Inženjerstvo zahteva

Principi softverskog inženjerstva, *Elektrotehnički fakultet Univerziteta u Beogradu*

Terminologija: zahtev / zainteresovana strana

Definicija 1: Zahtev (*requirement*) *izvor: [IEEE Std. 610.12-1990]*

- (1) uslov ili mogućnost potrebna korisniku da reši problem ili postigne neki cilj.
- (2) uslov ili sposobnost koju mora da ispuni ili poseduje sistem ili komponenta sistema da bi zadovoljila ugovor, standard, specifikaciju ili neki drugi formalno propisani dokument.
- (3) dokumentovani prikaz uslova ili sposobnosti iz (1) ili (2).

Definicija 2: Zainteresovana strana (*stakeholder*)

- Zainteresovana strana je osoba ili organizacija koja ima (neposredan ili posredan) uticaj na zahteve sistema.

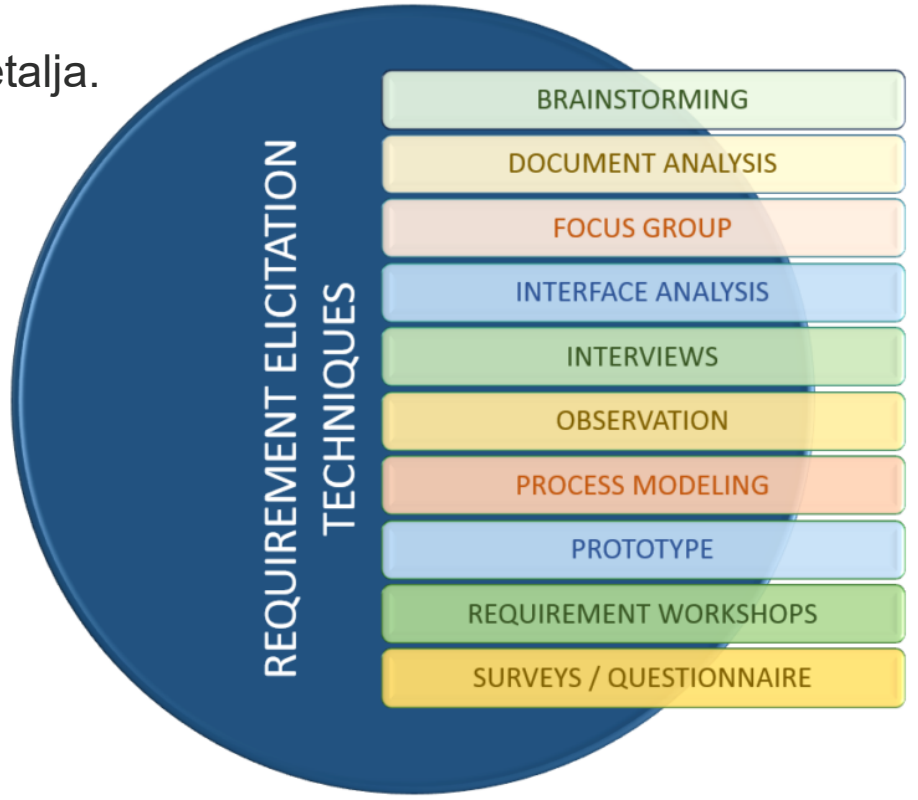
Terminologija: inženjerstvo zahteva

Definicija 3: Inženjerstvo zahtetva (*requirements engineering*)

- Inženjerstvo zahteva je sistematski i disciplinovan pristup specifikaciji i upravljanju zahtevima sa sledećim ciljevima:
  - Poznajući relevantne zahteve, postizanje konsenzusa među zainteresovanim stranama o ovim zahtevima, njihovo dokumentovanje prema datim standardima i sistematsko upravljanje zahtevima.
  - Razumevanje i dokumentovanje želja i potreba zainteresovanih strana, potom specifikovanje i upravljanje zahtevima na način da se smanji rizik od izrade sistema koji ne ispunjava želje i potrebe zainteresovanih strana.

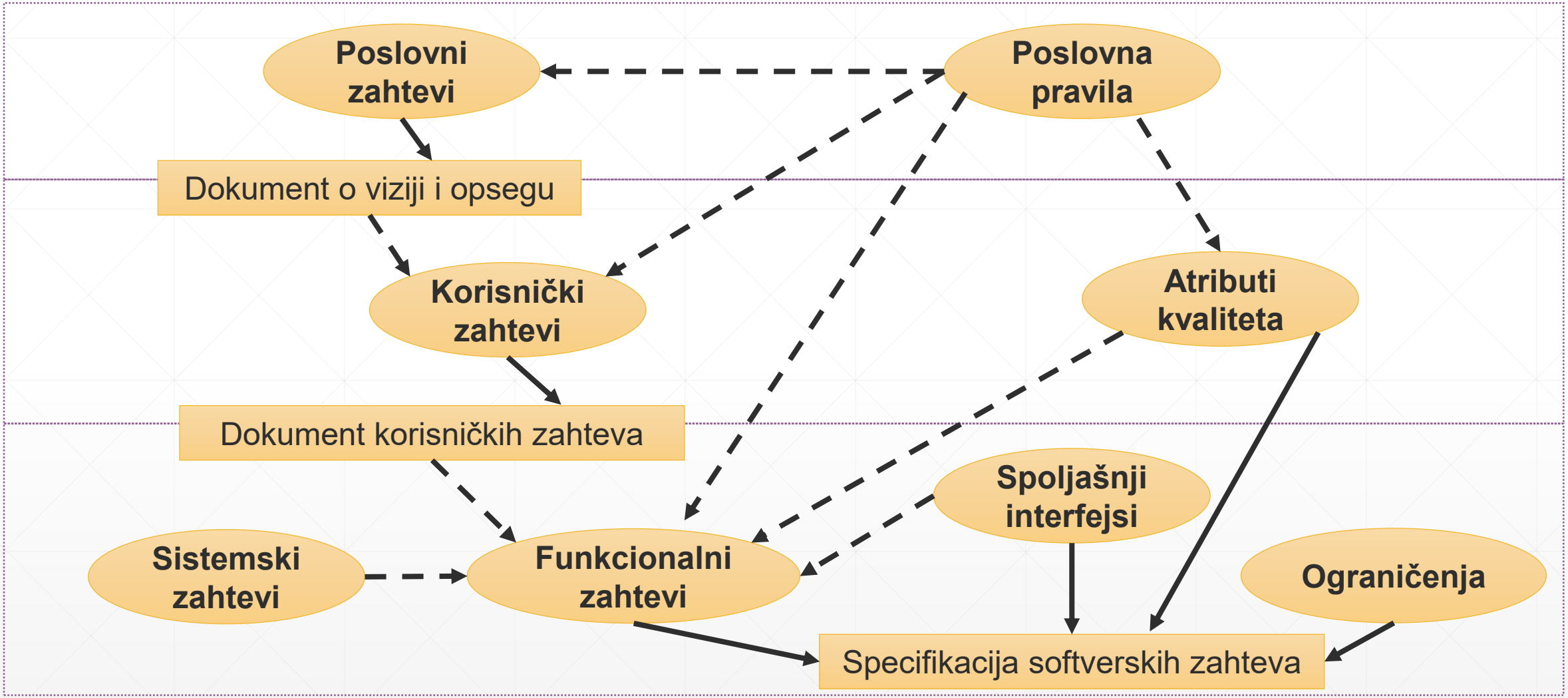
Glavne aktivnosti inženjerstva zahteva

- Elicitacija (prikupljanje): Različite tehnike za dobijanje zahteva od zainteresovanih strana i drugih izvora, i za razradu zahteva u više detalja.
- Dokumentovanje: Izneti zahtevi se adekvatno opisuju.
  Koriste se različite tehnike, od govornog jezika do konceptualnog modelovanja.
- Validacija i dogovaranje: U cilju garantovanja ispunjenosti prethodno definisanih kriterijuma kvaliteta, dokumentovani zahtevi moraju biti provereni i dogovoreni što ranije.
- Upravljanje: Upravljanje zahtevima je ortogonalno na sve druge aktivnosti i obuhvata sve mere koje su neophodne radi strukturiranja zahteva, njihovog pripremanja, tako da ih mogu koristiti različite uloge, održavanja konzistentnosti zahteva posle promena, kao i da se obezbedi njihovo implementiranje.



Preuzeto sa: https://www.linkedin.com/pulse/top-5-requirements-elicitation-techniques-anar-solutions/

Tipovi zahteva



Tipovi zahteva: Poslovni zahtevi

- **Poslovni zahtevi** opisuju zašto organizacija uvodi sistem tj. koje poslovne prednosti se organizacija nada da će postići. Fokus je na poslovnim ciljevima organizacije ili kupca koji traži sistem.
- Primer: Avio kompanija želi da smanji troškove osoblja aerodroma za 25%.
  Ovaj cilj može da dovede do ideje o izgradnji kioska koji putnici mogu da koriste da se prijave za svoje letove na aerodromu.
- Poslovni zahtevi obično dolaze od finansijskog sponzora projekta, kupca, direktora stvarnih korisnika, marketinga, ili vizionara proizvoda. Obično želimo da opišemo poslovne zahteve u *Dokumentu o viziji i opsegu*.

Tipovi zahteva: Zahtevi korisnika

- **Zahtevi korisnika** opisuju ciljeve ili zadatke koje korisnici moraju biti u stanju da obavljaju sa proizvodom, koji će obezbediti vrednost za nekoga. Domen zahteva korisnika takođe uključuje opise atributa proizvoda ili karakteristike koje su važne za zadovoljstvo korisnika.
- Načini predstavljanja zahteva korisnika uključuju *slučajeve korišćenja*, *korisničke priče*.
  U idealnom slučaju, stvarni predstavnici korisnika će obezbediti ovu informaciju.
- Zahtevi korisnika opisuju šta će korisnik moći da uradi sa sistemom.
- Primer slučaja korišćenja je „prijava za let“ koristeći veb sajt avio kompanije ili kiosk na aerodromu. Napisan kao korisnička priča, isti zahtev može glasiti: „Kao putnik, želim da se prijavim za let tako da mogu da se ukrcam u avion / na svoj let“.
- Važno je zapamtiti da većina projekata ima više klasa korisnika (korisničkih uloga / rola), kao i druge zainteresovane strane čije potrebe takođe moraju biti zabeležene.

Tipovi zahteva: Funkcionalni i nefunkcionalni zahtevi

- **Funkcionalni zahtevi** opisuju ponašanje proizvoda u određenim uslovima.
- Oni opisuju šta programeri moraju implementirati da omoguće korisnicima da obave posao (korisnički zahtev), na taj način zadovoljavajući poslovni zahtev.
- Primer: „Putnik treba da može da odštampa karte za let, za sve segmente leta za koje se prijavio“ ili „Ako korisnik nije naveo želju za posebnim sedištem, sistem za rezervaciju treba da dodeli sedište putniku automatski.“
- Poslovni analitičar dokumentuje funkcionalne zahteve u specifikaciji softverskih zahteva (SRS), koja opisuje u meri koliko je neophodno očekivano ponašanje softverskog sistema.
- SRS se koristi u razvoju, testiranju, osiguranju kvaliteta, upravljanju projektom i srodnim aktivnostima. Ovaj artefakt može da se zove drugačije: poslovni zahtevi, funkcionalna specifikacija, lista zahteva, itd.
- **Nefunkcionalni zahtevi** specifikuju ne ono što sistem treba da radi, nego koliko dobro treba da radi. Tu spadaju zahtevi vezani za proizvod (npr. peformanse, sigurnost), za razvojni proces (na primer programski jezik i standardi) i spoljna ograničenja (sve ostalo).

