



Funded by the  
Erasmus+ Programme  
of the European Union



---

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

---



## KI206 - PROCES I METODOLOGIJE RAZVOJA SOFTVERA

Inženjerstvo zahteva

Lekcija 03

PRIRUČNIK ZA STUDENTE

# KI206 - PROCES I METODOLOGIJE RAZVOJA SOFTVERA

## Lekcija 03

### **INŽENJERSTVO ZAHTEVA**

- ✓ Inženjerstvo zahteva
- ✓ Poglavlje 1: Funkcionalni i nefunkcionalni zahtevi
- ✓ Poglavlje 2: Dokument sa zahtevima
- ✓ Poglavlje 3: Specifikacija zahteva
- ✓ Poglavlje 4: Procesi inženjerstva zahteva
- ✓ Poglavlje 5: Utvrđivanje i analiza zahteva
- ✓ Poglavlje 6: Validacija zahteva
- ✓ Poglavlje 7: Upravljanje zahtevima
- ✓ Poglavlje 8: Modelovanje zahteva
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

### UVOD

#### *Uvod u nastavnu jedinicu*

Cilj ove nastavne jedinice je da vas uvede u metode specificiranja zahteva korisnika . Ova nastavna jedinica vam omogućava da:

- razumete koncepte zahteva korisnika i sistema i zašto ovi zahtevi treba da se ne pišu na različite načine:
- razumete razliku između funkcionalnih i nefunkcionalnih softverskih zahteva:
- razumete kako zahtevi mogu da budu organizovani u dokumentu o softverskim zahtevima:
- razumete glavne aktivnosti inženjeringu zahteva: prikupljanje zahteva, analize i validacije, kao i odnose između ovih aktivnosti:
- razumete zašto je upravljanje zahtevima neophodno i kako ono podržava druge aktivnosti inženjeringu zahtev

## ▼ Poglavlje 1

# Funkcionalni i nefunkcionalni zahtevi

## ŠTA SU ZAHTEVI ZA SOFTVER?

*Zahtevi za razvoj softverskog sistema su opisi onoga što treba sistem da radi. Inženjerstvo zahteva je proces nalaženja, analize, dokumentovanja i provere ovih servisa i ograničenja.*

**Zahtevi za razvoj softverskog sistema** su opisi onoga što treba sistem da radi. To su servisi koji obezbeđuju i ograničavaju svoju operaciju. Ti zahtevi odražavaju potrebe korisnika za sistemom koje služe za određene svrhe, kao što su: kontrola uređaja, postavljanje naloga, ili pronalaženje informacije.

Inženjerstvo zahteva je proces nalaženja, analize, dokumentovanja i provere ovih servisa i ograničenja.

**Zahtevi** korisnika su opšti apstrakti zahteva, a zahtevi sistema predstavljaju detaljne opise onoga što sistem treba da radi.

- **Zahtevi korisnika** su iskazi u običnom jeziku i sa dijagramima, o servisima sistema koje očekuju korisnici sistema i ograničenja pod kojima mora da sistem radi.
- **Zahtevi sistema** su detaljniji opisi funkcija softverskog sistema, servisa i operacionih ograničenja. Dokument sa zahtevima sistema (ponekad se taj dokument naziva i „funkcionalna specifikacija“) definiše precizno šta treba da bude ostvareno (implementirano). To može biti i prilog ugovoru između kupca sistema i proizvođača softvera.

Koriste se različiti nivoi zahteva jer se prenose informaciju o sistemu različitim kategorijama čitaoca

Različiti nivoi detaljnosti definisanja zahteva su korisni jer ne treba svim korisnicima sistema isti nivo detaljnosti zahteva. Slika 1 prikazuje razliku između zahteva korisnika i zahteva sistema, a na primeru sistema za upravljanje pacijentima sa mentalnim problemima.(MHC-PMC sistem). Vidi se da su zahtevi korisnika dosta uopšteni. S druge strane, zahtevi sistema su detaljniji, jer moraju da obezbede specifične informacije o servisima i funkcijama koje treba da obezbedi sistem koji se razvija.

1. MHC-PMS treba da generiše mesečne menadžment izveštaje koji prikazuju cene prepisanih lekova od strane svake klinike za vreme tog meseca.

**Specifikacija zahteva sistema:**

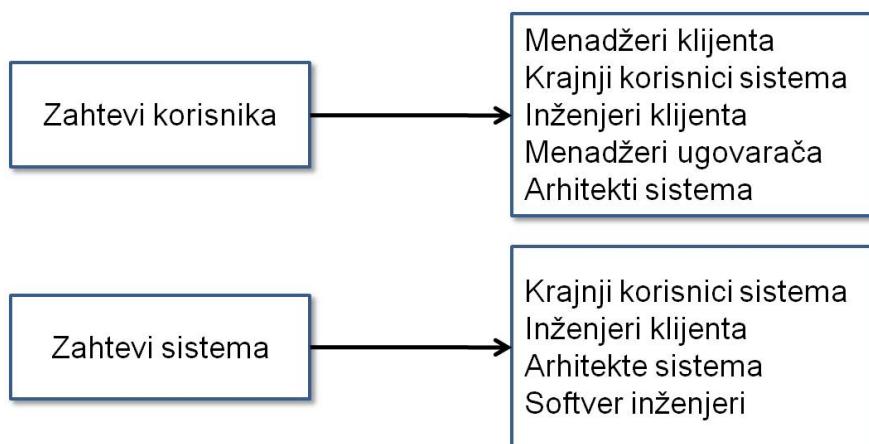
- 1.1 Na kraju svakog radnog dana u mesecu, treba generisati pregled prepisanih lekova, njihova cena, i klinike koje su prepisali lekove.
- 1.2 Sistem treba da automatski generiše izveštaj za štampanje posle 17:30 zadnjeg radnog dana u mesecu.
- 1.3 Izveštaj treba da se kreira za svaku kliniku, a on treba da sadrži listu imena lekova, ukupan broj prepisanih lekova, broj doza koji je preписан, ukupna cena prepisanih lekova.
- 1.4 Ako su lekovi koriste različite jedinice za doze (npr. 10 mg, 20 mg) onda za svaku jedinicu doze treba pripremiti poseban izveštaj.
- 1.5 Pristup izveštajima o troškovima se ograničava samo autorizovanim korisnicima koji su na listi kontrolnog spiska za pristup.

Slika 1.1.1 Primer zahteva korisnika i sistema

## KATEGORIJE KORISNIKA SISTEMA I VRSTE ZAHTEVA

*Postoje zahtevi korisnika i zahtevi sistema. Različiti korisnici su zainteresovani za različite kategorije zahteva.*

Različiti korisnici sistema zahtevaju različite nivoe detaljnosti definisanja zahteva. Na slici 2 prikazani su zahtevi od interese za različite kategorije korisnika sistema, tj. aktere sistema. Čitaoci zahteva korisnika obično nisu zainteresovani kako sistem treba da bude primenjen. Čitaoci zahteva sistema, s druge strane, znaju precizno šta bi sistem trebalo da radi, jer se oni brinu o podršci poslovnim procesima ili rade na primeni sistema.



Slika 1.1.2 Kategorije korisnika koji postavljaju zahteve

Zahtevi sistema se mogu podeliti u dve kategorije zahteva:

- **Funkcionalni zahtevi:** su izjave o servisima koje sistem treba da obezbedi, i definišu kako sistem reaguje na različite ulaze, i kako se ponaša u pojedinim situacijama. Oni definišu funkcije sistema.

- **Nefunkcionalni zahtevi:** su ograničenja servisa ili funkcija koje sistem obezbeđuje. Mogu da obuhvate vremenska ograničenja, ograničenja procesa razvoja, i ograničenja nametnuta od strane standarda. Nefunkcionalni zahtevi se primenjuju na ceo sistem, a ne na pojedina svojstva sistema ili servise. To su najčešće zahtevi o performansama sistema.

U stvarnosti, ne postoji vrlo jasna razlika između funkcionalnih i nefunkcionalnih zahteva. Na primer, zahtev korisnika koji se odnosi na bezbednost informacija, može se pojaviti i kod nefunkcionalnih zahteva. S druge strane, detaljni zahtevi sistema, mogu da izazovu javljanje novih zahteva, tj. novih servisa koji su jasno funkcionalni.

Ovo pokazuje da zahtevi *nisu međusobno nezavisni*, te se često dešava da jedan zahtev generiše ili ograničava druge zahteve. Zahtevi sistema, zbog toga, često, pored definisanja servisa i svojstava sistema koje sistem treba da ima, specificiraju neophodnu funkcionalnost koja obezbeđuje da se ovi servisi isporučuju na pravi način.

## ZADATAK 1

### *Identifikovati korisnike informacionog sistema univerziteta*

Za potrebe rada univerziteta treba razviti sistem koji podržava osnovne procese rada. Neophodno je obezbediti administraciju ljudskih resursa (profesora i studenata) što podrazumeva dodavanje/ažuriranje novog zaposlenog ili studenta. Pored toga, studentskoj službi sistem mora da obezbedi administraciju predmeta, angažovanja profesora na predmetima i dodeljivanje predmeta studentima. Svaki profesor treba da ima mogućnost unosa svih predispitnih poena i ocena za predmete za koje je angažovan, dok studentu treba omogućiti pregled predmeta sa trenutnim poenima i ocenama. Student takođe ima mogućnost prijave ispita na sistemu.

**Definišite kategorije i identifikujte korisnike predloženog sistema.**

### ✓ 1.1 Funkcionalni zahtevi

#### ŠTA SU FUNKCIONALNI ZAHTEVI?

*Funkcionalni zahtevi sistema opisuju šta bi sistem trebalo da radi. Korisnici ih definišu na apstraktan način, a inženjeri razvoja ih pretvaraju u detaljne zahteve (ulaz, izlaz, izuzeci...)*

Funkcionalni zahtevi sistema opisuju šta bi sistem trebalo da radi. Oni zavise od vrste softvera koji se razvija, očekivanih korisnika softvera i opštег pristupa koji primenjuje organizator kada definiše (piše) zahteve.

Ako su izraženi u vidu zahteva korisnika, funkcionalni zahtevi su obično opisani na jedan apstraktan način, u skladu sa razumevanjem korisnika sistema. Međutim, kada su detaljno

opisani specifični funkcionalni zahtevi sistema, onda oni predstavljaju funkcije sistema, njegove ulaze i izlaze, izuzeća, i dr.

**Funkcionalni zahtevi sistema** variraju od opštih zahteva koji opisuju ono što bi sistem trebalo da radi, pa do vrlo detaljnih vrlo specifičnih zahteva, koji odražavaju lokalni način rada ili neki postojeće sisteme organizacije. Na primer, daju se primjeri funkcionalnih zahteva MHC-PMS sistema:

1. Korisnik treba da ima mogućnost da izabere liste sastanaka za sve pacijente.
2. Sistem treba da izradi svakog dana, za svaku kliniku, listu pacijenata koji imaju zakazane pregleda tog dana.
3. Svaki zaposleni koji upotrebljava sistem mora da bude jedinstveno identifikovan sa svojim brojem sa osam cifara.

Ovi funkcionalni zahtevi korisnika definišu funkcije koje sistem treba da obezbedi. Oni pokazuju da se funkcionalni zahtevi mogu pisati sa različitim nivoima detaljnosti.

Zahtevi treba da budu jasni i nedosvremeneni. Oni treba da se sprovedu i time promene sistem. Ako nisu dobro definisani, i korisnik kasnije ima primedbi, postupak se mora ponoviti, što pravi troškove i izaziva kašnjenja u projektu. Zbog toga, zahtev *mora bude kompletan i konsistentan*. **Kompletost zahteva** znači da su definisani svi servisi koje korisnik zahteva. Konsistentnost zahteva znači da nema kontradiktornosti među zahtevima.

Kod velikih i složenih sistema praktično je nemoguće ostvariti kompletost i konsistentnost funkcionalnih zahteva. Razlozi su:

- greške kod pisanja zahteva (a ima ih veliki broj),
- korisnici sistema imaju različite interese te postavljaju suprotstavljene zahteve, što ih čini nekonsistentnim.

## ZADATAK 2

*Definišite funkcionalne zahteve za sistem predložen u zadatku 1*

1. Definišite funkcionalne zahteve za sistem definisan u zadatku 1.

### ✓ 1.2 Nefunkcionalni zahtevi

#### ŠTA SU NEFUNKCIONALNI ZAHTEVI?

*Nefunkcionalni zahtevi su zahtevi koji nisu direktno povezani sa servisima koje sistem treba da obezbedi svojim korisnicima, već definišu ograničenja implementacije sistema.*

Nefunkcionalni zahtevi su zahtevi koji nisu direktno povezani sa servisima koje sistem treba da obezbedi svojim korisnicima, već sa svojstvima tih servisa, kao što su pouzdanost, brzina odgovora, i zauzeće memorije. Oni definišu ograničenja implementaciji sistema, kao što su na primer, svojstva UI uređajima, ili predstavljanje podataka na interfejsu sa drugim sistemima.

