potrebe korisnika?
- Ostvarivost
  - mogu li se sve funkcionalnosti implementirati uz danu tehnologiju i proračun?
- Provjerljivost
  - mogu li se svi zahtjevi provjeriti?
- Sljedivost
  - je li naveden izvor dokumenta iz kojeg dolazi zahtjev i razlozi uvrštavanja zahtjeva?
- Adaptabilnost
  - mogu li se zahtjevi mijenjati bez velikog utjecaja na druge zahtjeve?

Upravljanje promjenama u zahtjevima

- Upravljanje ili rukovanje zahtjevima (engl. *requirements management*)
- Promjene nastupaju zbog promijenjenog modela poslovanja, boljeg razumijevanja procesa tijekom razvoja ili konfliktnih zahtjeva u različitim pogledima
- Evolucija zahtjeva:

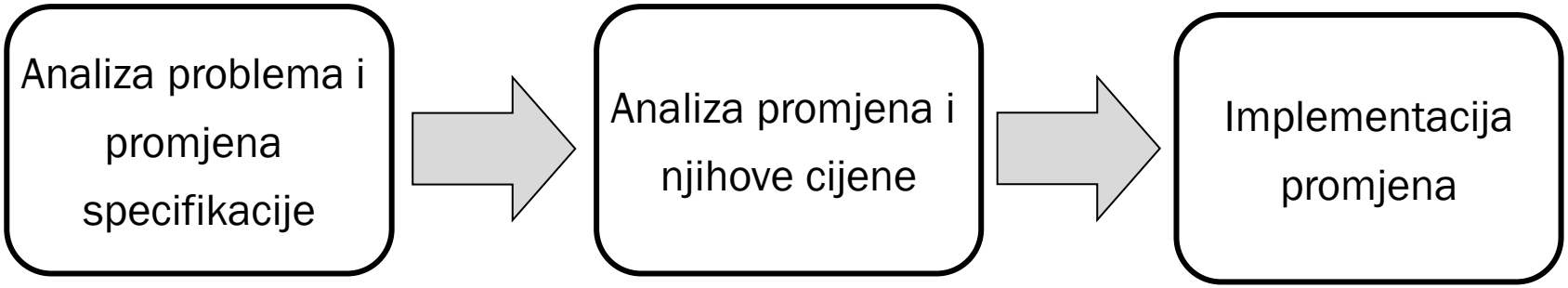


Klasifikacija promjena zahtjeva

- **Okolinom promijenjeni zahtjevi**
  - promjena zbog promjene okoline u kojoj organizacija posluje
    - npr. bolnica mijenja financijski model pokrivanja usluga
- **Novonastali zahtjevi**
  - zahtjevi koji se pojavljuju kako kupac sve bolje razumije sustav koji se oblikuje
- **Posljedični zahtjevi**
  - zahtjevi koji nastaju nakon uvođenja sustav u eksploataciju, a rezultat su promjena procesa rada u organizaciji nastalih upravo uvođenjem novoga sustava
- **Zahtjevi kompatibilnosti**
  - zahtjevi koji ovise o procesima drugih sustava u organizaciji; ako se ti sustavi mijenjaju to traži promjenu zahtjeva u novouvedeni sustav

Upravljanje procesom promjena

- Planiranje promjena
  - identifikacija zahtjeva (individualno identificiranje zahtjeva)
- Upravljanje procesom promjena
  - proces koji slijedi kada se utvrdi potreba za promjenom



- Sljedivost
  - koje informacije su potrebne za povezivanje zahtjeva
- Izbor CASE alata
  - automatiziranje skladištenja, unošenja promjena i povezivanje dokumenata

Zaključak

- Proces inženjerstva zahtjeva uključuje:
  - studiju izvedivosti, izlučivanje zahtjeva i analizu;
  - specifikaciju i validaciju zahtjeva te rukovanje (upravljanje) zahtjevima.
- Izlučivanje i analiza zahtjeva je **iterativan proces** koji uključuje razumijevanje domene primjene, prikupljanje, klasifikaciju, strukturiranje zahtjeva i određivanje prioriteta
  - sustavi imaju više dionika s različitim zahtjevima
  - društveni i organizacijski čimbenici utječu na zahtjeve sustava
- Validacija zahtjeva bavi se valjanošću, konzistentnošću, kompletnošću, realizmom u izvedbi i provjerljivošću zahtjeva
- Rukovanje zahtjevima uključuje **planiranje** i **upravljanje promjenama** (*engl. requirements management*) u zahtjevima