Tipovi zahteva: Sistemski zahtevi

- **Sistemski zahtevi** opisuju zahteve za proizvod koji se sastoji od više komponenti ili podsistema (ISO / IEC / IEEE 2011). Sistem može biti čist softver ili može uključivati i softverske i hardverske podsisteme. Ljudi i procesi su deo sistema, tako da određene funkcije sistema mogu biti dodeljene ljudima.
- Dobar primer „sistema“ je kompjuter na kasi u supermarketu. Postoji bar kod skener integrisan sa kasom, kao i ručni bar kod skener. Blagajnica ima tastaturu, displej i fioku za novac. Postoje čitači kartica i tastatura za unos PIN koda. Postoje štampači fiskalnog računa, izveštaja kreditne kartice. Ovi hardverski uređaji su svi pod kontrolom softvera.
- Zahtevi za sistem ili proizvod u celini navode poslovnog analitičara da izvede specifične funkcionalnosti koje moraju biti dodeljene jednoj ili drugoj komponenti sistema, kao i interfejse između tih komponenata.

Tipovi zahteva: Poslovna pravila

- **Poslovna pravila** uključuju korporativne politike, vladine propise, industrijske standarde i kompjuterske algoritme.
- Poslovna pravila nisu sama po sebi softverski zahtevi, jer ona imaju egzistenciju izvan granica bilo koje specifične softverske aplikacije. Ona često diktiraju koju funkcionalnost sistem mora da sadrži u skladu sa odgovarajućim pravilim. Ponekad, kao sa korporativnim bezbednosnim pravilima, poslovna pravila se prevode u specifične atribute kvaliteta.
  Dakle, možemo da pratimo genezu pojedinih funkcionalnih i nefunkcionalnih zahteva u određenom poslovnom pravilu (*traceability*).

Kategorije nefunkcionalnih zahteva

Dokument o viziji i opsegu sistema

1. **Business requirements**
   - 1.1 Background
   - 1.2 Business opportunity
   - 1.3 Business objectives
   - 1.4 Success metrics
   - 1.5 Vision statement
   - 1.6 Business risks
   - 1.7 Business assumptions and dependencies
2. **Scope and limitations**
   - 2.1 Major features
   - 2.2 Scope of initial release
   - 2.3 Scope of subsequent releases
   - 2.4 Limitations and exclusions
3. **Business context**
   - 3.1 Stakeholder profiles
   - 3.2 Project priorities
   - 3.3 Deployment considerations

- ***Poslovni zahtevi*** opisuju osnovne prednosti koje će novi sistem pružati svojim sponzorima, kupcima i korisnicima.
- ***Opseg i ograničenja*** - treba da se navede šta rešenje koje se razvija sadrži, a šta ne.
- ***Poslovni kontekst*** - ova sekcija predstavlja profile glavnih zainteresovanih strana, prioriteta za upravljanje projektom, kao i informacija i aktivnosti koje su potrebne da bi se obezbedila efikasno raspoređivanje rešenja u svoje radno okruženje.

Šablon za pisanje izjave o viziji

Za potrebe [ciljna mušterija]

Kome treba [rečenica o potrebi ili poslovnoj šansi]

Proizvod [ime proizvoda]

Koji pripada [kategoriji proizvoda]

Radi [glavne mogućnosti, ključna korist, jak razlog za nabavku ili upotrebu]

Za razliku od [primarna konkurentska alternativa, postojeći sistem, postojeći poslovni proces]

Naš proizvod [izjava o prvenstvenom razlikovanju i prednosti novog proizvoda]

Opseg sistema

- Opis opsega ustanovljava granicu i veze između sistema koji se razvije i svega ostalog.
- Kontekstni dijagrami (*context diagram*) i stabla karakteristika (*feature trees*) su neki od načina vizuelnog reprezentovanja opsega sistema.
- Dijagrami slučajeva korišćenja mogu takođe reprezentovati granice između slučajeva korišćenja sistema i spoljnih aktera.

Kontekstni dijagram

- **Kontekstni dijagram** vizuelno ilustruje granicu sistema. Ona identifikuje spoljne entitete (takozvane terminatore) izvan sistema koji interaguju sa njim na neki način, kao tok podataka, kontrole i materijala između terminatora i sistema.
- **Kontekstni dijagram** je najviši nivo u dijagramu toka podataka razvijen u skladu sa principima strukturne sistem analize (SSA), ali je koristan model i za projekte koji ne koriste SSA.

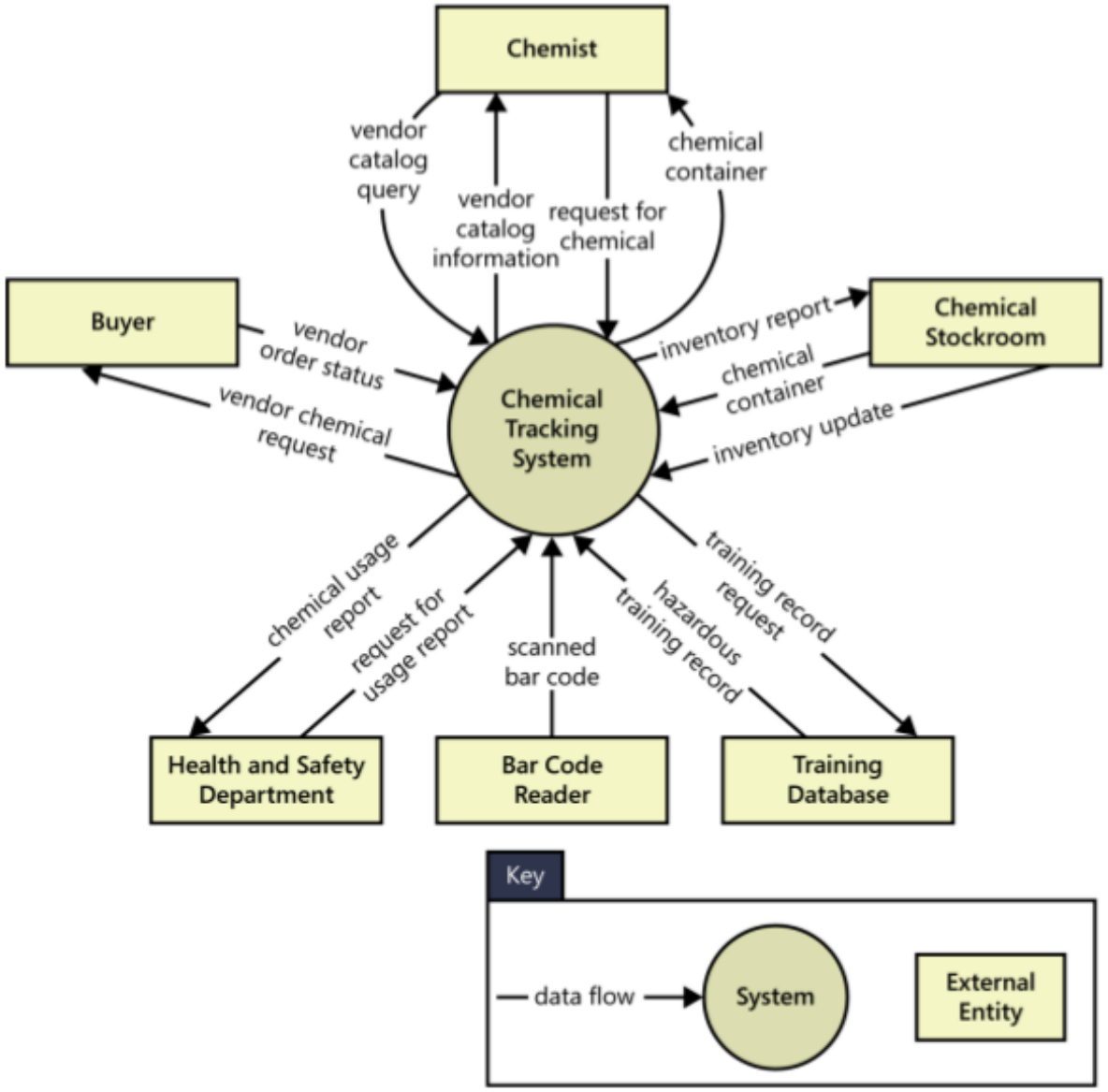
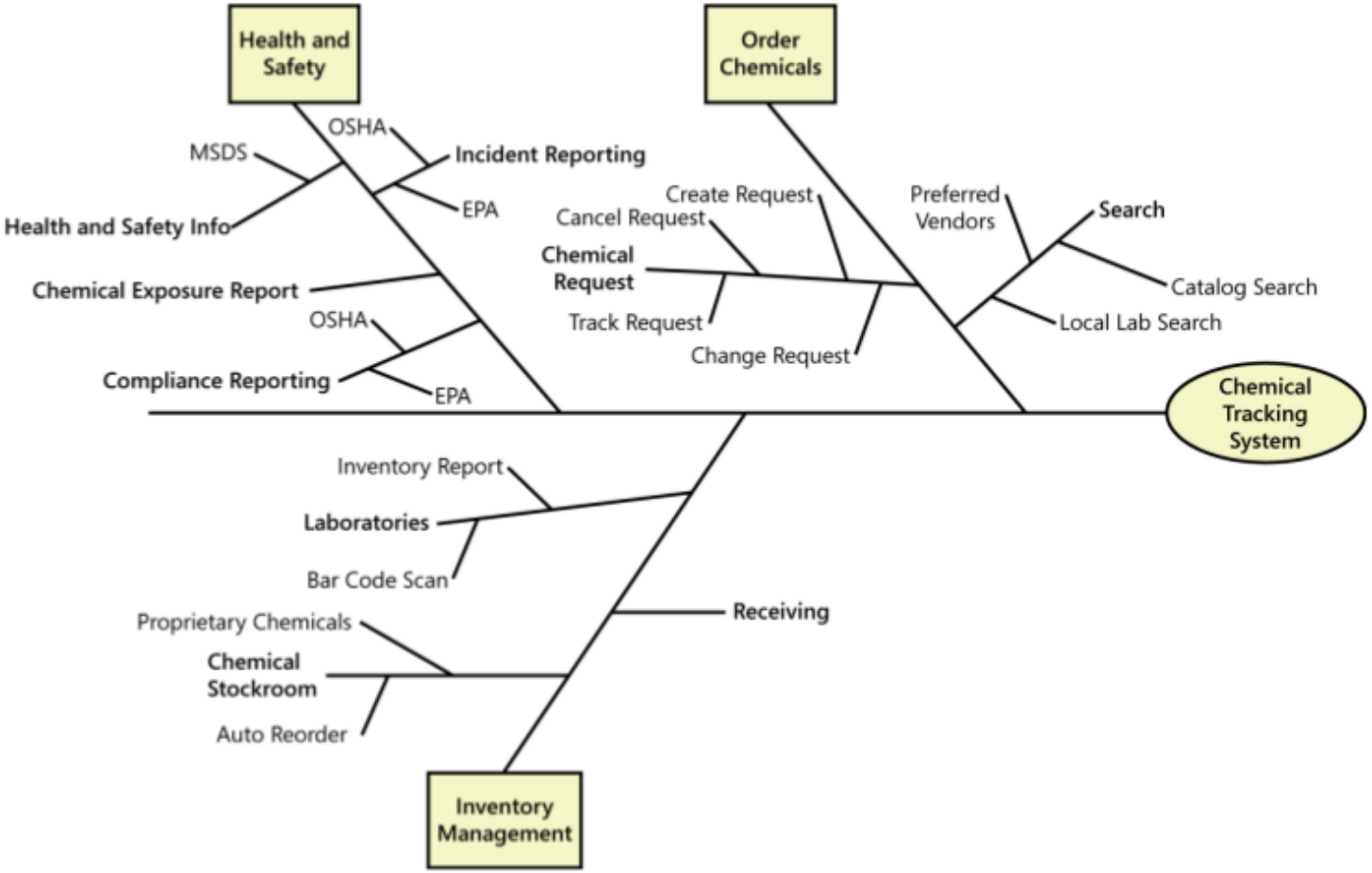


FIGURE 5-6 Partial context diagram for the Chemical Tracking System.