Nefunkcionalni zahtev, kao što su performanse, bezbednost ili raspoloživost, često određuju ili ograničavaju karakteristike celog sistema. Zato su često i kritičniji nego funkcionalni zahtevi. Korisnik može da na drugi način dobije od sistema odgovor na funkciju koju sistem direktno ne obezbeđuje zbog loše definisanog funkcionalnog zahteva. Ali, ako je nefunkcionalna zahtev loše definisan, pa to čini ceo sistem sporim, ili nepouzdanim ili nebezbednim, onda se ceo sistem odbacuje i traži njegova popravka.

Dok kod funkcionalnih zahteva, najčešće se može utvrditi koja sistemska komponenta obezbeđuje traženu funkciju, kod nefunkcionalnih zahteva to nije tako jasno. Implementacija nefunkcionalnih zahteva zahvata mnoge delove sistema jer:

- Nefunkcionalni zahtevi mogu da *utiču na celu arhitekturu sistema*, a ne na pojedinačne komponente sistema.
- Jeden pojedinačni nefunkcionalni zahtev (npr zahtev bezbednosti), *može da generiše niz povezanih funkcionalnih zahteva* koji definišu nove servise sistema i koje realizuje mnogo komponenti sistema.

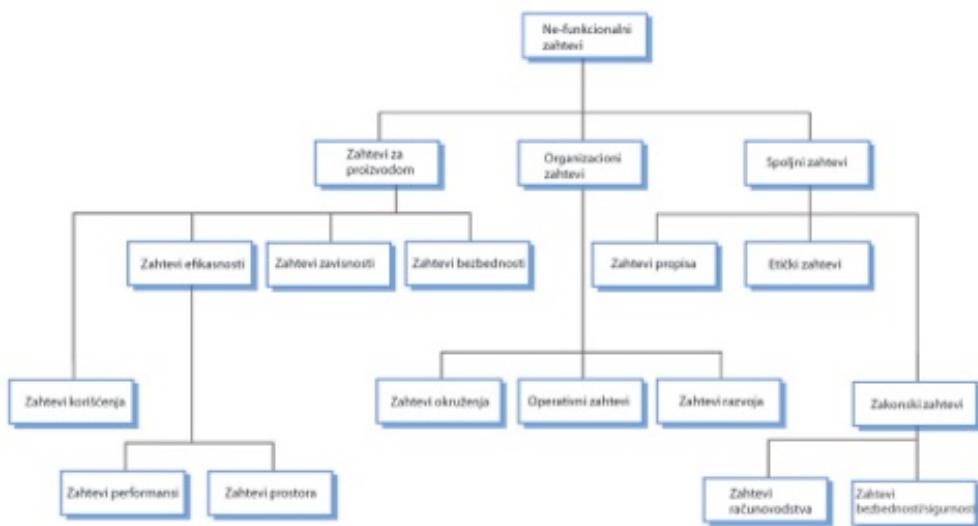
Nefunkcionalni zahtevi su rezultat potreba korisnika, kao što su ograničenje budžeta, organizaciona politika, zahtevi za interoperabilnošću sa drugim sistemima, ili spoljni faktori, kao što su propisi bezbednosti.

## VRSTE NEFUNKCIONALNIH ZAHTEVA

*Osnovne vrste nefunkcionalnih zahteva su: zahtevi proizvoda, organizacioni zahtevi i spoljni zahtevi.*

Na slici 1 su prikazane vrste nefunkcionalnih zahteva. Može se uočiti nekoliko osnovnih grupa nefunkcionalnih zahteva:

1. **Zahtevi proizvoda:** Ovi zahtevi opisuju ili ograničavaju ponašanje softvera (npr, performanse).
2. **Organizacioni zahtevi:** Ovi zahtevi su sistemski zahtevi koji su rezultat organizacionog ustrojstva kupca sistema (npr. poslovni procesi i standardi)
3. **Spoljni zahtevi:** Oni odražavaju zahteve koji dolaze iz spoljnog okruženja sistema (npr., propisi)



Slika 1.2.1 Vrste nefunkcionalnih zahteva

## PRIMER NEFUNKCIONALNIH ZAHTEVA SISTEMA MHC-PMS

*Zahlev proizvoda specificiraju ponašanje softvera, zahtevi organizacija - su sistemski zahtevi organizacije korisnika, a spoljni zahtevi su zahtevi okruženja.*

Na slici 2 prikazani je primer nefunkcionalnih zahteva (projekt MHC-PMS).

**ZAHTEVI PROIZVODA**  
MHC-PMS treba da bude raspoloživ svim klinikama za vreme svog normalnog radnog vremena (ponedeljak-petak, od 8.30 do 17.30). Nijedan prekid u tom vremenu ne sme da traje duže od pet sekundi.

**ZAHTEVI ORGANIZACIJE**  
Korisnici MHC-PMS sistema se identificuju sistemu upotrebom svojih zdravstvenih identifikacionih kartica.

**SPOLJNI ZAHTEV**  
Sistem treba da obezbedi privatnost pacijenata u skladu sa Hstan-03-2006-priv.

Slika 1.2.2 Primer nefunkcionalnih zahteva sistema MHC-PMS

Čest problem kod definisanja nefunkcionalnih zahteva opštost zahteva koje daju korisnici. Tumačenje uopštenih zahteva može biti različito, što onda dovodi do sistema koji se možda neće ponašati onako kao su korisnici očekivali.

Na primer, neki menadžer može postaviti sledeći zahtev:

*„Sistem bi trebalo da bude lak za upotrebu od strane zdravstvenih radnika i trebalo bi da bude tako organizovan da greške korisnika budu minimalne“*

Taj uopšteni zahtev bi trebalo konkretizovati, da bi se mogao da meri njegova implementacija. Na primer, on se može i ovako definisati:

*„Medicinsko osoblje trebalo bi da koriste sistem posle četiri sata obuke. Posle obuke, prosečan broj grešaka koje čine iskusni korisnici sistema ne bi trebalo da pređe dve po satu rada sistema.“*

## METRIKA NEFUNKCIONALNIH ZAHTEVA

*Metrika se može koristiti kod specifikacije nefunkcionalnih zahteva.*

Svuda gde je mogućno, pri definisanju nefunkcionalnih zahteva, potrebno je navesti kvantifikovane pokazatelje, koji se onda mogu testiranjem proveriti. Slika 3 pokazuje metriku koja se može koristiti kod specifikacije nefunkcionalnih zahteva. Navedene karakteristike se mogu meriti za vreme testiranja, te im i proveriti da li sistem zadovoljava nefunkcionalne zahteve.

Nije uvek lako koristiti merljiva svojstva prilikom definisanje nefunkcionalnih zahteva. Na primer, lakoća održavanja, teško se može kvantifikovano izraziti. Ponekad, i kada je ovo moguće, korisnici možda ne mogu da odrede poželjne vrednosti. Ponekad, cena ispunjenje nekog zahteva može biti isuviše velika za korisnika, te i o tome se mora voditi računa pri definisanju nefunkcionalnih zahteva.

Svojstvo	Mera
Brzina	Broj transakcija/sekunda Korisnik/Vreme odziva na događaj Vreme osvežavanja monitora
Veličina	MB Broj ROM čipova
Lakoća upotrebe	Vreme obuke Broj frejmova sa pomoć informacijama
Pouzdanost	Srednje vreme otkaza Verovatnoća otkaza rada sistema Učestanost pojavljivanja otkaza Raspoloživost sistema
Robusnost	Vreme ponovnog rada sistema posle otkaza Procenat događaja koji dovode do prekida rada sistema Verovatnoće oštećenja podataka prilikom prekida rada sistema
Prenošljovost	Procenat iskaza koji su zavisni od cilja Broj ciljnih sistema

## KONTRADIKTORNOST NEFUNKCIONALNIH ZAHTEVA

*Nefunkcionalni zahtevi mogu da bude kontradiktorni, tj. u sukobu sa drugim funkcionalnim ili nefunkcionalnim zahtevima*

Nefunkcionalni zahteva mogu da bude kontradiktorni, tj. u sukobu sa drugim funkcionalnim ili nefunkcionalnim zahtevima. Na primer, zahtev za identifikaciju korisnika na slici 2 može da zahteva čitač kartica koji se zbog toga mora instalirati na svakom personalnom računaru koji je povezan sa sistemom. Međutim, neki zahtevi mogu zahtevati mobilne uređaje za pristup sistemu, a oni uglavnom ne sadrže čitače kartica, te se moraju primeniti neki drugi metodi identifikacije korisnika.

U praksi, nije uvek lako odvojiti funkcionalne i nefunkcionalne zahteve u dokumentu koji definiše zahteve. Dobro je se u dokumentu ukaže na njihovu međuzavisnost. To se može uraditi i tako što se takvi zahtevi zajedno prikazuju u posebnom poglavlju u dokumentu.

Nefunkcionalni zahtev, kao što su pouzdanost, bezbednost i poverljivost, su od posebnog značaja za tzv. kritične sisteme.

## ZADATAK 3

### *Definišite nefunkcionalne zahteve za sistem predložen u zadatku 1*

1. Definišite nefunkcionalne zahteve za sistem definisan u zadatku 1.
2. U kojim situacijama se javlja kontradiktornost nefunkcionalnih zahteva.

## ✓ 1.3 Utvrđivanje zahteva

### ŠTA JE UTVRĐIVANJE ZAHTEVA?

*Utvrđivanje zahteva je proces prikupljanja informacija o zahtevanom sistemu, ali i o postojećim sistemima, i izdvajanje zahteva korisnika i sistema iz ovih informacija.*

Utvrđivanje zahteva (engl. requirements discovery, ili elicitation) je proces otkrivanja i prikupljanja informacija o zahtevanom sistemu, ali i o postojećim sistemima, i izdvajanje (destilacija) zahteva korisnika i sistema iz ovih informacija. Izvori informacija su: dokumentacija, akteri sistema, i specifikacija sličnih sistema. Interakcija sa sistemskim akterima se odvija u vidu intervjeta (razgovora), osmatranjem, a mogu s koristiti i scenariji i prototipovi, kako bi se pomoglo akterima da razumeju kakav sistem bi trebalo da bude.

Aktere čine različite kategorije ljudi, od krajinjih korisnika sistema, pa do menadžera spoljnih aktera, kao što su regulacione institucije, koje daju potvrde za prihvatljivost sistema. Na primer, u slučaju informacionog sistema institucije koja leči pacijente sa mentalnim problemima (MHC-PMS sistem), akteri sistema su:

- pacijenti,
- doktori koji leče pacijente,
- medicinske sestre, koje administriraju neke terapije i koordinišu konsultacije sa doktorima,

- zdravstveni recepcionisti, koji zakazuju pregledе pacijenata,
- IT inženjeri i tehničari, koji instalishu i održavaju sistem,
- menadžer za medicinsku etiku, koji se stara o poštovanju etičkih principa u radu sa pacijentima,
- menadžeri u zdravlju koji koriste informacije iz sistema,
- zaposleni koji vode arhivu zdravstvenih kartona pacijenata i koji se brinu o održavanju i zaštititi tih informacija i procedura za pristup tim informacijama.

Pored aktera sistema, izvor za otkrivanje zahteva je domen primene i slični sistemi. Različiti izvori pružaju i različite poglede na sistem. Svaki od njih pokazuje odgovarajući podskup zahteva sistema. Svaki pogled na problem vidi problem na drugačiji način (iz svog ugla). Često se ti pogledi delimično preklapaju jer ukazuju na zajedničke zahteve. Pogledi se mogu koristiti i za strukturisanje i za dokumentovanje zahteva sistema

## ✓ Poglavlje 2

### Dokument sa zahtevima

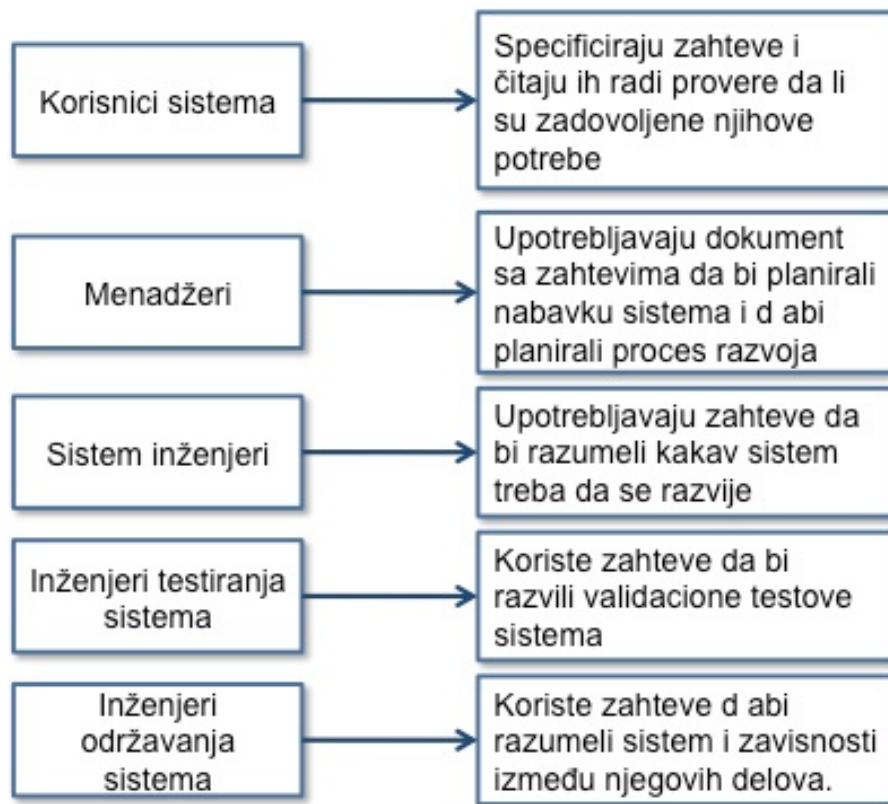
#### ŠTA JE DOKUMENT SA ZAHTEVIMA?