Stablo karakteristika

- **Stablo karakteristika**
  - vizuelni prikaz karakteristika proizvoda organizovanih u logičke grupe,
  - hijerarhijski dekomponuje svaku funkciju u dalje nivoe detalja.



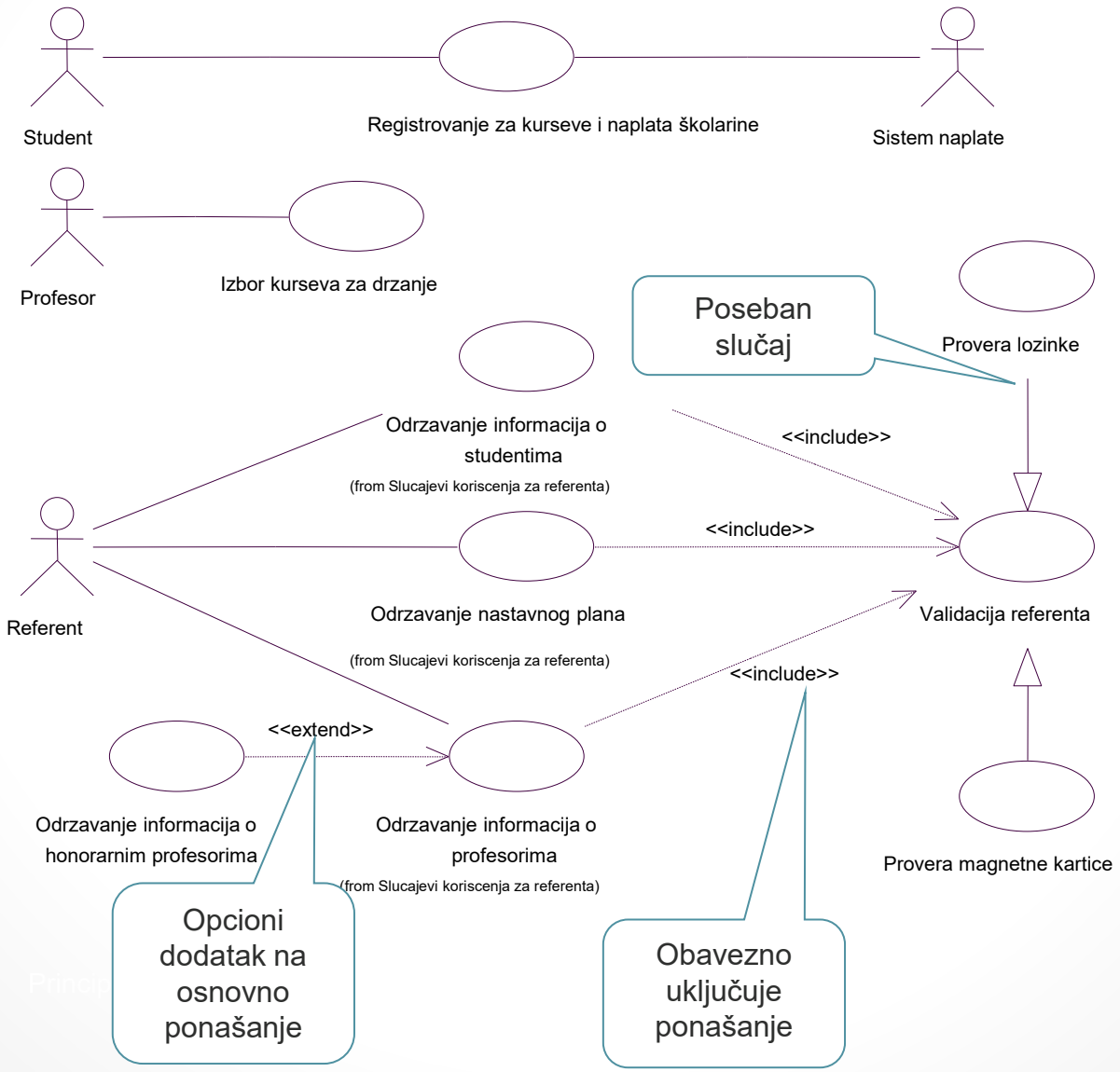
- Stablo funkcija daje sažet prikaz svih funkcija planiranih za projekat, što ga čini idealnim modelom da se prikaže opseg sistema.
- Stablo karakteristika može da prikaže do tri nivoa funkcija, obično nazvanih nivo 1 (L1), nivo 2 (L2) i nivo 3 (L3). L2 karakteristike su potfunkcija L1 funkcija, a L3 karakteristike su potfunkcija L2 funkcija.

Predstavljanje korisničkih zahteva

- **Zahtevi korisnika** leže između poslovnih zahteva koji postavljaju ciljeve za projekat i funkcionalnih zahteva koji opisuju ono što programeri moraju implementirati.
- Dve najčešće korišćene tehnike za istraživanje zahteva korisnika:
  - slučajevi korišćenja,
  - korisničke priče.
- UML je standardizovani vizuelni jezik za modelovanje softvera.

- **Slučaj upotrebe** je opis skupa sekvenci akcija, koje sistem obavlja da bi proizveo vidljiv rezultat od vrednosti za pojedinog **aktera**.
- Sekvenca akcija reprezentuje interakciju aktera sa sistemom (jedan moguć scenario)
- Slučaj upotrebe specificira šta sistem (podsistem, klasa, interfejs) radi, a ne kako radi.
- **Akter** predstavlja neki skup (korisničkih) uloga.
- Akter može biti čovek (korisnik) ili neki sistem sa kojim modelirani sistem interaguje A.
- Aplikativni sistem interaguje sa jednim ili više spoljašnjih aktera.

Primer dijagrama slučajeva upotrebe



Opis zahteva

- Student može da izabere određen broj kurseva koje želi da sluša.
- Školarina zavisi od broja i trajanja kurseva.
- Profesor daje ponudu za kurs koji namerava da drži u semestru.
- Referent održava informacije o studentima, profesorima i nastavnom planu.
- Neki profesori nisu redovno zaposleni, već su honorarni.
- Pre nego što se referentu dozvoli rad mora se izvršiti provera prava pristupa.
- Provera prava pristupa se može temeljiti na unosu lozinke ili na magnetnoj kartici.

Proces modelovanja sistema

1) Ustanovljavanje zajedničkog rečnika pojmova iz domena problema na bazi korisničkih zahteva
2) Nalaženje aktera i slučajeva upotrebe
3) Opis *use case* modela - grupisanje povezanih aktera i slučajeva upotrebe u pakete
4) Opis slučajeva upotrebe – detaljna specifikacija
5) Strukturiranje *use case* modela - određivanje relacija između slučajeva upotrebe
6) Pregled modela - verifikacija kompletnosti i konzistentnosti modela

Tekstualna specifikacija slučaja upotrebe i primer

1. **Use Case Name**
   1.1 Brief Description
2. **Flow of Events**
   2.1 Basic Flow
   2.2 Alternative Flows
      2.2.1 < First Alternative Flow >
      2.2.2 < Second Alternative Flow >
3. **Special Requirements**
   3.1< First special requirement > relevantni nefunkcionalni zahtevi
4. **Pre-Conditions**
   4.1 < Pre-condition One > stanje sistema da bi se izvršio
5. **Post-Conditions**
   5.1 < Post-condition One > stanje sistema neposredno posle primene
6. **Extension Points**
   6.1 <name of extension point>

Use Case: Manage Backpack
ID: UCS
Actors: Player
Preconditions: 1. Game play is paused 2. Backpack contents are visible
Flow of Events: 1. Use case starts when player selects a backpack item 2. If player selects “remove” item” 2.1 System deletes selected item from backpack display 3. If player selects “add item” 3.1 Systems places new item in backpack display below the selected item
Postconditions: 1. Backpack display is updated