*Dokument sa zahtevima za softver je zvanični dokument koji razvojni tim treba da primeni. On uključuje i zahteve korisnika i detaljnu specifikaciju zahteva za sistem.*

Dokument sa zahtevima za softver (ili specifikacija softverskih zahteva) je zvanični dokument koji razvojni tim treba da primeni. On uključuje i zahteve korisnika, i detaljnju specifikaciju zahteva za sistem. Ponekad i korisnički i sistemski zahtevi se navode u istom dokumentu, tj. sa istim opisom. Ponekad, se oni prikazuju u odvojenim dokumentima.

Dokument sa zahtevima je neophodan kao prilog ugovora po kom se angažuju organizacije koje su podizvođači u razvojnom projektu. Međutim, kod primene metoda agilnog razvoja, ovaj dokument se najčešće i ne koristi. Kod Ekstremnog programiranja, zahtevi se definišu inkrementalno i zapisuju na karticama kao korisničke priče ili scenariji. Kod poslovnih aplikacija, koje nemaju stabilne zahteve, to je i dobra praksa. Ipak, poželjno je i u tim slučajevima sačiniti dokument o poslovnim zahtevima i zavisnostima, i o tome voditi računa kada se zahtevi parcijalno i inkrementalno primenjuju u toku inkrementalnog razvoja sistema.

Dokument o zahtevima ima različite korisnike, od rukovodstva firme, do inženjera razvoja (slika 1 ).



Slika 2.1 Korisnici dokumenta sa zahtevima (Kotonya, Sommerville 1998)

## STRUKTURA DOKUMENTA SA ZAHTEVIMA

*IEEE definiše standardnu strukturu dokumenta koji sadrži sve zahteve koje softverski sistem treba da zadovolji.*

Zbog različitosti korisnika dokumenta, onda mora da bude napisan tako da zadovolji sve njihove potrebe. Dovoljno precizan i detaljan za inženjere razvoja i testiranja, a i da sadrži informacije neophodne za evoluciju softvera.

Novo detaljisanja zahteva zavisi od vrste sistema koji se razvija. Kritični sistemi treba da imaju detaljne zahteve. Kada se sistem razvija u drugoj kompaniji, dokument sa zahtevima mora da bude detaljan i precizan. Ako se radi unutar firme, o ako se primenjuje iterativan razvoj, dokument može biti manje detaljan jer se može razjašnjavati za vreme razvoja siste

U Tabeli 1 prikazan je jedan od organizovanja dokumenta sa zahtevima koji se bazira na IEEE standardu za dokument sa zahtevima (IEEE; 1998). Taj standard je opšti, i može se prilagoditi specifičnim upotrebam.

Odeljak	Opis
Predgovor	Trebalo bi da definije očekivanu upotrebu dokumenta i da opiše svoju istoriju, uključujući i razlog postojanja nove verzije i rezime promena koje je svaka verzija donela.
Uvod	Opisuje potrebe za sistemom. Sadrži kratak opis funkcija sistema i objašnjava kako radi sa drugim sistemima. Opisuje i kako se sistem uklapa u celokupan posao ili strategijske ciljeve organizacije koja je naručila softver.
Rečnik	Objašnjava tehničke termine koji se koriste u dokumentu. Ne smete da pravite prepostavke oko iskustva ili znanja čitaoca.
Definicija zahteva korisnika	Opisuju se servisi koji se obezbeđuju za korisnika. Daju se i nefunkcionalni zahtevi. Koristi se svakodnevni jezik, dijagrami i druge oznake koje su razumljive korisnicima. Specificiraju se standardi koji proizvod i proces treba da zadovolji.
Arhitektura sistema	Predstavlja uopšten prikaz arhitekture sistema, pokazuje distribuciju funkcija po modulima sistema. Posebno se navode komponente koje su višestruko upotrebljive (preuzete iz arhive)
Specifikacija zahteva sistema	Detaljnije opisuje funkcionalne i nefunkcionalne zahteve. Ako je potrebno, dodaju se i ranije ne navedeni nefunkcionalni zahtevi, kao i interfejsi prema drugim sistemima.
Modeli sistema	Uključuje grafičke modele koji pokazuju relacije između komponenata sistema, sistem, i okruženja. Primeri mogućih modela su objekti modela, modeli toka podataka, i semantičkih modela podataka.
Evolucija sistema	Opisuje prepostavke na kojim se sistem razvija, navodi promene koje se očekuje u hardveru, potrebama korisnika, i dr. Ovo koriste sistem inženjeri kako bi izbegli rešenja koja bi ograničila buduće očekivane promene sistema.
Dodaci	Sadrži detaljne informacije o sistemu koji se razvija, kao što je opis hardvera i baze podataka. Zahtevi hardvera sadrže minimalne i optimalne konfiguracije sistema. Zahtevi baza podataka sadrže logičku organizaciju podataka i relacije između podataka.
Indeksi	Može se koristiti nekoliko indeksa u dokumentu.: Alfabetski indeks, indeks dijagrama, indeks funkcija i dr.

## ▼ Poglavlje 3

### Specifikacija zahteva

#### ŠTA JE SPECIFIKACIJA ZAHTEVA?

*Specifikacija zahteva je proces pisanja korisničkih i sistemskih zahteva u dokument sa zahtevima*

Specifikacija zahteva je proces pisanja korisničkih i sistemskih zahteva u dokument sa zahtevima. *U idealnom slučaju, ovi zahtevi bi trebalo da budu: jasni, nedvosmisleni, lako razumljivi, kompletni, i konsistentni.* To je u praksi teško ostvarljivo jer zainteresovani akteri tumače zahteve na različite načine, te skoro uvek imamo suprostavljene zahteve i njihovu nekonsistentnost.

Zahtevi korisnika bi trebalo da opišu funkcionalne i nefunkcionalne zahteve, tako da su razumljiviji od strane korisnika softvera koji nemaju detaljna tehnička znanja. Idealno, *ovi zahtevi bi trebalo da opišu samo spoljnje ponašanje sistema.* Dokument sa zahtevima ne bi trebalo da uključi detalje o arhitekturi sistema niti o projektnom rešenju sistema. **Zato se zahtevi korisnika pišu u običnom jeziku korisnika, sa jednostavnim tabelama, i intuitivnim dijagramima.**

Zahtevi sistema su proširene verzije zahteva korisnika i koriste se od softverskih inženjera kao početna tačka projektovanja sistema.

**Zahtevi sistema** dodaju detalje i objašnjavaju **kako sistem treba da realizuje zahteve korisnika.** Oni mogu da se iskoriste i kao deo ugovora za implementaciju sistema, te moraju da budu kompletni i da predstavljaju detaljnu specifikaciju celog sistema. Iako zahtevi sistemi, po teoriji, ne bi trebalo da određuju kako će sistem biti projektovan i kako će ih sistem realizovati, iz sledećih razloga, nije moguće da se potpuno isključi informacija o arhitekturi iz specifikacije zahteva:

1. **Početna arhitektura sistema** pomaže strukturisanju specifikacije zahteva. Zahtevi sistema su organizovani po pod-sistemima koji čine sistem.
2. Najčešće, sistem treba da komunicira sa drugim, postojećim sistemima, koji ograničavaju projektno rešenje i postavljaju dodatne zahteve novom sistemu.
3. Da bi se zadovoljili neki nefunkcionalni zahtevi, neophodno je **upotrebiti specifičnu arhitekturu**, te je zato **potrebno nju navesti u specifikaciji zahteva.** Na primer, spoljna kontrola poštovanja nekog propisa ili standarda će potvrditi da je sistem bezbedan ako se koristi određena arhitektura, prikladna za zadovoljenje tih propisa bezbednosti.

## NAČINI PISANJA SISTEMSKIH ZAHTEVA

*Zahtevi sistema, pored korišćenja običnog jezika, mogu da koriste i druge načine pisanja, kao što su grafički modeli ili matematički sistemi, a da bi detaljnije objasnile sistemske zahteve.*

Za razliku od zahteva korisnika, zahtevi sistema, pored korišćenja običnog jezika, mogu da koriste i druge načine pisanja, kao što su grafički modeli ili matematički sistemi, a da bi detaljnije objasnile zahteve koje sistem treba da ispunji. Grafički modeli su od posebne pomoći kada se treba objasniti kada sistem treba da promeni svoje stanje ili kada se treba navesti neki redosled akcija koje sistem treba da realizuje. Unified Modelling Language (UML) se u ove svrhe najčešće koristi. Matematički sistemi se ponekad koriste kod bezbednostno kritičkih sistema, ali se retko koriste u drugim slučajevima.

U Tabeli 1 navedeni su načini za pisanje specifikacije zahteva sistema.

Notacija (način pisanja)	Opis
Rečenice govornog jezika	Zahtevi si pišu upotrebom numerisanih rečenica primenom običnog, govornog jezika. Svaka rečenica se odnosi na jedan zahtev.
Strukturiran govorni jezik	Zahtevi se upisuju u standardni formular korišćenjem običnog, govornog jezika.
Jezik za opisivanje projektnog rešenja (dizajna)	Ovaj pristup koristi neki jezik, kao što može biti i programski jezik, ali sa većim nivoom apstrakcije svojstava, radi definisanja zahteva nekog operativnog modela sistema. Ovaj način se sada retko koristi, sem donekle, kod specifikacije interfejsa.
Grafička notacija	Grafički modeli, sa dodatnim tekstualnim komentarima, se koriste pri definisanju funkcionalnih zahteva sistema. Najčešće se koriste UML dijagrami – slučajevi upotrebe (engl. Use case) i sekvenčni dijagrami.
Matematička specifikacija	Najšće se koristi za opis maština konačnih stanja ili skupova. Iako primena matematički zasnovane formalne specifikacije smanjuje nejasnoće u zahtevu, ređe se koristi, jer najveći broj naručiova softvera ne rezume ova formalne metode.

## SPECIFIKACIJA ZAHTEVA OBIČNIM JEZIKOM

*To može biti samo jedna rečenica po zahtevu, koja objašnjava i razlog za postavljanje zahteva, a može sadržati i informaciju ko je tražio taj zahtev.*

Primena običnog, govornog jezika u specifikaciji zahteva je izražajna, intuitivna i univerzalna. Široko se koristi, što može da dovodi do nejasnoća jer shvatanje značenja pojedinih termina zavisi od obrazovanja i znanja onoga koji čita tekst specifikacije. Da bi se minimizirali ovi nedostaci, daju se sledeće preporuke za korišćenje:

Treba koristiti neki standardni format (oblik) za definisanje zahteva. To može biti samo jedna rečenica po zahtevu, koja objašnjava i razlog za postavljanje zahteva, a može sadržati i informaciju ko je tražio taj zahtev.

1. Jasno navedite razliku između obaveznih i poželjnih zahteva.
2. Podvlačenjem reči (ili koristite iskošena slova, zadebljana slova) istaknuti ključne delove zahteva.

3. Ne koristite tehničke termine, tehnički žargon, skraćenice i slično, jer može uvesti do nerazumevanja.
4. Svuda gde je moguće, navesti razlog za postavljanje zahteva.

Slika 1 pokazuje primer primene ovih preporuka u slučaju sistema MHC-PMS (videti opis ove studije slučaja date u 1. lekciji)

3.2 Sistem treba da meri šećer u krvi i da isporuči insulin, ako je potrebno, na svakih 10 minuta. (promena šećera u krvi je relativno spora t ečešća merenja nisu potrebna; češća merenja bi mogla da dovedu do nepotrebnih visokih nivoa šećera u krvi).

3.6 Sistem će aktivirati testiranje samog sebe svakog minuta pod uslovima testiranja i izvršiće akcije date u Tabeli 1 (Rutina samo-testiranja može da otkrije hardverske i softverske probleme i da upozori korisnika da normalni uslovi rada nisu mogući).