Odnos slučajeva upotrebe i funkcionalnih zahteva

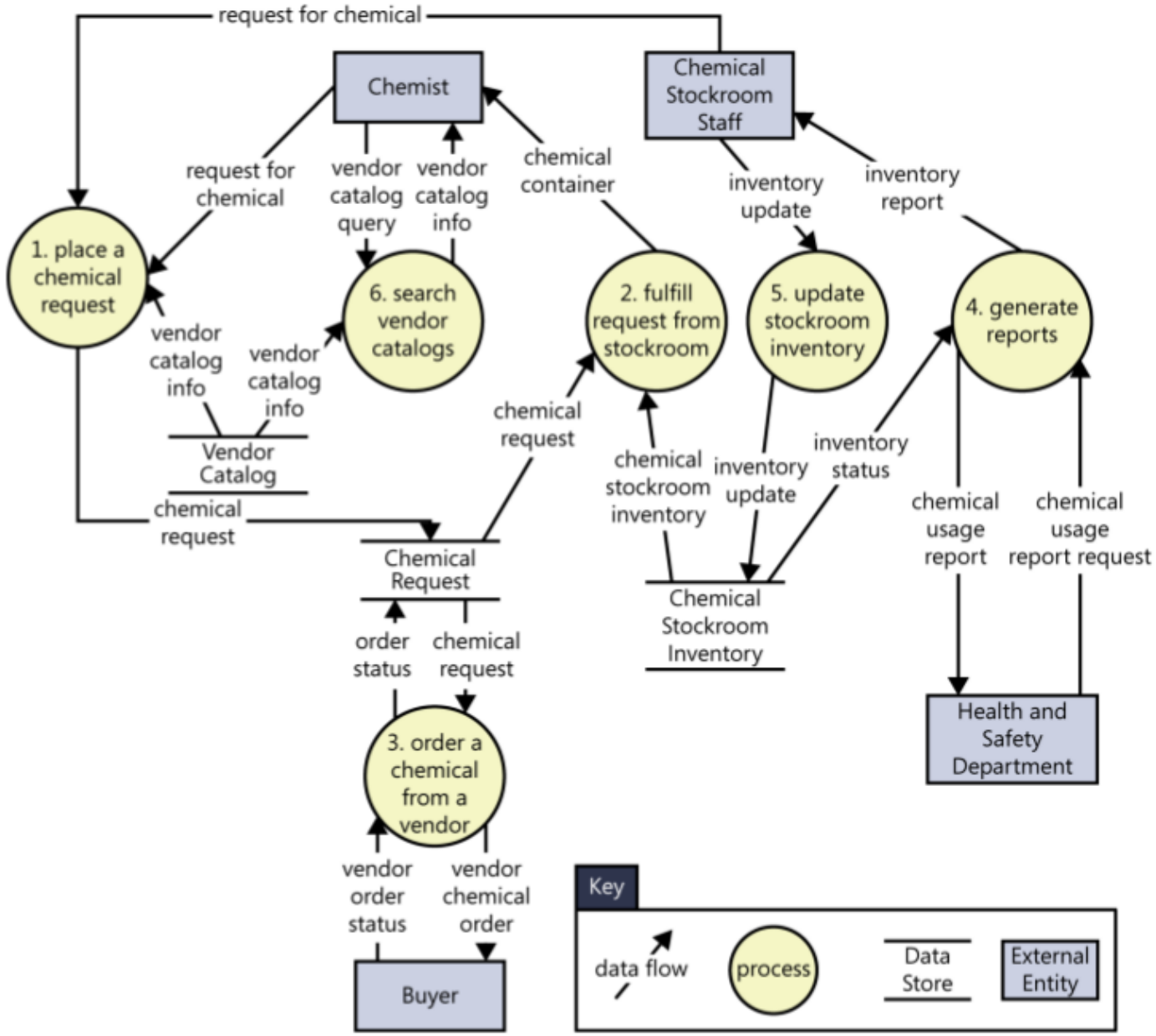
- Programeri ne implementiraju poslovne zahteve ili zahteve korisnika.
  Oni implementiraju funkcionalne zahteve, tj. određene delove ponašanja sistema.
- Slučajevi upotrebe opisuju perspektivu korisnika, gledajući spolja vidljivo ponašanje sistema.
  Mnogi funkcionalni zahtevi proizilaze neposredno iz koraka interakcije aktera i sistema .
- Neki su očigledni, kao što je „sistem će dodeliti jedinstveni redni broj za svaki zahtev“.
- Nema smisla duplirati ih na drugom mestu, ako su jasni iz slučaja korišćenja.
- Ostali funkcionalni zahtevi se ne pojavljuju u opisu slučaja korišćenja. Na primer, uobičajeni način dokumentovanja slučajeva upotrebe ne precizira šta sistem treba da uradi ako uslov nije zadovoljen. Ovo je primer kako slučajevi upotrebe često ne pružaju sve potrebne informacije za programera da zna šta treba da se implementira. Takve situacije pokriva SRS.

Ostale metode vizuelnog predstavljanja korisničkih zahteva

- Dijagrami toka podataka (eng. *Data Flow Diagrams*, skr. *DFD*)
- Swimlane dijagrami
- Dijagrami stanja (eng. *State transition diagrams*)
- Mape dijaloga
- Tabele i stabla odlučivanja
- Modelovanje zahteva za podatke

Dijagrami toka podataka

- Dijagram toka podataka je osnovni alat za strukturnu analizu.
- DFD identifikuje transformacione procese sistema, kolekcije (skladišta) podataka ili fizičkih materijala kojima sistem manipuliše i tokova podataka ili materijala između procesa, skladišta, i spoljnog sveta.
- Modelovanje toka podataka predstavlja funkcionalni pristup dekompoziciji u analizi sistema, razbijanjem složenih problema u detaljnije nivoe. Ovo dobro funkcioniše za informacione sisteme.
- Ranije predstavljeni kontekstni dijagram precizira se DFD-om nivoa 0.



DFD - pravila

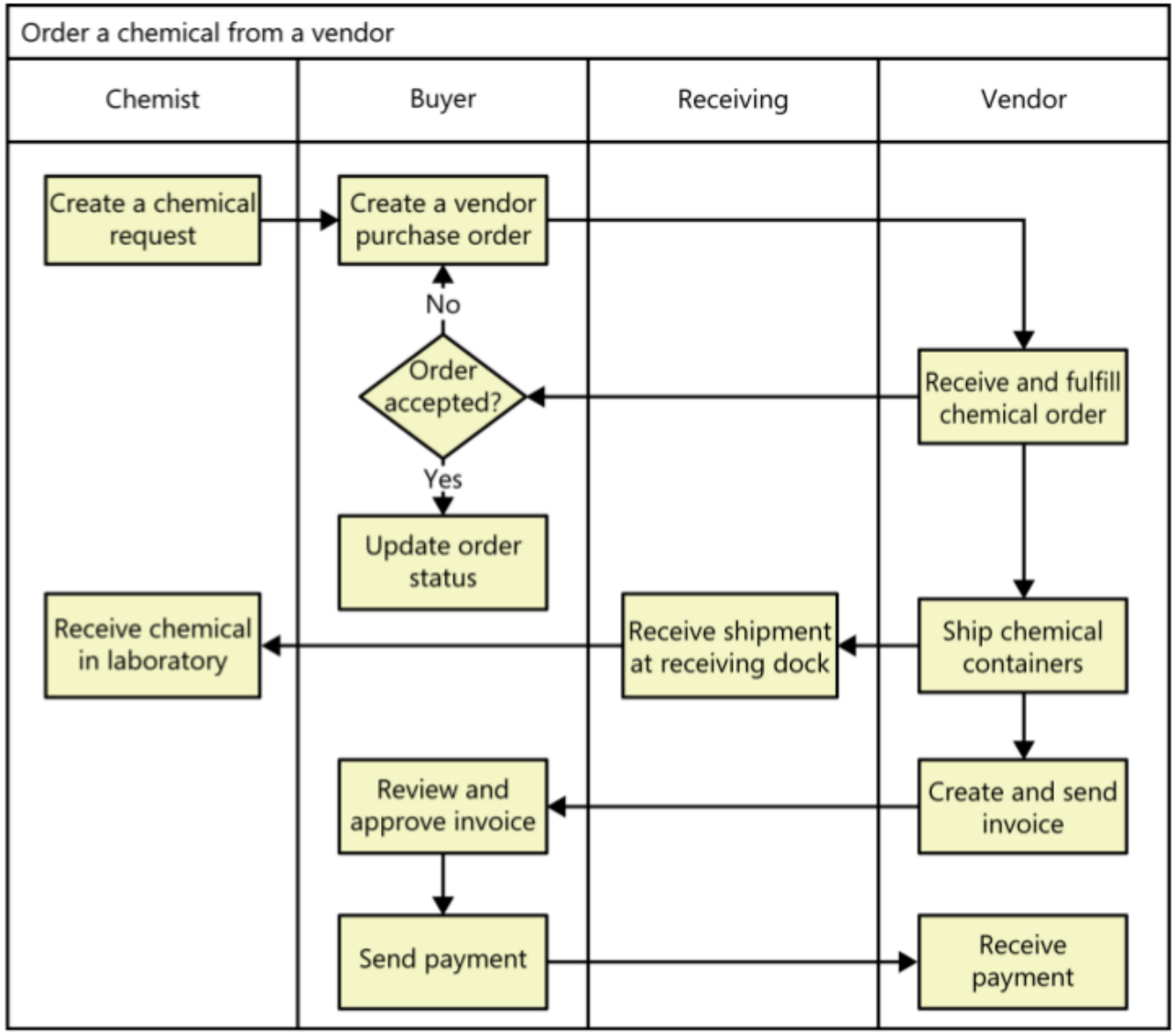
- Svaki proces koji se pojavljuje kao poseban oval na dijagramu nivoa 0 može se dodatno proširiti u poseban DFD da otkrije više detalja o njegovom funkcionisanju.
- Analitičar nastavlja progresivno raščlanjivanje do dijagrama na najnižem nivou koji sadrže samo primitivne procese koji se mogu jasno predstaviti tekstualno, pseudokodom, *swimlane* dijagramom, ili dijagramom aktivnosti.
- Funkcionalni zahtevi će precizno definisati šta se dešava u svakom primitivnom procesa.
- Svaki nivo DFD mora biti uravnotežen i u skladu sa nivoom iznad njega, tako da se svi ulazni i izlazni tokovi na dijagramu nižeg nivoa moraju poklopiti sa tokovima roditeljskog dijagrama. Kompleksne strukture podataka u dijagramu na visokom nivou mogu se podeliti na sastavne elemente, što se definiše u rečniku podataka.

DFD - konvencije pri crtanju

- Procesi komuniciraju kroz skladišta podataka, ne direktno.
- Podaci ne mogu direktno iz jednog skladišta u drugo, ili između skladišta i spoljnog entiteta, bez da prođu kroz proces.
- Ne pokušavati specificirati redosled procesiranja na DFD-u.
- Imenovanje procesa: glagolska imenica+objekat (npr. generisanje izveštaja).
  Koristiti nazive iz poslovnog domena, smislena mušterijama.
- Jedinstvena i hijerarhijska numeracija procesa (na nivou 0, procesi 1, 2,...).
  Ako se za proc.1 napravi poseban DFD, onda na njemu proc. 1.1, 1.2, itd.
- Ne prikazivati više od 8-10 procesa na jednom dijagramu zbog lakšeg razumevanja (ako ima više, grupisati u asptraktnije procese).
- Procesi normalno imaju i ulaz i izlaz, sumnjivi su oni koji imaju samo jedno od toga.

Swimlane dijagrami

- ***Swimlane* dijagram** pruža način za predstavljanje sastavnih koraka poslovnog procesa ili operacija predloženog softverskog sistema.
- On je varijacija dijagrama toka, podeljen na vizuelne potkomponente zvane staze ili trake. Staze mogu predstavljati različite sisteme ili aktere koji izvršavaju korake u procesu.
- Svimlane dijagram se najčešće koristi da pokaže poslovne procese, tokove posla ili interakcije sistema i korisnika. On je sličan UML dijagramu aktivnosti. Svimlane dijagram se ponekad naziva unakrsni funkcionalni (cross-functional) dijagram.



Swimlane dijagram - struktura

- Sastavni delovi swimlane dijagrama:
- Procesni koraci, prikazani kao pravougaonici.
- Prelazi između procesnih koraka, prikazani kao strelice koje povezuju parove pravougaonika.
- Odluke, prikazane kao rombovi sa više izlaznih grana.
  Grane su označene konkretnim ishodom odluke.
- Staze za podelu procesa, prikazane kao horizontalne ili vertikalne linije na dijagramu.
  U stazama su najčešće pojedine uloge, odeljenja, ili sistemi.
  One pokazuju ko ili šta izvršava korake u datoj traci.

Dijagram stanja i prelaza

State transition diagram (STD)
sadrži tri vrste elemenata :

- Moguća stanja sistema, prikazana pravougaonicima.
- Dozvoljene promene stanja ili prelaze, prikazane kao strelice koje povezuju parove pravougaonika.
- Događaje ili uslove koji uzrokuju svaku tranziciju, prikazane tekstom na svakoj tranziciji. Tekst može identifikovati kako ulazni događaj tako i odgovarajući odgovor sistema.
- STD za objekat koji prolazi kroz definisani životni ciklus će imati jedno ili više završnih (terminacionih) stanja. Završna stanja imaju samo dolazne tranzicije, ne i odlazne.