Slika- 1 Primer sistemskih zahteva slučaja sa insulin pumpom

## SPECIFIKACIJA ZAHTEVA STRUKTURISANOM FORMOM

*Strukturisana specifikacija koristi formulare (uzorke) za pisanje zahteva sistema.*

Strukturisan običan jezik je način pisanja zahteva koji ograničava slobodu pisanja pisca zahteva i kod koga se svi zahtevi pišu na jedan standardizovani način. Na ovaj način se zadržava izražajnost i razumljivost običnog jezika, ali se obezbeđuje i da određeni nivo uniformnosti specifikacije. **Strukturisana specifikacija koristi formulare (uzorke) za pisanje zahteva sistema.** Specifikacija može da koristi i iskaze u programskom jeziku da bi se prikazale alternative i iteracije, a može i da označi ključne delove teksta podvlačenjem zadebljanjem slova, promenom fonta (tip slova) u delu teksta ili senčenjem.

Može se koristiti jedan ili više standardnih formulara za definisanje sistemskih zahteva. Oni se mogu strukturirati prema sistemu, funkciji koju sistem ostvaruje, ili prema događaju koju obrađuje sistem.

U TAbeli 2 je prikazan jedan primer strukturisanog načina definisanja jednog sistemskog zahteva u slučaju sistema MHC-PMS

**Insulin pumpa(Softver upravljanja/SRS/5.3.2**

<b>Funkcija:</b>	Proračun doze insulina: Siguran nivo šećera
<b>Opis:</b>	Računa dozu insulina za davanje kada tekući izmereni nivo šećera je od 3. do 7. jedinice.
<b>Ulazi:</b>	Trenutno očitana vrednost šećera ( $r_2$ ), prethodna dva merenja ( $r_0$ i $r_1$ ).
<b>Izvor:</b>	Trenutno izmerena vrednost šećera dobijena od senzora. Čitanja se dobijaju iz memorije.
<b>Izlazi:</b>	CompDose – doza insulina koja se treba dati pacijentu
<b>Destinacija:</b>	Glavna upravljačka petlja
<b>Akcija:</b>	CompDose je nula ako je nivo šećera stabilan i ako opada ili ako se povećava ali brzina porasta opada. Ako se nivo povećava i ako se brzina uvećavanja povećava, ondase CompDose računa deljenjem razlike trenutno očetanje vrednosti i prethodno očitane vrednosti sa 4, sa zaokruženjem rezultata. Ako je zaokruženi rezultata nula, onda se daje predviđena minimalna doza za CompDose.
<b>Zahtevi:</b>	Dva prethodna čitanja dabi se izračunala brzina promene nivoa.
<b>Preduslov:</b>	Rezervoar insulina sadrži najmanje količinu insulina koja odgovara jednoj, ali maksimalnoj dozi insulina.
<b>Izlazni uslov:</b>	$r_0$ se zamenjuje sa $r_1$ , a onda $r_1$ se zamenjuje sa $r_2$ .
<b>Uzgredni efekti:</b>	Nema

## STRUKTURISANA SPECIFIKACIJA PRIMENOM STANDARDZOVANOG FORMULARA

*Primenom strukturisanog načina definisanja zahteva smanjuje se njihova promenljivost, a zahtevi su efektivnije organizovani*

Kada se standardizovani formular koristi za specifikaciju funkcionalnih zahteva, on bi trebalo da sadrži sledeće informacije:

1. Opis funkcije ili specificiranog entiteta.
2. Opis ulaza i izvora odakle dolaze.
3. Opis izlaza i naznake gde idu.
4. Neophodne informacije za proračun ili druge entitete u sistemu a koji se koriste.
5. Opis akcije koja se preduzima.
6. Ako se primenjuje funkcionalni pristup, postavljen preduslov mora da bude ispunjen pre nego što se funkcija izvrši, a specificiran izlazni kriterijum se mora da ispuni posle realizacije funkcije.
7. Opis uzgrednih efekata operacije (ako ih ima).

Primenom strukturiranog načina definisanja zahteva smanjuje se njihova promenljivost, a zahtevi su efektivnije organizovani. Međutim, ponekad i to ne pomaže da se ne jave nejasne i dvosmisleni zahtevi, naročito kod složenijih proproračuna. U tim slučajevima mogu se dodati dodatna objašnjenja, koristeći običan jezik, ili upotrebljavajući tabele ili grafičke modele radi prikaza toka proračuna, ili prikazivanja načina promene stanja sistema, ili načina interakcije korisnika i sistema, ili redosleda akcija koje treba izvršiti. Slika 4 prikazuje primer tabelarnog opisa kako se brzina promene šećera u krvi koristi za računjanje potrebne doze insulina koje treba dati pacijentu, a za slučaj utvrđivanja zahteva u sistemu MHC-PMS

Uslov	Akcija
Nivo šećera pada ( $r_2 < r_1$ )	ComDose = 0
Nivo šećera je stabilan ( $r_2 = r_1$ )	CompDose = 0
Nivo šećera raste a brzina porasta opada ( $(r_2 - r_1) < (r_1 - r_0)$ )	CompDose = 0
Nivo šećera raste, i brzina porasta raste ili je stabilna	CompDose = zaokruži $((r_2 - r_1)/4)$ Ako je rezultat zaokružavanja = 0 onda CompDose = MinimumDose

## ZADATAK 4

### *Formulisanje zahteva sistema za prodaju železničkih karata*

- Otkrij nejasnoće ili nedostajuće informacije u sledećem iskazu o zahtevima za deo sistema za prodavanje karata: .
 

*"Jedan automatski sistem za prodaju karata prodaje vozne karte na železnici. Korisnici biraju mesto u koje žele da putuju i daju svoju kreditnu karticu i svoj jedinstveni matični broj. Sistem izdaje kartu korisniku, i skida mu sa računa novac u skladu sa cenom vozne karte. Kada korisnik pritisne START dugme, meni na monitoru pokazuje moguće destinacije, tj. mesta za koje je obezbeđen prevoz vozovima, sa porukom koja daje instrukcije korisniku kako da izbere svoju destinaciju - mesto putovanja. Po izboru destinacije, traži se od korisnika da unese svoju kreditnu karticu. Proverava se njena validnost i onda se traži od korisnika da unese svoj PIN. Nakon validacije kreditne kartice, sistem izdaje kartu korisniku. ."*
- Napiši gornji opis primenom strukturisanog pristupa, kao što je dato na predavanju. Na dogovarajući način reši navedene nejasnoće i nedostatke.
- Napiši skup nefunkcionalnih zahteva za sistem za prodaju karata, uzimajući u obzir očekivanu pouzdanost i vreme odziva sistema.

## ZADATAK 5

### *Definisati zahteve za sistem za prodaju goriva sa samoposluživanjem*

Upotrebom tehnika datih na predavanju, primenom prirodnog (govornog) jezika izraženog u standardnoj formi, napiši pouzdane zahteve korisnika za sledeće funkcije:

- Stanica za podrazu benzina sa samoposluživanjem. Korisnik ubacuje kreditnu karticu u čitač kartica i ukucava sumu za koju želi da kupi gorivo.
- Pumpa onda sipa gorivo u rezervoar njegovog automobila i skida navedenu sumu sa računa kupca, primenom funkcije za plaćanje keša bankovnog ATM-a.
- Vrši se provera grešaka u kucanju i vši se korekcija funkcije u processoru za analizu ispravnosti kucanaja teksta.

## ✓ Poglavlje 4

# Procesi inženjerstva zahteva

## ŠTA JE PROCES INŽENJERSTVA ZAHTEVA?

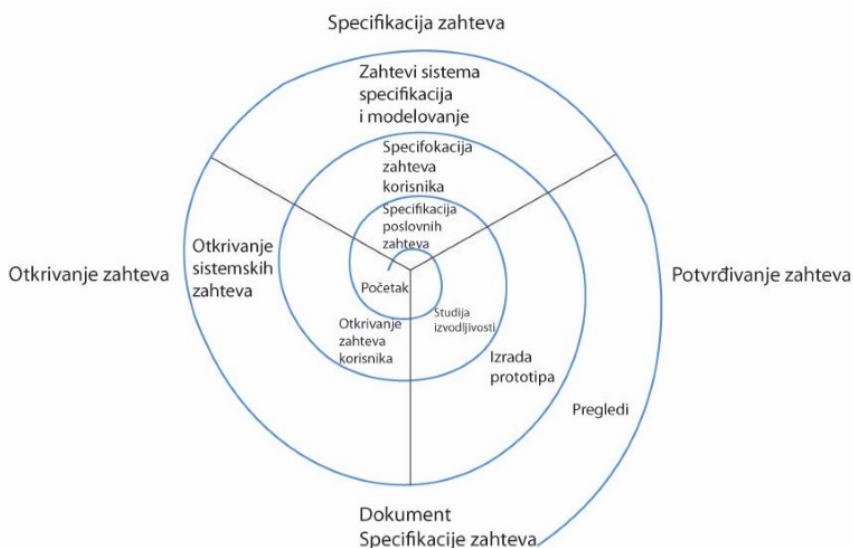
*Procesi inženjerstva zahteva čine četiri osnove aktivnosti: studija izvodljivosti, prikupljanje i analiza zahteva, specifikacija zahteva i validacija.*

Procesi inženjerstva zahteva čine četiri osnove aktivnosti:

1. Studija izvodljivosti – ocena da li je proces koristan za poslovanje,
2. Prikupljanje i analiza zahteva – otkrivanje zahteva
3. Specifikacija zahteva – konverzija zahteva u standardni oblik
4. Validacija – provera da zahtevi odražavaju sistem koji kupac/korisnik želi

U praksi, inženjerstvo zahteva je jedan iterativan proces, a ne strogo redni (sekvencijalan). Na slici je prikazan ovaj proces kao iterativni proces u vidu spirale, sa konačnim izlazom u vidu dokumenta sa specifikacijom zahteva sistema. U svakoj vremenskoj fazi troši se u svakoj aktivnosti odgovarajući rad i vreme.

U početku se najviše vremena troši na razumevanje glavnih poslovnih i nefunkcionalnih zahteva, kao i zahteve korisnika sistema. Spiralni model inženjerstva zahteva da se uzime u obzir različit nivo detaljnosti zahteva. Sa povećanjem broja iteracija oko spirale, i broj i detaljnost zahteva se povećava. Ako se koristi agilni pristup razvoju, umesto izrade prototipa, zajedno se radi utvrđivanje zahteva i implementacija sistema.



Slika 4.1 Spiralni pogled na proces inženjerstva zahteva

## ŠTA JE PROCES INŽENJERSTVA ZAHTEVA? (NASTAVAK)

*Primena grafičkih modela nije dovoljna za inženjerstvo zahteva. Pri prikupljanju zahteva, ljudska aktivnost je dominantna.*

Pri utvrđivanju zahteva mogu se da koristi strukturisani metod analize, koje se oslanja na primenu grafičkih metoda, kao što je analiza slučajeva upotrebe sistema (engl. use case models), radi izrade specifikacije zahteva sistema. Skup tih metoda opisuju ponašanje sistema pri čemu se dodaju dodatne informacije, kao što su to zahtevane performanse ili zahtevana pouzdanost sistema.

Međutim, primena grafičkih modela nije dovoljna za inženjerstvo zahteva. Na primer, pri prikupljanju zahteva, ljudska aktivnost je dominantna, jer ljudi obično ne vole da budu ograničeni primenom rigidnih modela sistema.

### Studija izvodljivosti

Studija izvodljivosti je analiza koja se realizuje u ranim fazama procesa inženjerstva zahteva. Ona treba da odgovori na tri ključna pitanja:

1. Da li softverski sistem u razvoju treba da doprinese ukupnim ciljevima poslovanja organizacije?
2. Da li se sistem može razviti sa planiranim budžetom za projekat razvoja i u okviru planiranog roka?
3. Da li se novi sistem može da integriše sa ostalim sistemima koji se koriste u organizaciji.

Ukoliko je neki od odgovorana ova tri pitanja negativan, ne bi trebalo da nastavite ovaj projekat razvoja,

## ✓ Poglavlje 5

# Utvrđivanje i analiza zahteva

## OPŠTI MODEL UTVRĐIVANJA I ANALIZE ZAHETVA

*Ova aktivnost utvrđuje domen primene softvera, koje bi servise softvera trebalo da obezbedi, zahtevane performanse, hardverska ograničenja i dr.*

U ovoj aktivnosti procesa inženjerstva zahteva, softverski inženjeri rade sa korisnicima/kupcем softvera da bi utvrdili domen primene softvera (aplikacije), koje servise bi sistem trebalo da obezbedi, zahtevane performanse sistema, hardverska ograničenja i dr. Ovo zahteva kontakt se vrlo različitim ljudima u organizaciji korisnika sistema. Svi koji imaju neki direktni ili indirektni interes na zahteve sistema se nazivaju akterima sistema (engl. stakeholders). To mogu biti krajnji korisnici sistema koji u interakciji sa sistemom obavljaju svoj posao, a mogu biti i oni koji sam zavise od sistema, iako ga direktno ne koriste. (poslovni menadžeri. Eksperti domena, predstavnici sindikata, inženjeri razvoja ili održavanja sistema).