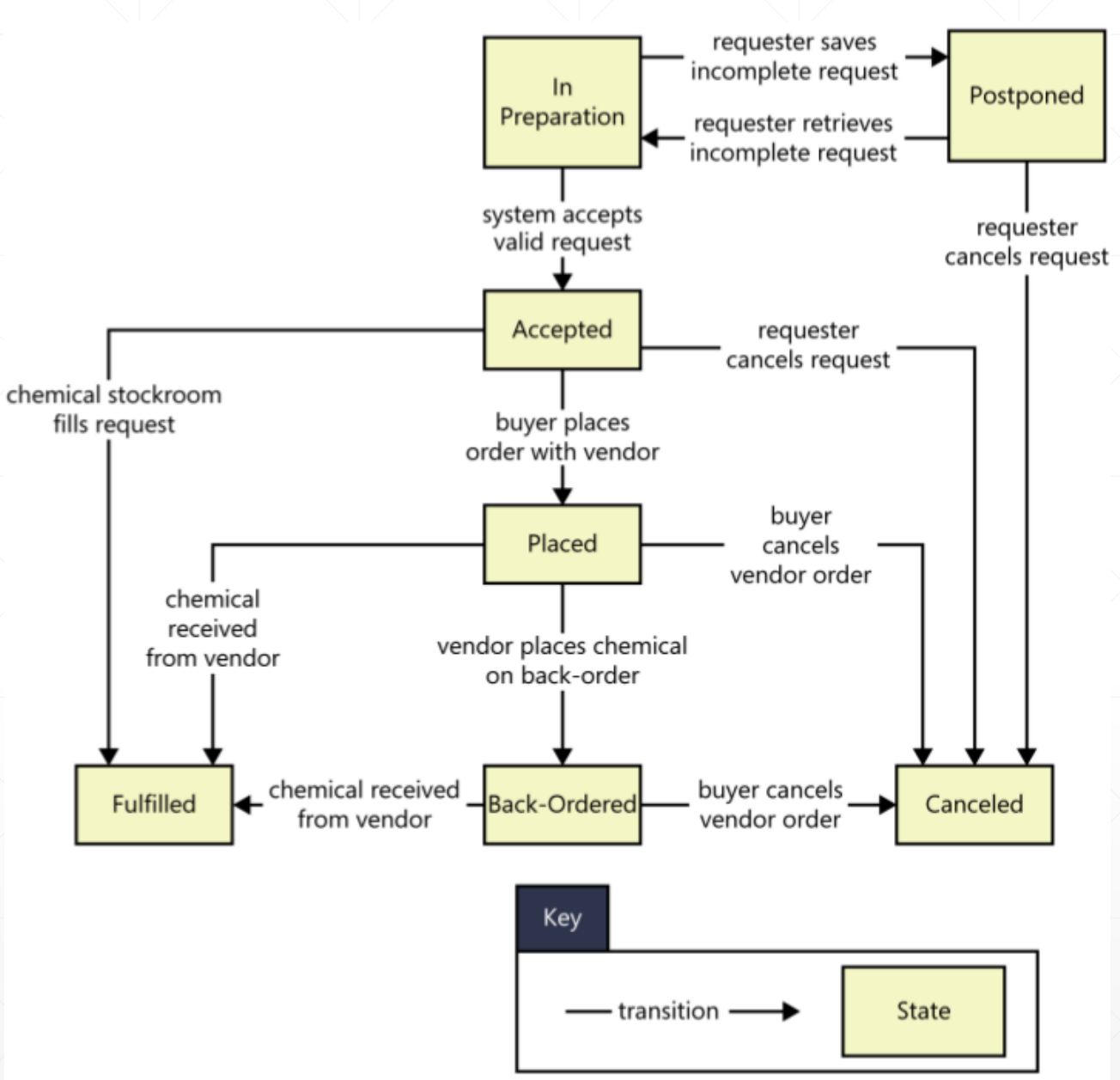
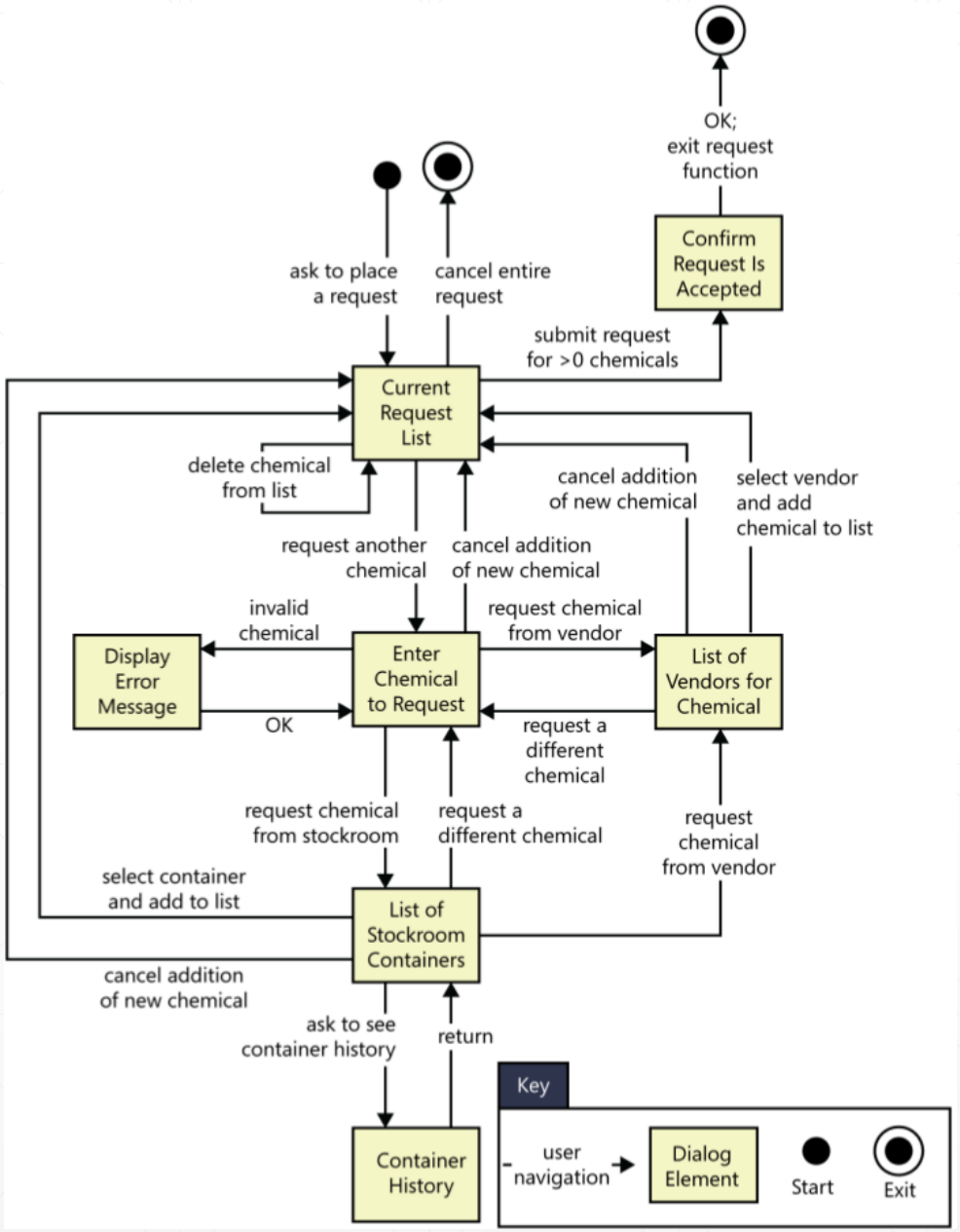


Tabela stanja – tekstualna reprezentacija dijagama

	In preparation	Postponed	Accepted	Placed	Back-Ordered	Fulfilled	Canceled
In Preparation	no	user saves incomplete request	system accepts valid request	no	no	no	no
Postponed	user retrieves incomplete request	no	no	no	no	no	no
Accepted	no	no	no	buyer places order with vendor	no	chemical stockroom fills request	requester cancels request
Placed	no	no	no	no	vendor places chemical on back-order	chemical received from vendor	buyer cancels vendor order
Back-Ordered	no	no	no	no	no	chemical received from vendor	buyer cancels vendor order
Fulfilled	no	no	no	no	no	no	no
Canceled	no	no	no	no	no	no	no

Mapa dijaloga

- **Mapa dijaloga** predstavlja korisnički interfejs modelovan u obliku dijagrama stanja. Ona hvata suštinu interakcija sa korisnicima sistema i tokove zadataka bez opterećivanja detaljnim izgledom ekrana.
- Mapa dijaloga prikazuje svaki element dijaloga kao stanje (pravougaonik) i svaku dozvoljenu opciju navigacije kao tranziciju (strelica). Uslov za aktiviranje navigacije prikazan je kao tekst tranzicije. Postoji nekoliko tipova uslova:
- Korisnička akcija, kao što je pritisak na funkcijski taster, pritiskom na hipervezu, ili pravljenje gesta na ekranu osetljivom na dodir.
- Vrednost podatka, kao što je neispravna ulazna vrednost koja aktivira ekran poruke o grešci.
- Stanje sistema, kao što je otkrivanje da štampač nema papira.
- Neka kombinacija prethodnog, kao što je kucanje broja opcije menija i pritisak na taster Enter.



Tabele i stabla odlučivanja

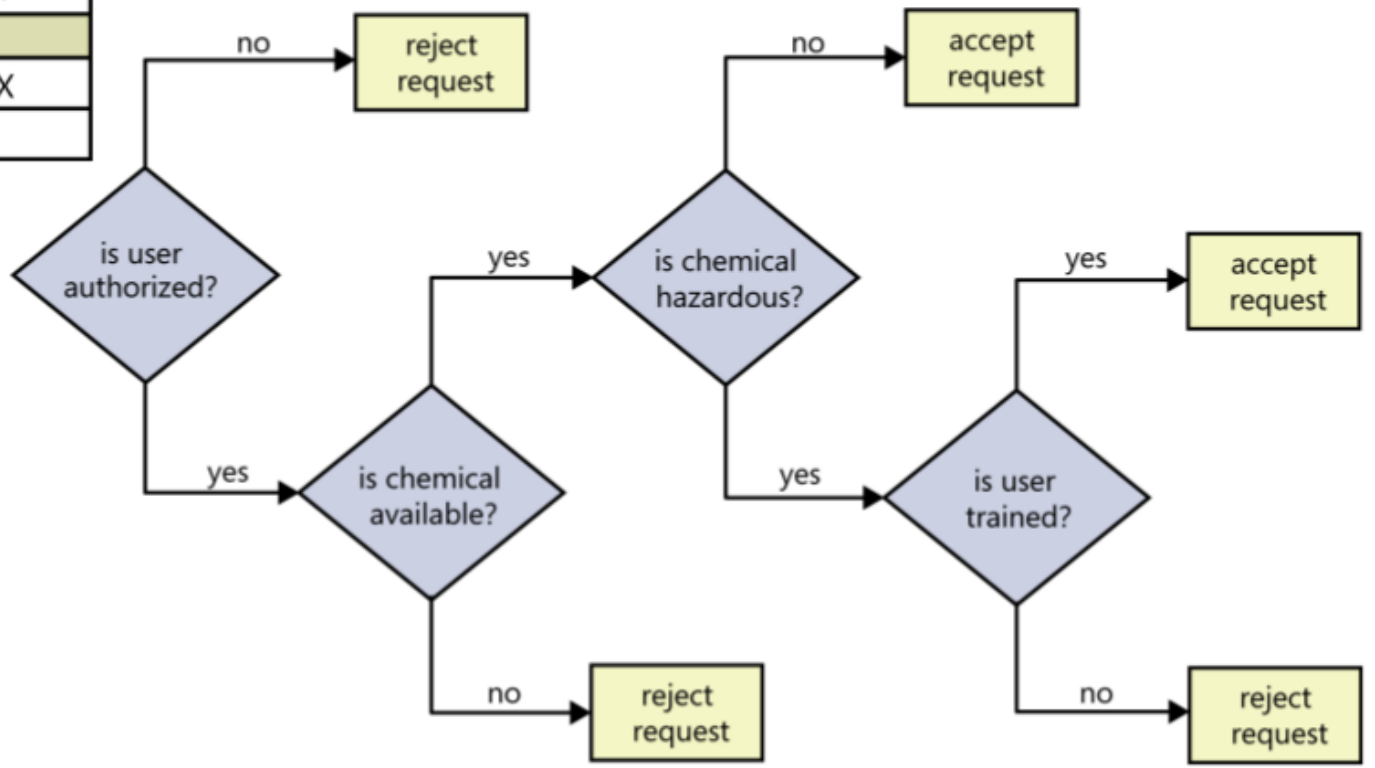
- Tabele i stabla odlučivanja - dve alternativne tehnike za specifikaciju šta sistem treba da radi kada su u pitanju složena logika i odluke.
- Potrebno je specifikovati šta sistem radi pod svakom kombinacijom uslova.
- U tekstualnom opisu lako je preskočiti kombinaciju, što se teško uočava pregledom.

- Tabela odlučivanja navodi različite vrednosti za sve faktore koji utiču na ponašanje i pokazuje očekivanu akciju sistema kao odgovor na svaku kombinaciju faktor.
- Faktori mogu biti prikazani ili kao iskazi sa mogućim uslovima istinito i lažn, ili kao pitanja sa mogućim odgovorima da i ne, ili kao pitanja sa više od dve moguće vrednosti.

Requirement Number					
Condition	**1**	**2**	**3**	**4**	**5**
User is authorized	F	T	T	T	T
Chemical is available	—	F	T	T	T
Chemical is hazardous	—	—	F	T	T
Requester is trained	—	—	—	F	T
Action					
Accept request			X		X
Reject request	X	X		X	

Ekvivalentno stablo odlučivanja

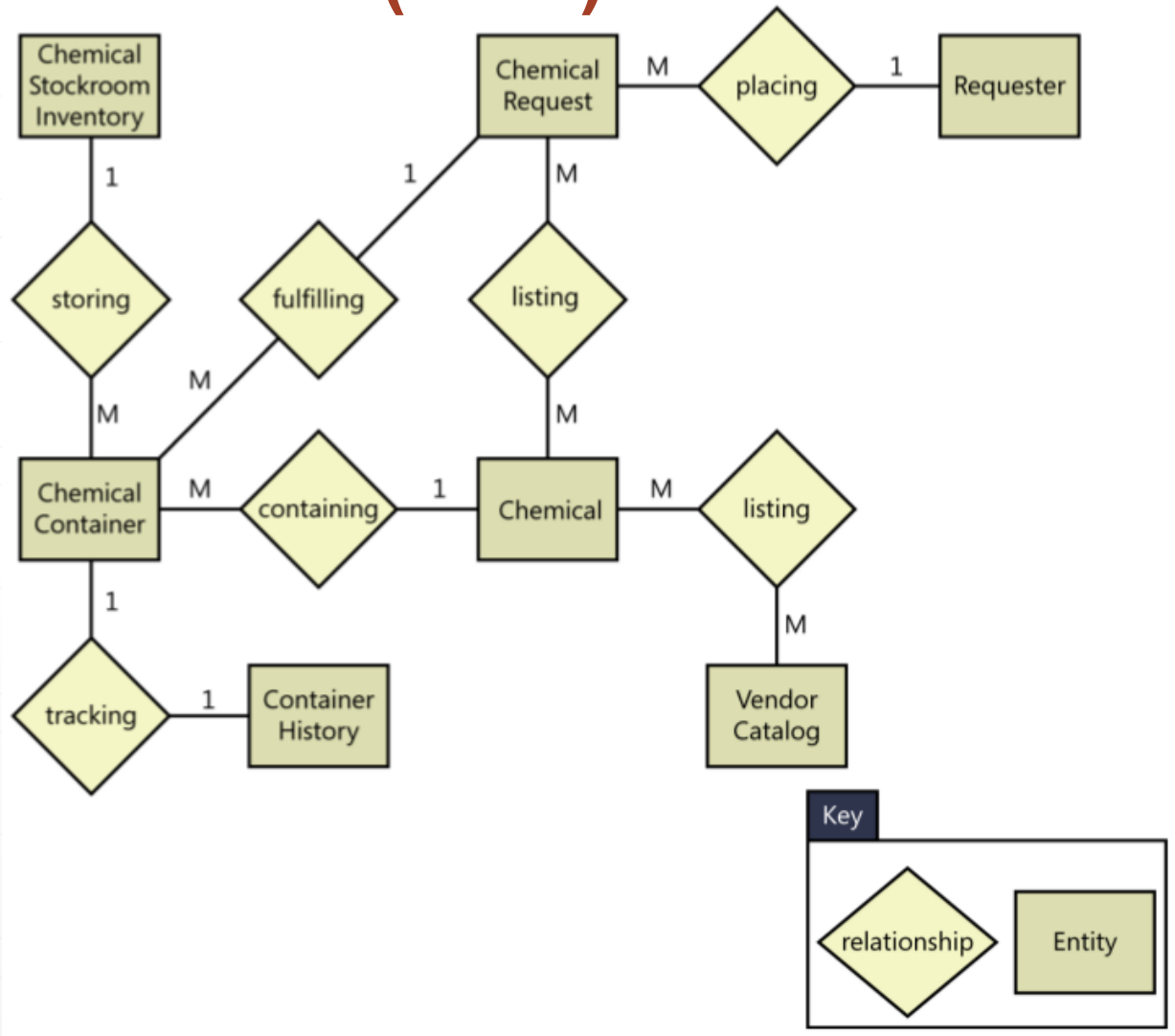
Requirement Number					
Condition	**1**	**2**	**3**	**4**	**5**
User is authorized	F	T	T	T	T
Chemical is available	—	F	T	T	T
Chemical is hazardous	—	—	F	T	T
Requester is trained	—	—	—	F	T
Action					
Accept request			X		X
Reject request	X	X		X	



Modelovanje zahteva za podatke

- Najčešće korišćen model podataka je ***Dijagram entiteti-veze*** ili *ERD*.
- U fazi analize zahteva, ERD nam pomaže da razumemo i razjasnimo komponente podataka poslovanja ili sistema, bez obaveze da proizvod mora da uključi bazu podataka.
- Kada kreiramo ERD tokom projektovanja, tada definišemo logičku ili fizičku (implementacionu) strukturu baze podataka sistema.
- **Entiteti** mogu da predstavljaju fizičke jedinice (uključujući i ljude) ili agregacije podataka koji su od značaja za poslovanje koje se analizira ili za sistem koji nameramo da izgradimo.
- Svaki entitet je opisan jednim ili više atributa, pojedinačni slučajevi jednog entiteta će imati različite vrednosti atributa. **Rečnik podataka** sadrži precizne definicije tih atributa.
- Rombovi u ERD predstavljaju relacije, koje identifikuju logičke veze između parova entiteta.
- Kada pitate klijente da pregledaju neki ERD, zamolite ih da provere da li su prikazane relacije sve ispravne i odgovarajuće. Takođe ih zamolite da identifikuje bilo kakve nedostajuće entitete ili eventualne nedostajuće veze između entiteta.
- Kardinalnost, ili multiplicitet, svakog odnosa je prikazan brojem ili slovom na granama koje povezuju entitete i odnose.
- Različite ERD notacije koriste različite konvencije za predstavljanje kardinalnosti.

Model entiteta i veza (ERD)



Model entiteta i veza (ERD): alternativne notacije

- Jedan entitet **C.S.Inventory** u relaciji je **stores** sa nula ili više entiteta **Ch. Container** (notacija James Martin)

- UML notacija (primetiti da je treći odeljak u klasama ostavljen prazan)

Rečnik podataka

- ***Rečnik podataka*** je kolekcija informacija o entitetima podataka koji se koriste u aplikaciji.
- Prikupljanje informacija o sastavu, tipovima podataka, dozvoljenim vrednostima, itd na jednom mestu služi da identifikuje kriterijume valjanosti podataka, pomaže programerima da pravilno pišu programe i smanjuje probleme integracije .
- Stavka u rečniku podataka može predstavljati sledeće tipove elemenata podataka:
- Primitivni-primitivan element podataka je onaj za koji dalja dekompozicija nije moguća ili potrebna.
- Strukturirani- struktura podataka (ili zapis) se sastoji od više elemenata podataka. Elementi podataka koji se pojavljuju u strukturi takođe moraju imati definicije u rečniku podataka. Ako je element u strukturi podataka neobavezan (vrednost ne mora da se definiše od strane korisnika ili sistema) , stavlja se u zagrade.
- Ponavljanjajuća grupa - Ukoliko više instanci određenog elementa podataka može da se pojavi u strukturi, staviti tu stavka u {} zagrade. Dozvoljeni broj mogućih ponavljanja u obliku minimum: maksimum piše se ispred zagrade.

Rečnik podataka - primer

Data Element	Description	Composition or Data Type	Length	Values
Chemical Request	request for a new chemical from either the Chemical Stockroom or a vendor	Request ID + Requester + Request Date + Charge Number + 1:10{Requested Chemical}		
Delivery Location	the place to which requested chemicals are to be delivered	Building + Lab Number + Lab Partition		
Number of Containers	number of containers of a given chemical and size being requested	Positive integer	3	
Quantity	amount of chemical in the requested container	numeric	6	
Quantity Units	units associated with the quantity of chemical requested	alphabetic characters	10	grams, kilograms, milligrams, each
Request ID	unique identifier for a request	integer	8	system-generated sequential integer, beginning with 1

CRUD matrica

- Tokom analize podataka, moguće je mapirati različite predstave informacija jedne naspram drugih da bismo pronašli nedostatke, greške i nedoslednosti.
- CRUD matrica je rigorozna tehnika analize podataka za otkrivanje nedostajućih zahteva.
- CRUD označava **Create**, **Read**, **Update** i **Delete**.
- CRUD matrica koreliše sistemske akcije sa entitetima podataka da pokaže gde i kako se svaki entitet stvora, čita, ažurira i briše.
- Moguće je ovim osnovnim operacijama dodati i **L** u matrici da ukaže na to da se entitet pojavljuje kao lista za izbor, **M** da ukaže premeštanje podataka sa jedne lokacije na drugu, a možda i drugo **C** do ukaže na kopiranje podataka.