Na slici je prikazan opšti model prikupljanja i analize sistema Svaka organizacija kreira svoj specifičnu verziju ovog modela, zavisno od lokalnih faktora, kao što je znanje zaposlenih, tip sistema koji se razvija, standardi koji se koriste i dr.

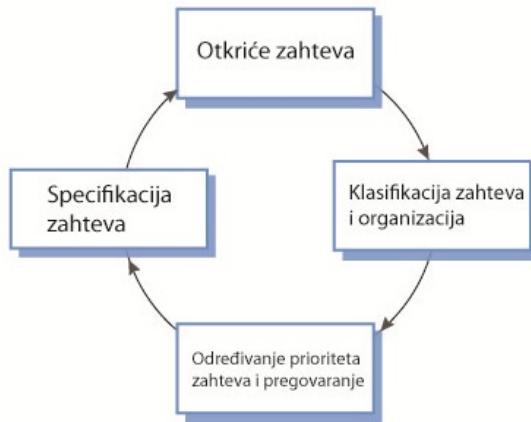
Aktivnosti procesa utvrđivanja i analize zahteva su sledeće:

**1. Otkriće zahteva:** To je aktivnost komuniciranja sa akterima sistema. Utvrđuju se zahtevi domena od ovih aktera i dokumentacija.

**2. Klasifikacija i organizacija zahteva:** Aktivnost grapiše prikupljene nestrukturisane zahteve, i organizuje ih u koherentan klaster. Obično se zahtevi grapišu po podsistemima i modulima arhitekture sistema. Zato se u praksi paralelno radi na prikupljanju zahteva i izradi arhitekture sistema.

**3. Određivanje prioriteta zahteva** i pregovaranje:

**4. Specifikacija zahteva:** Zahtevi se dokumentuju i koristi se za ulaz u sledeću aktivnost spirale.



Slika 5.1.1 Prikupljanje i analiza zahteva

## POTEŠKOĆE U PRIKUPLJANJU ZAHTEVA

*Da bi se neki suprotstavljeni zahtevi razjasnili, organizuju se sastanci sa akterima sistema na kojima se usaglašavaju stavovi oko zahteva*

Posle više ciklusa duž spirale modela, dolazi se do kraja, tj. do dokumenta sa specifikacijom zahteva.

Prikupljanje zahteva je otežano iz sledećih razloga:

1. Sistemski akteri često ne znaju da definišu zahteve dovoljno detaljno i jasno.
2. Akteri sistema definišu zahteve koristeći svoje termine i svoje implicitno znanje. Možda inženjeri razvoja, ne razumeju te termine.
3. Različiti akteri imaju različite zahteve i izražavaju ih na različite načine. Inženjer koji prikuplja zahteve mora da otkrije sve potencijalne izvore zahteva i da otkrije i konfliktne zahteve.
4. Politički faktori mogu da imaju uticaje na zahteve sistema. Neki menadžeri mogu da nešto zahtevaju samo da bi povećali svoj uticaj u organizaciji.
5. Ekonomsko i poslovno okruženje u kome se prikupljaju zahtevi je dinamično. Ono se menja za vreme analize zahteva. Važnost nekog zahteva može da se promeni. Mogu se javiti i novi zahtevi od strane novih aktera, koji ranije nisu bili konsultovani.

Da bi se neki suprotstavljeni zahtevi razjasnili, organizuju se sastanci sa akterima sistema na kojima se usaglašavaju stavovi oko zahteva. To je neka vrsta pregovaranja, jer se često traže kompromisi.

U toku ovog procesa, rade se rane verzije dokumenta koji specificira zahteve sistema. Iako je to nezavršen dokument u razvoju, on je koristan jer ukazuje na preostale poslove koji se moraju uraditi oko prikupljanja zahteva. Pri tome, mogu se koristiti različite, privremene forme zapisivanja zahteva (tabele, kartice i dr.).

Praksa pokazuje da je prikupljanje zahteva korisnika vrlo težak i rizičan posao. Težak - zato što morate da razgovarate s mnogim ljudima i na slična pitanja iz jedne uske oblasti rada

sistema (šta bi sistema trebalo da radi npr. u službi marketina) dobijate vrlo različite, a često i sprostavljenje odgovore. Rizičan - jer ako dobro ne proverite relevantnost takvih zahteva, možete projektovati sistem za koji će korisnici kasnije reći da nije to ono što njima treba i odgovara, i traže izmene. Raditi izmene zahteva kad aje sistem već u fazi korisničkog prototipa je jako skupa operacija, jer mora da ponovite jedan dobar deo razvoja sistema.

## ŠTA JE UTVRĐIVANJE ZAHTEVA?

*Utvrdjivanje zahteva je proces prikupljanja informacija o zahtevanom sistemu, ali i o postojećim sistemima, i izdvajanje zahteva korisnika i sistema iz ovih informacija.*

Utvrdjivanje zahteva (engl. requirements discovery, ili elicitation) je proces otkrivanja i prikupljanja informacija o zahtevanom sistemu, ali i o postojećim sistemima, i izdvajanje (destilacija) zahteva korisnika i sistema iz ovih informacija. Izvori informacija su: dokumentacija, akteri sistema, i specifikacija sličnih sistema. Interakcija sa sistemskim akterima se odvija u vidu intervjua (razgovora), osmatranjem, a mogu s koristiti i scenariji i prototipovi, kako bi se pomoglo akterima da razumeju kakav sistem bi trebalo da bude.

Aktere čine različite kategorije ljudi, od krajinjih korisnika sistema, pa do menadžera spoljnih aktera, kao što su regulacione institucije, koje daju potvrde za prihvatljivost sistema. Na primer, u slučaju informacionog sistema institucije koja leči pacijente sa mentalnim problemima (MHC-PMS sistem), akteri sistema su:

- pacijenti,
- doktori koji leče pacijente,
- medicinske sestre, koje administriraju neke terapije i koordinišu konsultacije sa doktorima,
- zdravstveni recepcionisti, koji zakazuju pregledne pacijenata,
- IT inženjeri i tehničari, koji instaliju i održavaju sistem,
- menadžer za medicinsku etiku, koji se stara o poštovanju etičkih principa u radu sa pacijentima,
- menadžeri u zdravlju koji koriste informacije iz sistema,
- zaposleni koji vode arhivu zdravstvenih kartona pacijenata i koji se brinu o održavanju i zaštiti tih informacija i procedura za pristup tim informacijama.

Pored aktera sistema, izvor za otkrivanje zahteva je domen primene i slični sistemi. Različiti izvori pružaju i različite poglede na sistem. Svaki od njih pokazuje odgovarajući podskup zahteva sistema. Svaki pogled na problem vidi problem na drugačiji način (iz svog ugla). Često se ti pogledi delimično preklapaju jer ukazuju na zajedničke zahteve. Pogledi se mogu koristiti i za strukturisanje i za dokumentovanje zahteva sistema

## 5.1 Intervjui

### INTERVJUI I VRSTE INTERVJUA

*U intervjuima, tim inženjera postavlja pitanja akterima o sistemu koji oni trenutno koriste i o sistemu koji treba da se razvije. Analizom dobijenih odgovora, utvrđuju se zahtevi.*

Najveći broj zahteva se prikupi korišćenjem formalnih ili neformalnih intervjeta sa akterima sistema. U njima, tim inženjera postavlja pitanja akterima o sistemu koji oni trenutno koriste i o sistemu koji treba da se razvije. Analizom dobijenih odgovora, utvrđuju se zahtevi.

Postoje dve vrste intervjeta:

1. *Zatvoreni intervjui*, kada akteri odgovaraju na unapred pripremljena pitanja.
2. *Otvoreni intervjui*, kada ne postoje unapred definisana pitanja.

U praksi, intervjeti su najčešće mešavina obe vrste intervjeta. Unapred pripremljena pitanja obezbeđuju širinu prostora za prikupljanje informacija, a dobijeni odgovori daju priliku za postavljanje dodatnih pitanja, kako bi se utvrdile dodatne i nove informacije.

Intervjeti su dobri za razumevanje čime se akteri bave, kako bi mogli da komuniciraju sa sistemom, kao i za utvrđivanje poteškoća koje oni imaju sa sadašnjim sistemom. Međutim, intervjeti nisu tako korisni za razumevanje zahteva iz domena aplikacije, iz dva razloga

1. Svi specijalisti u nekom domenu aplikacije upotrebljavaju specifičnu terminologiju i žargon. Može se desiti da inženjeri koji rade intervju, loše shvate zahteve, jer ne razumeju dovoljno tu terminologiju.
2. Neki specijalisti su toliko bliski sa nekim domenskim znanjem, da smatraju da ne treba ni da ga pomenu, kao neki zahtev, „jer se podrazumeva“. Na taj način, lice koje vodi intervju može da ispusti iz vida zahtev koji nije eksplicitno naveden u intervjuu.

Odmah posle obavljenog intervjua, sačinite belešku i zapis šta vam je sagovornik rekao. Obavezno mu to dostavite na verifikaciju, je je moguće da ga niste u potpunosti razumeli ili pogrešno interpretirali. A često i on, ili doda nešto novo ili se predomislio u odgovorima koje je već dao.

### ZADATAK 6

*Navesti primer intervjeta*

1. Navesti primer intervjeta za proces utvrđivanja zahteva sistema definisanog u zadatu 1.

## ❖ 5.2 Upotreba scenarija

### ŠTA JE SCENARIO?

*Scenariji su opisi interakcije aktera i sistema, tj. interaktivnih sesija.*

**Scenariji** su opisi interakcije aktera i sistema, tj. interaktivnih sesija. Svaki scenario obično pokriva jednu ili mali broj mogućih interakcija. Postoje različite forme scenarija i one obezbeđuju različite tipove informacija na različitim nivoima detalja opisa sistema. Primena scenarija može biti posebno korisna u cilju dodavanja detalja prethodno definisanih zahteva.

Ako osoba koja vodi intervju, prikaže intervjuisanoj osobi scenario u kome opisuje očekivanu interakciju aktera sa sistemom, intervjuisana osoba će mnogo lakše izneti svoje sugestije i zahteve, nego ako se razgovor vodi apstraktno, bez početne skice scenario. Inženjer zahteva (osoba koja vodi intervju= na taj način prikupi korisne informacije koje mu pomažu da onda formuliše zahteve.

Scenario počinje skicom jedne interakcije aktera sa sistemom. Za vreme otkrivanja zahteva, vrši se dodavanja detalja da bi se dobila kompletan opis te interakcije. Jedan scenario, sadrži sledeće elemente:

1. **Opis šta sistem, i korisnici očekuju** kada počne scenario.
2. **Primarni scenario:** Opis normalnog toka događaja u scenario.
3. **Sekundarni scenario:** Opis šta može da ide pogrešnim tokom i kako taj problem razrešiti.
4. **Informacija o drugim aktivnostima** koje mogu da se istovremeno izvršavaju.
5. **Opis stanja sistema** kada se scenario završi.

Prikupljanje zahteva putem scenario zahteva rad sa akterima radi utvrđivanja scenario i radi prikupljanja detalja koje treba uključiti u scenario. Scenario se može pisati u vidu teksta, podržan dijagramima, slikama monitora i dr. Umesto scenario, mogu se koristiti scenario događaja ili slučajevi upotrebe (engl., use cases)

### PRIMER SCENARIJA PROJEKTA MHC-PMS

*Za svaki slučaj korišćenja, definiše se normalni (primarni) scenario i više (sekundarnih) scenario koji opisuju situacije do kojih može da se dođe nekim greškama*

Kada u kliniku uđe novi pacijent, otvara mu se e-karton na recepciji od strane medicinskog receptionera i upisuju mu se lični podaci u karton. Zatim sestra razgovara sa pacijentom i prikuplja medicinsku istoriju. Zatim, pacijent ima početnu konsultaciju sa doktorom koji određuje dijagnozu i, ako je potrebno, određuje terapiju. Scenario pokazuje šta se dešava kada se preuzme medicinska istorija.

**POČETNA PRETPOSTAVKA:** Pacijenta je primio medicinski recepcioner i za njega otvorio prijemni dokument u sistemu zajedno sa linim podacima o pacijentu. Medicinska sestra je uključena u sistem i prikuplja medicinski istorijat pacijenta.