Korelacije

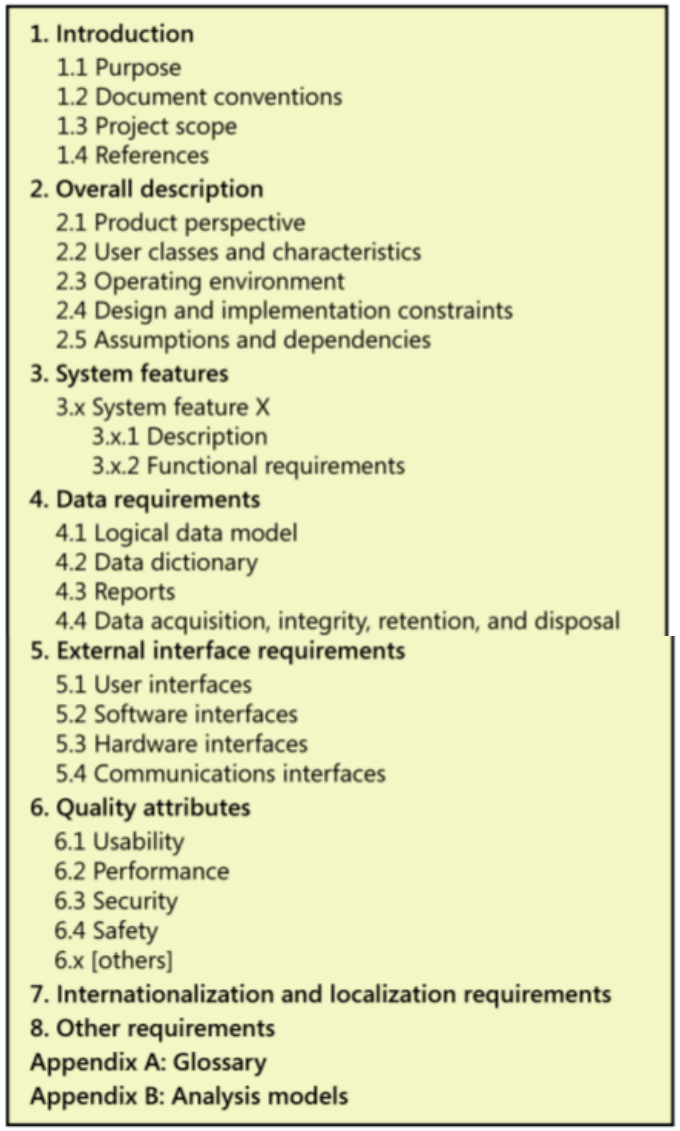
- U zavisnosti od korišćenog pristupa specifikaciji zahteva, mogu se ispitati različite vrste korelacija, uključujući sledeće:
  - **Entiteti podataka i korisničke priče ili slučajevi upotrebe**
  - **Entiteti podataka i sistemski događaji**
  - **Klase objekata i slučajevi upotrebe**

CRUD matrica: primer

Use Case \ Entity	Order	Chemical	Requester	Vendor Catalog
Place Order	C	R	R	R
Change Order	U, D		R	R
Manage Chemical Inventory		C, U, D		
Report on Orders	R	R	R	
Edit Requesters			C, U	

Specifikacija softverskih zahteva

- eng. *Software requirements specification* – SRS
- SRS navodi:
  - funkcije i mogućnosti koji softverski sistem mora obezbediti,
  - njegove karakteristike i ograničenja koja se moraju poštovati.
- SRS bi trebalo da opiše, onoliko detaljno koliko je neophodno, ponašanje sistema u različitim uslovima, kao i željene kvalitete sistema kao što su performanse, bezbednost i upotrebljivost.
- SRS je osnova za kasnije planiranje projekta, dizajn i kodiranje sistema, kao i osnova za testiranje sistema i korisničke dokumentacije. Međutim, SRS ne bi trebalo da sadrži detalje projektovanja, implementacije, testiranja ili upravljanja projektom osim unapred poznatih dizajnerskih i implementacionih ograničenja.



Karakteristike dobrog dokumenta o zahtevima

- precizan i nedvosmislen
- korektan i kompletan
- konzistentan
- omogućava verifikaciju i praćenje zahteva
- opisuje isključivo spoljno ponašanje sistema
- opisuje ograničenja
- jednostavan je za izmenu
- anticipira buduće potrebe
- opisuje reakcije na neželjene događaje
- razumljiv za korisnika

Problemi sa zahtevima

„Buka“ - prisustvo teksta koji ne sadrži informacije vezane za problem.

„Tišina“ - izostavljene informacije

Prespecifiranost - određivanje specifičnog rešenja.

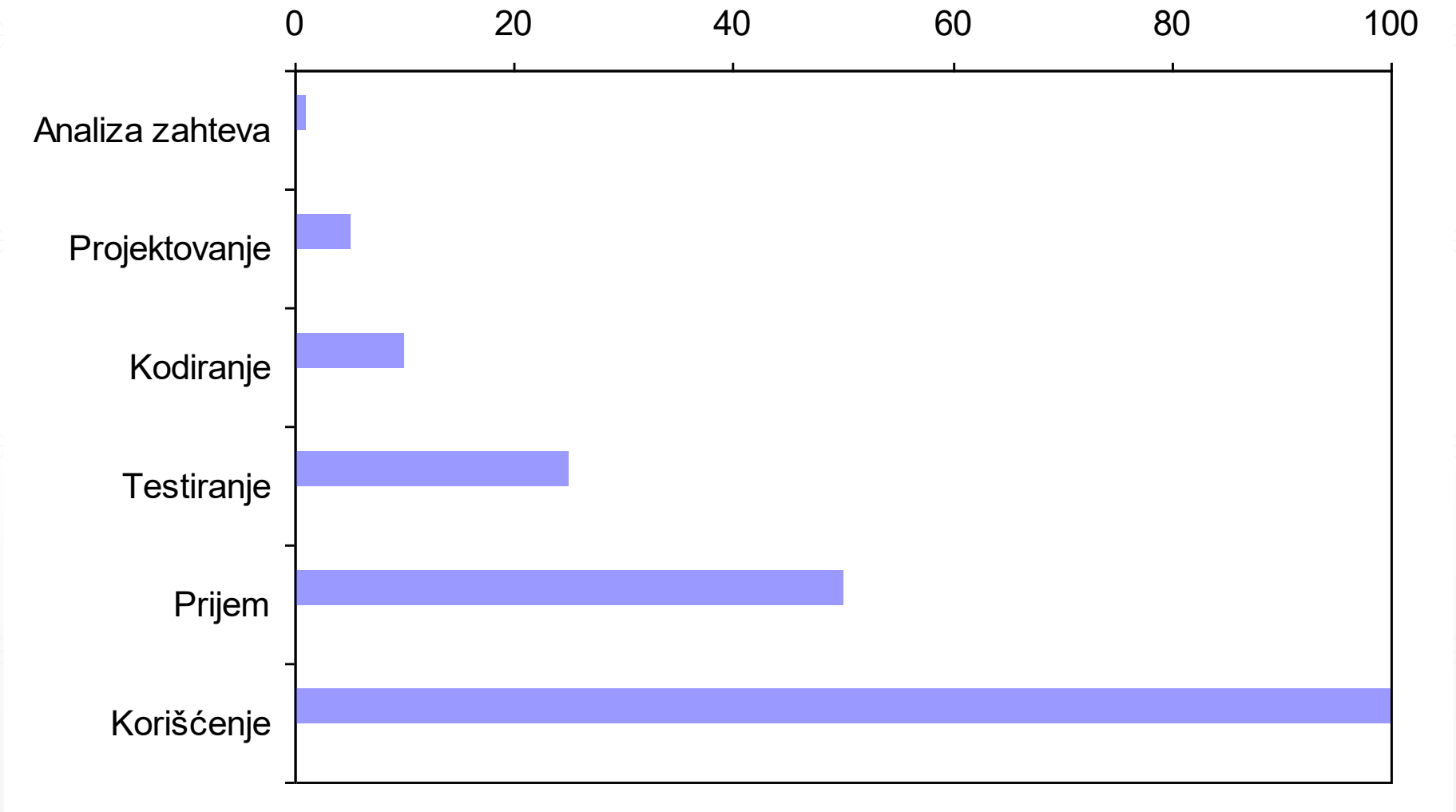
Kontradiktornost - različiti opisi iste stvari

Nejasnoća - korišćenje pojmova sa više značenja, a nije jasno na koje se misli.

Referenciranje unapred - kada se koristi pojam koji se definiše kasnije u tekstu. Odslikava slabu strukturiranost dokumenta.

„lepe želje“ - nerealni zahtevi.

Troškovi otklanjanja greške u zahtevima



Verifikacija i validacija zahteva

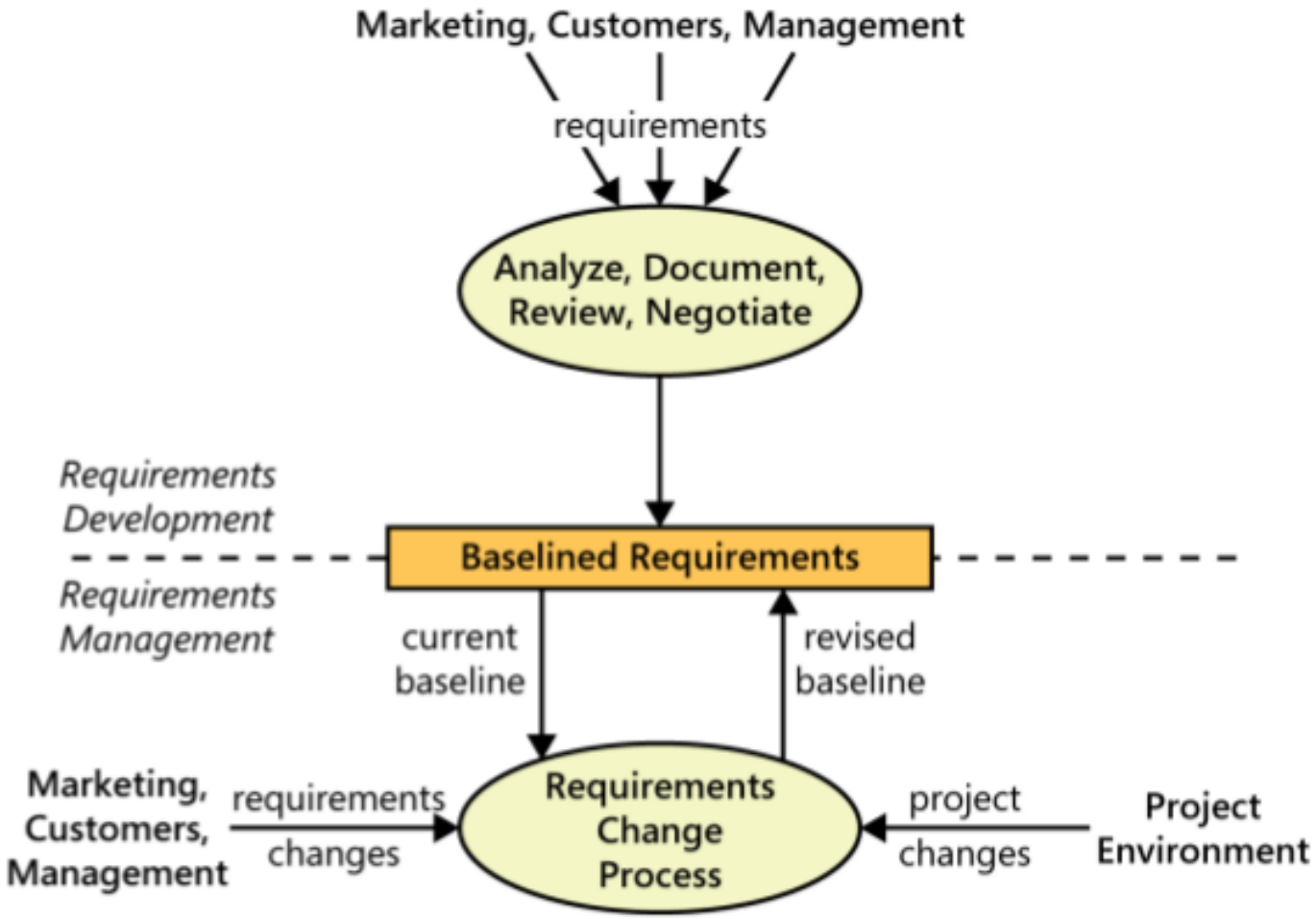
- Značajna aktivnost, jer je otklanjanje greške specifikacije skupo u fazama dizajna i implementacije.
- Pristupi:
  - izvršna specifikacija
  - korišćenje prototipa
  - zajednički pregledi

Upravljanje zahtevima

Aktivnosti upravljanja zahtevima uključuju sledeće:

- Definisanje polazne osnove zahteva, to jest, snimka u vremenu koji reprezentuje dogovoreni, pregledani i odobreni skup funkcionalnih i nefunkcionalnih zahteva, često za specifično izdanje proizvoda ili iteraciju projekta.
- Procenjivanje uticaja predloženih izmena zahteva i uključivanje odobrenih izmena u projekat na kontrolisan način.
- Ažuriranje planova u skladu sa zahtevima kako oni evoluiraju.
- Dogovaranje novih obaveza na osnovu procenjenog uticaja izmena zahteva.
- Definisanje relacija i zavisnosti između pojedinih zahteva.
- Praćenje pojedinih zahteva do njihovoh dizajna, koda i testova.
- Praćenje statusa zahteva i aktivnosti izmena kroz ceo projekat.

Upravljanje zahtevima



Uloga prototipa u softverskom razvoju

- Pomaže da se razjasne zahtevi.
- Sredstvo za validaciju zahteva.
- Preporuka: grafičke korisničke interfejse specificirati isključivo putem prototipa.
- Vrste:
  - ***makete*** (za bacanje) i prototipovi (podskup funkcionalnosti, prerastaju u operativni softver),
  - ***ilustrativni*** (rade samo za fiksan skup ulaznih podataka) i funkcionalni (potpuno realizuju podskup korisnićkih zahteva),
  - ***deskriptivni*** (rade u razvojnom okruženju) i operativni (rade u operativnom okruženju).

Agilni pristup inženjerstvu zahteva

- **Ponašanjem vođen razvoj** (softvera) (eng. *Behavior driven development*, skr. *BDD*) je specijalizovana varijanta razvoja vođenog testovima (TDD) koja se fokusira na specifikaciju ponašanja softverskih komponenata.
- **Razvoj vođen testovima** (TDD) je metodologija koja u suštini kaže da je za svaku jedinicu softvera, programer mora da:
  - prvo definiše skup testova za tu komponentu;
  - implementira komponentu;
  - potvrdi da implementacija komponente zadovoljava testove (da testovi uspešno prolaze).
- U BDD koristi se poluformalni format za specifikaciju ponašanja koji je pozajmljen iz specifikacije korisničkih priča (*user stories*) iz oblasti objektno orijentisane analize i dizajna.
- BDD propisuje da poslovni analitičari i programeri sarađuju u ovoj oblasti i treba da odrede ponašanje u smislu korisničkih priča, od kojih je svaka zapisana u posebnom dokumentu.

Struktura korisničke priče kod BDD

- Svaka korisnička priča treba da sledi sledeću strukturu:
- **Naslov**
  - Priča treba da ima jasan, eksplicitan naziv.
- **Narativ**
  - ukratko, uvodni deo koji određuje ko je pokretač ili primarni akter priče (akter koji izvlači poslovnu korist od priče)
  - koji efekat akter želi od priče
  - kakav poslovna vrednost za aktera proizilazi iz ovog efekta
- **Kriterijumi ili scenarija za prijem** - opis svakog konkretnog scenarija priče.

  Takav scenario ima sledeću strukturu:
  - On počinje navođenjem početnih uslova za koje se pretpostavlja da su ispunjeni na početku scenarija.
  - Uslov može da se sastoji od jedne ili više klauzula.
  - Događaj koji izaziva početak scenarija.
  - Očekivani ishod, u jednoj ili više klauzula.

Gherkin: jezik za opis ponašanja

- Primer korisničke priče (podebljane su ključne reči):

Feature: Some terse yet descriptive text of what is desired
In order to realize a named business value
As an explicit system actor
I want to gain some beneficial outcome which furthers the goal

Scenario: Some determinable business situation
Given some precondition
And some other precondition
When some action by the actor
And some other action
And yet another action
Then some testable outcome is achieved
And something else we can check happens too

Primer Gherkin specifikacije

```
# features/search.feature
Feature: Search
  In order to see a word definition
  As a website user
  I need to be able to search for a word

  Scenario: Searching for a page that does exist
    Given I am on "/wiki/Main_Page"
    When I fill in "search" with "Behavior Driven Development"
    And I press "searchButton"
    Then I should see "agile software development"

  Scenario: Searching for a page that does NOT exist
    Given I am on "/wiki/Main_Page"
    When I fill in "search" with "Glory Driven Development"
    And I press "searchButton"
    Then I should see "Search results"
```

Drugi primer Gherkin specifikacije – ETF sajt

Feature: sledi link osnovne studije na ETF sajtu

Da bih video odsek koji me interesuje

Kao student

Moram da vidim Softversko inzenjerstvo u spisku odseka osnovnih studija

@mink:selenium2

Scenario: sledi link osnovne studije

Given Nalazim se na strani „http://www.etf.bg.ac.rs“

When Pritisnem link „Основне студије“

Then Treba da dobijem „Софтверско инжењерство“

Problemi sa zahtevima

- Jezik *Gherkin* pruža način da se opiše ponašanje aplikacije u jeziku čitljivom korisnicima.
- Ali kako testirati da je zahtevano ponašanje zaista i implementirano?
  Tj. da aplikacija zadovoljava poslovna očekivanja opisana u scenarijima?
- Alati kao što je *Cucumber* (Ruby) ili *Behat* (PHP) pružaju način da se mapira 1-na-1 koraci scenarija (akcije) na stvarni PHP kod kroz definiciju koraka (regularni izraz u komentaru mapira korak iz *Gherkin* specifikacije na PHP funkciju):

```
/**
 * @When /^I do something with "([^"]*)"$/
 */
public function iDoSomethingWith($argument)
{
    // do something with $argument
}
```

Primer definicije koraka (Behat/Mink-Selenium 2 drajver)

```
<?php
. . . .
class FeatureContext extends RawMinkContext
{
. . . .
    /**
     * @Given /^Nalazim se na strani "([^"]*)"$/
     */
    public function nalazimSeNaStrani($arg1)
    {
        $this->getSession()->visit($arg1);
    }

    /**
     * @When /^Kliknem "([^"]*)"$/
     */
    public function kliknem($link)
    {
        $this -> getSession() -> getPage() -> clickLink($link);
    }
```

```
    /**
     * @Then /^Treba da dobijem "([^"]*)"$/
     */
    public function trebaDaDobijem($tekst)
    {
        $this -> getSession() -> wait(1000);
        $this -> assertSession() ->
                responseContains($tekst);
        //throw new PendingException();
    }
}
```

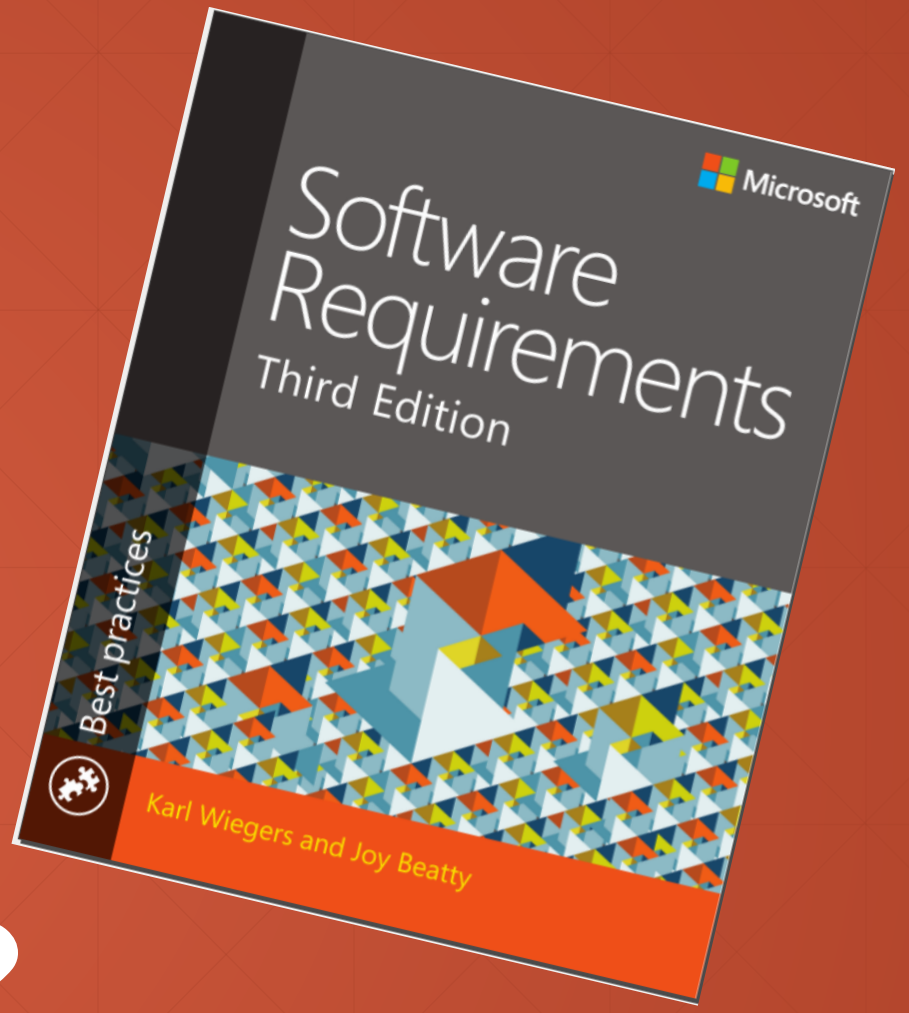
Primer rezultati izvršavana testa

- Napomena: U drugom terminalu potrebno je pre testiranja pokrenuti
- Selenium server: java -jar selenium-server-standalone-2.31.0.jar

```
db@toshiba:~/behat$ php behat.phar
Feature: sledi link osnovne studije na ETF sajtu
  Da bih video smer koji me interesuje
  Kao student
  Moram da vidim softversko inzenjerstvo u spisku smerova

  @mink:selenium2
  Scenario: sledi link osnovne studije                    # feature
    Given Nalazim se na strani "http://www.etf.rs" # Feature
Strani()
    When Kliknem "Основне студије"                        # Feature
    Then Treba da dobijem "Софтверско инжењерство" # Feature
jem()

1 scenario (1 passed)
3 steps (3 passed)
0m9.988s
```



Pitanja?

Hvala na pažnji.