**NORMALNI SCENARIO:** Sestra medicinsku dokumentaciju pacijenta prema njegovom prezimenu. Ako ima više pacijenata sa istim prezimenom, koristi se njegovo ime, i datum rođenja. Sestra izabira opciju meniju predviđenu za dodavanje medicinskog istorijata pacijenta. Sestra odgovara na niz pitanja sistema radi unošenja informacija i konsultacijama koje je pacijent imao u vezi njegovo mentalnog zdravlja (tekst slobodnog formata), postojećih medicinskih uslova (bira ih iz menija), o lekovima koje pacijent koristi (izbor iz menija), alergijama (slobodan tekst) i o kućnom životu (popuna formulara).

**SCENARIO AKO NEŠTO NE IDE PO PLANU:** Pacijentova elektronski zdravstveni karton ne postoji ili nije nađen. Sestra onda otvara novi zdravstveni karton i unosi neophodne informacije. Pacijentovo stanje i lekove koje uzima nisu uneti u zdravstveni karton. Sestra onda u formi slobodnog teksta unosi u rubriku „ostalo“ opis stanja pacijenta i lekove koje uzima. Pacijent ne može ili ne želi da pruži informacije o svojoj zdravstvenom istorijatu. U obliku slobodnog teksta, sestra onda unosi u sistem pacijentovu nesposobnost ili odbijanje da pruži tražene informacije. Sistem onda štampa standardni formular o isključenju u kome se nalazi stav da nedostatak informacija može da dovede do ograničene terapije ili do njenog pomeranja. Taj formular treba da potpiše pacijent.

**DRUGE AKTIVNOSTI:** Zdravstveni kartom se može koristiti, ali ne i menjati od drugog osoblja za vreme unosa informacija u njega.

**STANJE SISTEMA PO ZAVRŠETKU:** Korisnik je uključen u sistem. Pacijentov zdravstveni karton, zajedno sa njegovom zdravstvenom istorijom, se unosi u bazu podataka, dodatni slog je dodat i sistemsku log datoteku koja pokazuje početak i vreme završetka rada sa pacijentom, kao i zapis o imenu sestre koja je to ubacila u sistem

## ZADATAK 7

### *Proverite stečena znanja definisanja scenarija*

Za svaki slučaj korišćenja, definišite normalni (primarni) scenario i više (sekundarnih) scenarija koji opisuju situacije do kojih može da se dođe nekim greškama.

## ✓ 5.3 Slučajevi korišćenja sistema (Use Cases)

### SLUČAJEV KORIŠĆENJA

*Slučaj korišćenja utvrđuje učesnike (koji se u terminologiji UML nazivaju akterima) koji učestvuju u interakciji sa softverskim sistemom i ima svoj naziv*

Slučajevi korišćenja sistema, su detaljno opisani u prethodnoj lekciji koja obrađuje UML. Ovde ćemo samo ponoviti najvažnije stvari, a onda primeniti slučajeve korišćenja sistema u jednom primeru,

**Slučaj korišćenja** utvrđuje učesnike (koji se u terminologiji UML nazivaju akterima) koji učestvuju u interakciji sa softverskim sistemom i ima svoj naziv. Slučajevi korišćenja su postali osnovna osobina UML označavanja za opisivanje objektno-orientisanih modela sistema. **Skup slučajeva korišćenja treba da opiše sve moguće interakcije sa sistemom.** Akteri i slučajevi korišćenja u kojima učestvuju su povezani linijom sa strelicom na kraju.

Za svaki slučaj korišćenja neophodno je **definisati osnovni (primarni) scenario** i **pomoćne (sekundarne ) scenarije** koji u tekstualnoj formi opisuju interakciju aktera i sistema. **Osnovni scenario** opisuje osnovnu funkcionalnost sistema koju slučaj korišćenja opisuje.

**Pomoćni (sekundarni) scenari** definišu sve poremećaje koji se mogu desiti u realizaciji osnovnog scenarija interakcije. Sve te sekundarne scenarije treba definisati da bi se kasnije predvidelo kako da sistem razreši takve situacije.

**Dijagram slučajeva korišćenja** (UseCase) prikazuje spoljne učesnike, odnosno korisnike sistema i njihove veze sa korisničkim funkcijama koje sistem omogućava. Model slučajeva korišćenja obično se sastoji od više dijagrama slučajeva korišćenja.

Neke od najbitnijih stvari koje opisuju dijagram slučajeva korišćenja su:

1. Prikazuju ponašanja ili funkcionalnost sistema (grubi prikaz šta sistem ili podsistem radi, a ne kako sistem radi)
2. Kreiraju se u ranoj fazi da bi se razumeli zahtevi, a ne kako će se oni realizovati
3. Slučajevi korišćenja su načini prikupljanja funkcionalnih zahteva sistema, opisuju interakcije korisnika i sistema

Osnovne komponente dijagrama slučajeva korišćenja su:

- korisničke funkcije,
- učesnik (korisnik ili akter) i
- sistem koji se modelira, kao i različite veze: asocijacija, generalizacija i zavisnost između elemenata.

## SADRŽAJ DIJAGRAMA KORIŠĆENJA

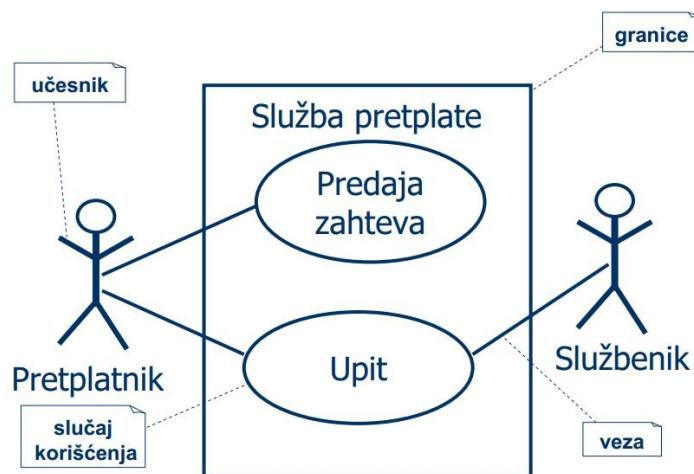
**Korisničke funkcije, kteri, relacije, komentari, paketi, i granice upotrebe korišćenja**

Dijagrami slučajeva korišćenja najčešće sadrže:

- Korisničke funkcije (Use case)
- Aktere,tj. uloge, učesnike (Actor)
- Relacije generalizacije , asocijacije i zavisnosti
- Komentare (Note)
- Pakete (Package)
- Granice Sistema

## Korisničke funkcije

Korisnička funkcija predstavlja funkcionalan zahtev posmatranog sistema. Korisnička funkcija izvršava određenu, sagledivu količinu posla. Iz perspektive datog izvođača, korisnička funkcija radi nešto što je izvođaču korisno. **Korisnička funkcija opisuje šta sistem radi ali ne definiše kako to radi.** Korisnička funkcija se u UML-u predstavlja kao oblačić (elipsa) sa svojim imenom koje je jedinstveno. Korisnička funkcija može da ima varijatete. Jedna korisnička funkcija može biti specijalizovana verzija neke druge korisničke funkcije (ostvaruje se relacijom generalizacije), može biti sadržana u nekim drugim korisničkim funkcijama (ostvaruje se relacijom obuhvatanja), ili može proširivati ponašanje nekih drugih korisničkih funkcija (ostvaruju se relacijom proširivanja). Korisnička funkcija predstavlja određenu funkcionalnost sistema iz perspektive korisnika tog sistema. Korisnička funkcija mora imati ime, koje obično predstavlja glagol.



Slika 5.2.1 Generalni primer dijagrama slučajeva korišćenja sa svim komponentama

Akter (izvođač) predstavlja ulogu koju korisnici korisničkih funkcija izvode kada su u interakciji sa tim korisničkim funkcijama. Uobičajeno je da izvođač predstavlja ulogu koju čovek, hardverski uređaj ili čak neki drugi sistem igra sa posmatranim sistemom. Uloga se u UML-u predstavlja kao ljudska figurica. Izvođači mogu biti povezani sa korisničkom funkcijom samo pomoću asocijacije

Korisničke funkcije i akteri mogu se organizovati definišući relacije generalizacije, zavisnosti (obuhvatanja/uključenja – include i proširivanje - extend) i asocijacije između njih.

## NAČIN CRTANJA VEZA U DIJAGRAMIMA SLUČAJEVA KORIŠĆENJA

### *Relacije zavisnosti (include, extend) na DSK i njihova primena*

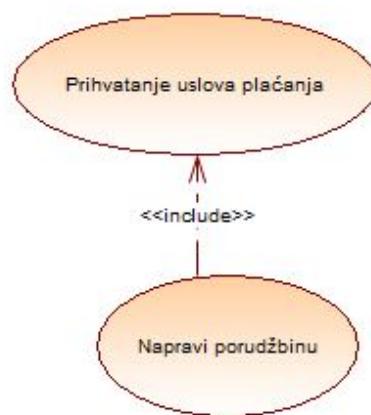
**Veza generalizacije** se na dijagramu slučajeva korišćenja koristi da predstavi nasleđivanje učesnika i nasleđivanje slučajeva korišćenja. Nema posebnih osobina.

**Veza zavisnosti** je uopštена UML veza kojom se mogu povezivati svi objekti bez ograničenja. Može i ne mora da ima definisan stereotip koji označava oblik zavisnosti.

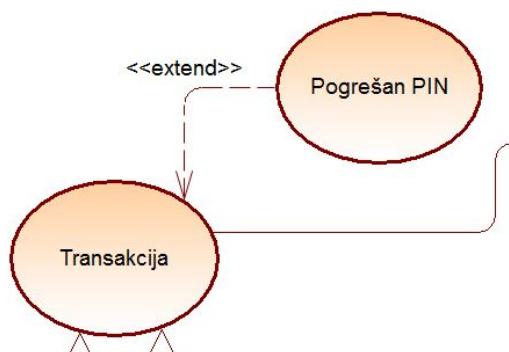
Na dijagramu slučajeva korišćenja koristi se za međusobno povezivanje slučajeva korišćenja za predstavljanje veze uključivanja (*include*) i veze proširivanja (*extend*).

Relacija uključivanja/obuhvatanja (*include*) može se primeniti samo između korisničkih funkcija. Relacija uključivanja (*include*) između korisničkih funkcija znači da osnovna korisnička funkcija eksplicitno obuhvata ponašanje druge korisničke funkcije. Obuhvaćena korisnička funkcija nikad ne može da stoji sama za sebe, već se javlja kao deo neće korisničke funkcije koja je obuhvata. Relaciju uključivanja treba koristiti da bi se izbeglo da se isti tok događaja opisuje više puta,

Relacija proširivanja (*extend*) znači da osnovna korisnička funkcija indirektno ugrađuje ponašanje druge korisničke funkcije na mestu koje je indirektno definisano proširujućom korisničkom funkcijom. Osnovna korisnička funkcija može stajati sama, ali pod određenim okolnostima, njeno ponašanje može biti prošireno ponašanjem druge korisničke funkcije. Osnovna korisnička funkcija može biti proširena samo na određenim mestima koja se nazivaju tačke proširenja. Proširivanje se može shvatiti kao da proširujuća korisnička funkcija ubacuje ponašanje u osnovnu korisničku funkciju ukoliko su zadovoljeni određeni uslovi. Relacija proširivanja koristi se za modelovanje dela korisničke funkcije koji korisnik može videti kao opcionalno ponašanje. Na taj način razdvaja se opcionalno od obaveznog ponašanja.



Slika 5.2.2 Primer relacije include



Slika 5.2.3 Primer relacije extend

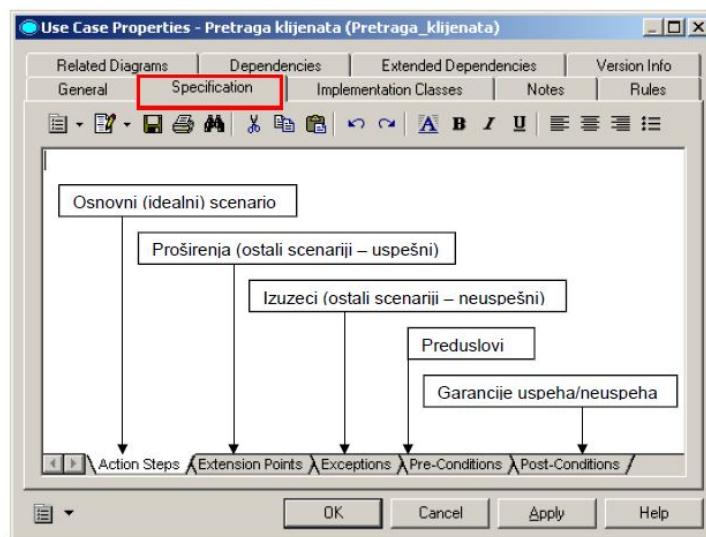
## SPECIFIKACIJA SLUČAJA KORIŠĆENJA

### *Specificiranje slučaja korišćenja korišćenjem Power Designer razvojnog alata*

Korisnička funkcija opisuje šta sistem radi, ali ne definiše kako to radi. Moguće je specifikovati ponašanje korisničke funkcije tekstualno opisujući tok događaja, dovoljno jasno da to lako razume i neko ko je neupućen. Kada se piše tok događaja, potrebno je naznačiti kada i kako korisnička funkcija započinje i završava, kada korisnička funkcija stupa u interakciju sa izvođačima, koji se objekti razmenjuju, potrebno je opisati osnovni i alternativne tokove ponašanja.

Specifikaciju korisničke funkcije moguće je odraditi u kartici „Specification“ na korisničkoj funkciji.

1. **Action steps** - opisuje se osnovni (uspešni) tok događaja. Pod osnovnim tokom događaja navodi se kada korisnička funkcija počinje svoje izvršavanje i ko inicira njeno izvršavanje. Ukoliko korisnička funkcija uključuje neke druge funkcije (postoji include zavisnost), navodi se kada se uključena korisnička funkcija izvršava.
2. **Extension Points** - alternativni tok događaja, ukoliko korisnička funkcija ima proširenja ona se ovde navode, uz potrebne uslove izvršavanja.
3. **Exceptions** - neuspešni tok događaja
4. **Pre - Conditions** - navođenje potrebnih preduslova da bi se korisnička funkcija uopšte izvršila
5. **Post - Conditions** - rezultati uspešnog i/ili neuspešnog izvršavanja korisničke funkcije



Slika 5.2.4 Kartica za opis izvršavanja slučajeva korišćenja

Opis osnovnog toka događaja je ustvari **osnovni (primarni) scenario** koji može da obuhvata i alternativne tokove događaja a u cilju opisa korisničke funkcije. Opisi izuzetaka

ustvari predstavljaju **pomoćne, tj. sekundarne scenarije**, koji mogu da ometu izvršenje korisničke funkcije.

## MODELOVANJE SLUČAJEVA KORIŠĆENJA SISTEMA

*Na šta treba обратити пажњу приликом modelovanja ponašanja posmatranog sistema*

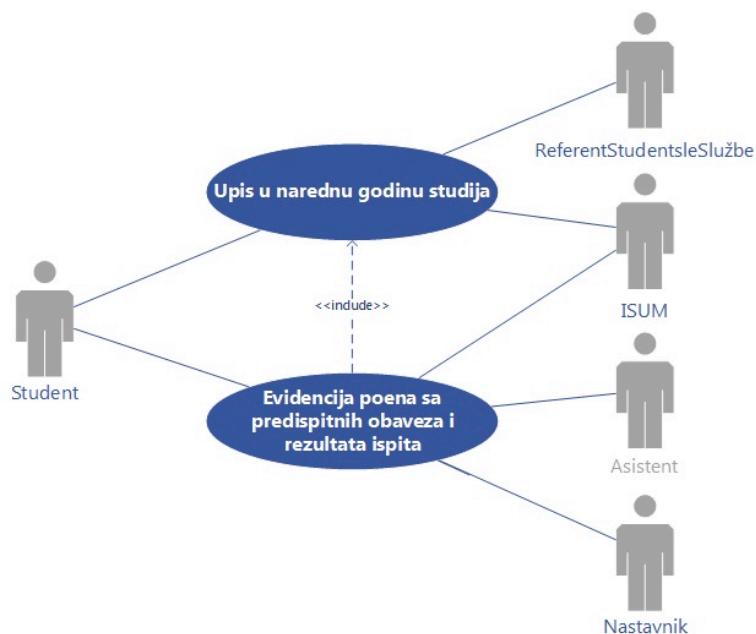
1. **Identifikujte aktere** (izvođače) koji su u interakciji sa posmatrаниm sistemom. Kandidati za uloge su grupe koje zahtevaju određeno ponašanje radi izvođenja svojih zadataka ili koji su neophodni direktno ili indirektno radi izvršavanja funkcija tog sistema
2. Za svakog izvođača **razmotriti ponašanje** koje on očekuje ili zahteva da sistem obezbedi i ta ponašanja potrebno je prikazati **kao korisničke funkcije**
3. **Organizujte aktere** identificujući opšte i specijalne uloge (primena generalizacije kod uloga)
4. Za svakog aktera ponaosob **odrediti načine na koje on stupa u interakciju** s sistemom koje menjaju stanje sistema ili njegove okoline ili koje obuhvataju slanje odgovora na neki događaj.
5. **Organizujte korisničke funkcije** primenjujući relacije proširivanja i obuhvatanja radi razlaganja na zajedničke delove ponašanja i razlikovanje alternativnih ponađanja ponašanja.
6. Za svaku korisničku funkciju, **odredite izuzetke** koji mogu da ometu njenu realizaciju,
7. Sve korisničke funkcije, aktere, relacije generalizacija, asocijacija i zavisnosti treba da imaju **imena koja govore o njihovoj nameni**
8. **Rasporediti elemente** tako da se broj linija koje se seku svedu na najmanju moguću meru
9. Ukoliko nije moguće prostorno organizovati elemente dijagrama tako da ponašanje i uloge koje su semantički bliske budu i fizički blisko raspoređeni, potrebno je **uvesti pakete u dijagram**

Obično se u praksi, daje neglasak na utvrđivanje korisničkih funkcija (šta sistem treba da radi). Međutim, mala sa pažnja poklanja na opis izuzetaka, tj. na sve ono što može da se javi u radu sistema, a što može da poremeti njegov rad. Za svaku korisničku funkciju (osnovni scenario) potrebno je da definišate sve moguće izuzetke (pomoćne scenarije) i za svaki izuzetak, treba da opište tok događaja koji dovodi do premećaja u radu sistema. Opisi ovih sekundarnih scenarija daju zadatke projektantima i programerima da u kodu softvera predvide te izuzezke i speče da se sistem blokira i da zaustavi izvršenje osnovne korisničke funkcije. Profesionalni softver treba da obuhvata, na ovaj način, sve moguće izuzetke, za svaku korisničku funkciju (osnovni scenario). Na taj način se razvija **robustan i pouzdan softverski sistem**, koga je "nemoguće blokirati i zbuniti", jer na svaki izuzetak, tj. izuzetnu situaciju koja ometa izvršenje neke korisničke i ima spremljeno rešenje. Programeri ne mogu da znaju šta u nekom radnom okruženju može da omete izvršenje softvera. To moraju analitičari da im saopštite definisanjem izuzetaka koji sa mogu javiti u radnom okruženju softvera.

## PRIMER SK: UPIS STUDENATA U SLEDEĆU GODINU STUDIJA

*Primer specifikacije slučaja korišćenja Upis u narednu godinu studija.*

Da bi ilustravili primenu slučajeva korišćenja za specifikaciju zahteva, pokazaćemo prime SK "Upis naredne godine studija". Na slici 5 prikazan je dijagram slučajeva korišćenja koji ima dva slučaja korišćenja. SK "Upis naredne godine studija" sadrži osnovnu korisničku funkciju, a SK "Evidencija poena sa predispitnih obaveza i ocena sa ispita" mora da bude uključena u osnovnu korisničku funkciju, tj. SK "Upisa u narednu godinu studija", jer bez evidencije poena sa predispitnih obaveza i rezultatata svih ispita, ona se ne može realizovati. Tabela 4 sadrži specifikaciju SK "Upis u narednu godinu studija".



Slika 5.2.5 Dijagram slučajeva korišćenja sanalise studija sluajeva.

Element slučaja korišćenja	Upisana vrednost ili opis
Naziv slučaja korišćenja	Upis u narednu godinu studija
Funkcija	Provera da li je student ostvario uslove za upis u narednu godinu
Akteri	Student, ISUM i Referent Studentske Službe
Okidlač	Završetak oktobarskog roka
Preduslovi	Unete ocene predispitnih obaveza i ispitnih ocena svih predmeta
Posledice	Upis u narednu godinu ili ponavljanje godine
Osnovni scenario	<ol style="list-style-type: none"> <li>Studentska služba, primenom ISUM-a, utvrđuje za svakog studenta da li je položio sve ispite iz prethodnih godina i sve ispite predmeta iz tekuće školske godine i sabira stečeni broj ESPB.</li> <li>Ako je student položio sve ispite predmeta iz prethodnih godina, i ako je stekao najmanje 37 ESPB na osnovu položenih ispti iz tekuće godine, onda ga <b>upisuju u sledeću školsku godinu</b></li> </ol>
Alternativni tokovi	<ol style="list-style-type: none"> <li>Ako student nema sve položene ispite iz prethodnih godina, ne može da upiše narednu godinu – <b>ponavlja godinu</b></li> <li>Ako je student položio sve ispite iz prethodnih godina, a nije ostvario 37 ESPB na predmetima tekuće godine, onda <b>ponavlja godinu</b></li> <li>Ako je student <b>upisao narednu godinu sa manje od 60 ESPB</b> na predmetima tekuće godine, on prenosu u narednu godinu ponene na predispitnim obavezama i nastavlja da ih radi i priprema ispit iz nepoloženih predmeta.</li> <li>Ako student <b>nije upisao narednu godinu</b>, njemu se poništavaju svi poeni na predispitnim obavezama predmeta koje nije položio, i on mora da ponovo prati nastavu na ovim predmetima i da ponovo radi sve predispitne obaveze (novi domaći zadaci i projekat), kao i ostali studenti tekuće školske godine</li> </ol>
Izuzeći – pomoćni scenariji	<ol style="list-style-type: none"> <li><b>Student dolazi iz druge VŠU sa nezavršenim studijama:</b> Analiziraju se svi predmeti koje je položio u prethodnoj VŠU, i određuje se % prihvatanja programa (od 0 do 100%). To radi posebna komisija za studijski program. Zavisno od broja priznatih ESPB, upisuju: a) drugu godinu, ako ima 37 do 96 ESPB, b) treću godinu, ako ima od 197 do 156 ESPB, treću godinu ako ima od 157 do 116 ESPB, i četvrtu godinu, ako ima najmanje 117 ESPB ili više.</li> <li><b>Student dolazi iz druge VŠU sa završenim OAS ili OSS sa 180 ESPB:</b> Komisija određuje predmete predmete 4. godine, s ciljem da ima najvažnije predmete, ako ih nije imao u prethodnoj VŠU. Ako je potrebno, mogu mu se odrediti i do 3 pripremana predmeta.</li> <li><b>Student UM nije ispunio uslove za upis u narednu godinu, ali navodi posebne razloge zbog kojih to nije ostvario:</b> Komisija predlaže a dekan odlučuje da li da dozvoli studentu upis u narednu godinu studija.</li> </ol>

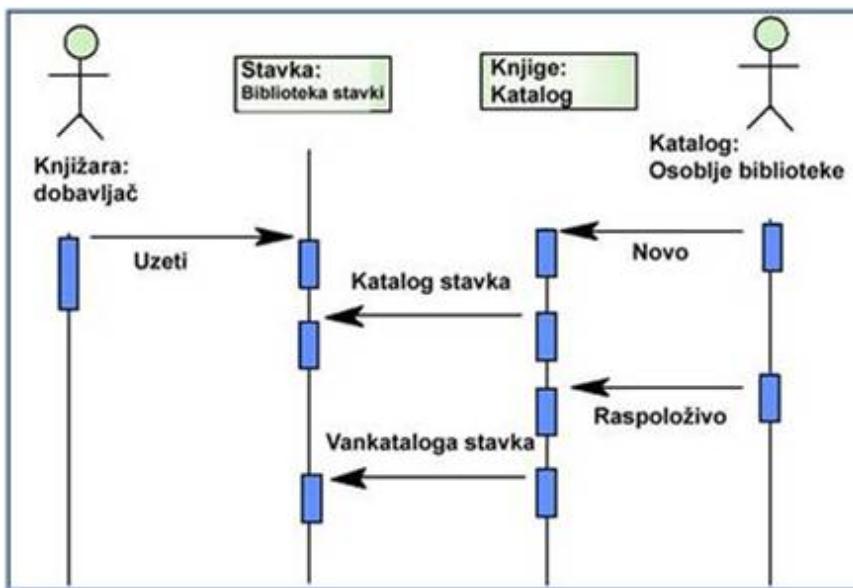
## PRIMENA UML SEKVENCIJALNIH DIJAGRAMA U SLUČAJEVIMA KORIŠĆENJA

*Sekvencijalni dijagram prikazuje redosled interakcija aktera sa osnovnim objektima sistema, ali prikazuje i interakciju između ovih objekata.*

Pored tekstualnog opisa scenarija, koji opisuju svaki slučaj korišćenja, moguće je da njihov i grafički opis, koristiti tzv. sekvencijalne dijagrame UML sistema. Unutar UML-a sekvencijalni dijagrami mogu biti korišćeni da dodaju informacije u slučaj korišćenja, objekte unutar sistema sa kojima su akteri u interakciji. Sekvencijalni dijagrami koji se koriste radi opisa scenarija su prilagođeni svrsi, te ne sadrže detalje koji koriste sekvencijalni dijagrami kada se koriste za projektovanje klase. Sekvencijalni dijagrama koji se koriste za opis scenarija, koji se odnose na slučajeve korišćenja, sadrže apstraktne definisane entitete ili objekte koji realizuju definisan scenario. Linije sa strelicama prikazuju operacije koje se realizuju u vezi sa prikazanim objektima.

Na slici 6 je prikazana interakcija obuhvaćena uzimanjem i katalogiziranjem knjiga za biblioteku. Slika ilustruje da postoje dva objekta klase Biblioteka stavki i Katalog obuhvaćeni slučajem korišćenja Upravljanje katalogom knjiga. Sekvenca akcija je od vrha naniže i oznake na strelicama između aktera i objekata ukazuju na ime operacije. Zbog toga, kada je knjiga

kupljena, operacija Novo je izvedena na Katalogu i operacija Uzeti na Stavki. Jednom kada je knjiga dostupna, Katalog stavka je izvedena na Stavki.



Slika 5.2.6 Primene UML sekvencijalnog dijagrama za opis scenarija slučaja korišćenja

## PRIMER ATM

### *Primer izrade dijagrama slučajeva korišćenja za ATM sistem*

Primer DSK dat je na slici 15 . Neki od narednih koraka opisani su na slici ispod.

#### Action steps:

- Korisnik inicira transakciju ubacujući karticu u bankomat
- Korisnik bira jedan od ponuđenih jezika na kome će se izvršiti transakcija
- Korisnik unosi svoj PIN code
- Sistem vrši proveru validnosti unetog pin code-a
- Korisnik bira jednu od ponuđenih transakcija

#### Extension Points:

- U slučaju da je korisnik uneo neodgovarajući PIN code prelazi se na slučaj korišćenja „Invalid PIN“

#### Exceptions:

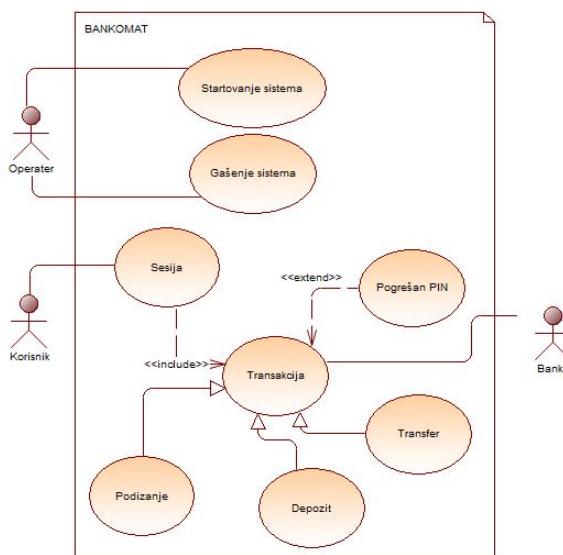
- Korisnik je tri puta uneo neispravan PIN code, kartica je oduzeta a transakcija otkazana.

#### Pre-Conditions:

- Bankomat je uključen
- Korisnik je uneo ispravan PIN code

#### Post - Conditions:

- Korisnik je uspešno izvršio željenu transakciju



Slika 5.2.7 Primer DSK za ATM sistem

## PRIMER ATM - NASTAVAK

### *Primer izrade dijagrama slučajeva korišćenja za ATM sistem II*

Specifikacija korisničke funkcije „Podizanje“

- **Action Steps**

1. Korisnik bira jedan od ponuđenih iznosa koji želi da podigne
2. Korisnik potvrđuje izabrani iznos
3. Sistem vrši proveru da li postoji dovoljan iznos na računu korisnika
4. Sistem vrši proveru da li u bankomatu postoje dovoljna sredstva
5. Korisnik bira da li želi štampanje izveštaja o izvršenoj transakciji

#### **Extension Points**

- U slučaju da korisnik nije izabrao nijedan od ponuđenih iznosa, nudi mu se mogućnost da sam unese željeni iznos
- U slučaju da je korisnik izabrao opciju da se štampa izveštaj o izvršenoj transakciji, nakon izvršene transakcije štampa se izveštaj

#### **Exceptions**

- Korisnik nema dovoljan iznos sredstava na svom računu
- U bankomatu ne postoji dovoljan iznos sredstava za isplatu

#### **Pre-Conditions**

- Bankomat je uključen

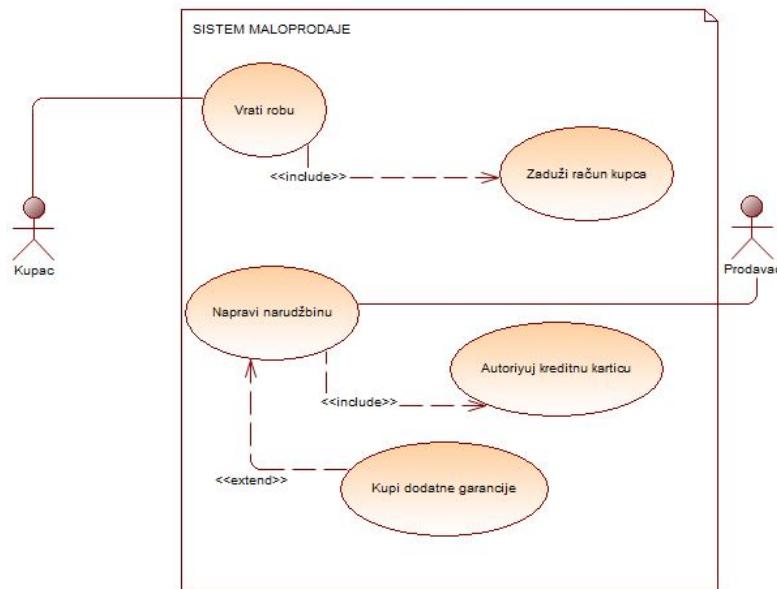
- Korisnik je uneo ispravan PIN code

### Post-Conditions

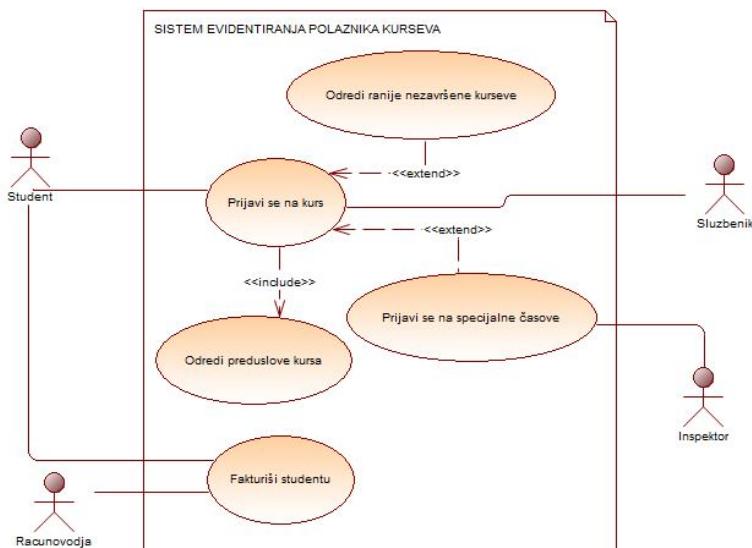
- Korisnik je uspešno podigao novac i dobio izveštaj o izvršenoj transakciji

## PRIMERI DSK ZA SISTEM EVIDENCIJE POLAZNIKA KURSEVA I SISTEMA MALOPRODAJE

*Primeri DSK za sistem evidencije polaznika kurseva kao i sistem maloprodaje*



Slika 5.2.8 Dijagram slučajeva korišćenja za sistem maloprodaje

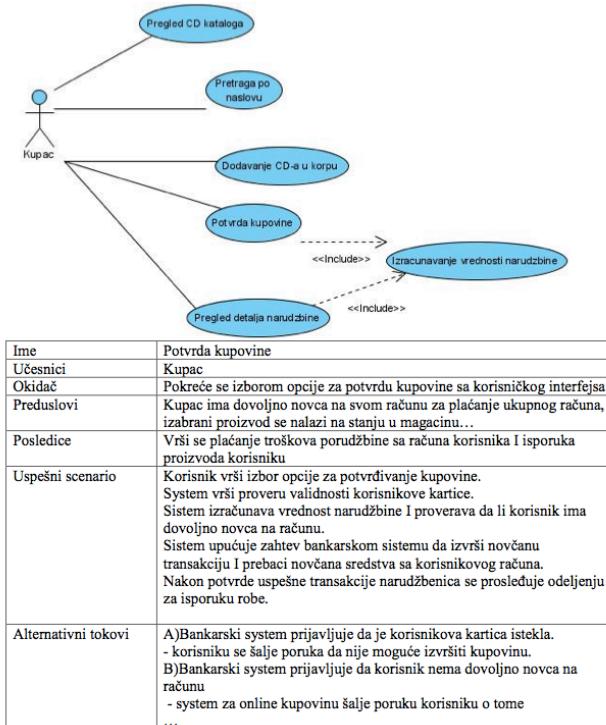


Slika 5.2.9 Dijagram Slučajeva korišćenja za sistem evidentiranja polaznika kurseva

## PRIMERI SISTEMA CD KATALOGA I LANSIRANJA RAKETA

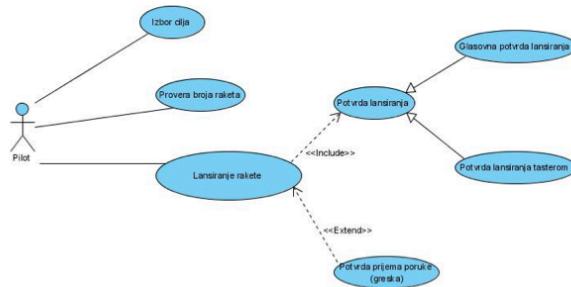
*Primeri DSK za dva različita scenarija uključujući i tabelarne opise dijagrama*

Kreirati dijagram slučajeva korišćenja za on-line CD katalog. Sistem ima jedinstvenog učesnika: on-line kupca. Kupac može da pregleda katalog. Pretražuje po naslovu, dodaje CD u narudžbinu, pregleda detaljnije narudžbine i potvrdi kupovinu. Dati specifikaciju najbitnijeg slučaja upotrebe sa opisima.



Slika 5.2.10 Primer rešenja postavljenog primera 1

Realizuje se softverski sistem za kontrolu lansiranja raketa. Pilot može izvršiti izbor cilja, lansiranje raketa, proveru broja raketa. Svako lansiranje rakete potrebno je da potvrdi pri čemu se potvrda može obaviti odgovarajućom glasnom komandom ili pritiskom odgovarajućeg dugmeta. Ukoliko se u toku lansiranja desi kvar na sistemu za lansiranje pilot će biti obavešten o ovom događaju i potrebno je da potvrdi prijem poruke. Dati specifikaciju najbitnijeg slučaja korišćenja sa opisom.



Ime	Lansiranje rakete
Učesnici	Pilot
Okidač	Pritisak na dugme za lansiranje
Preduslovi	Izvrsena potvrda lansiranja rakete Na lanseru rakete je postavljena raketa, Avion se nalazi na visini od preko 1000 m, Izabrani cilj se nalazi u dometu rakete,
Posledice	Ispaljuje se raketa sa lansera u pravcu izabranog cilja.
Uspešni scenario	Pilot vrši izbor tipa rakete koju zeli da ispalji na izabrani cilj. Pilot pritiska dugme za ispaljivanje rakete. Sistem trazi potvrdu lansiranja. Pilot potvrđuje (glasom ili tasterom) lansiranje rakete. Raketa se ispaljuje
Alternativni tokovi	A) Sistem prijavljuje da izabrana raketa ne postoji B) Sistem prijavljuje kvar na lanseru rakete C) Izabrani cilj se nalazi van dometa. a. Pilot dobija poruku da je cilj izasao iz dometa, b. Pilot potvrđuje prijem poruke I obustavlja lansiranje.

Slika 5.2.11 Primer rešenja postavljenog problema 2

## ZADATAK 8

### Specifikacija slučaja korišćenja

- U primeru slučaja korišćenja "Upis studenta u sledeću školsku godinu" dat je opis jednog od dva slučaja korišćenja koji primer koristi. Dajte specifikaciju drugog slučaja korišćenja "Evidencija poena sa predispitnih obaveza i rezultata ispita". Koristite isti uzorak tabele za specifikaciju koja je korišćena u tom primeru
- Savetuj kako bi inženjer odgovoran za utvrđivanje specifikacije zahteva mogao da poveže funkcionalne i nefunkcionalne zahteve

## ✓ 5.4 Etnografija

### ŠTA JE ETNOGRAFIJA?

*Etnografija je tehnika osmatranja koja se koristi radi razumevanja operacionih procesa. Analitičar svakodnevno osmatra i beleži stvarne zadatke koji učesnici rade.*

Softverski sistemi ne rade u izolaciji. Oni se koriste u određenom socijalnom i organizacijskom kontekstu i oni moraju da budu izvedeni iz takvog konteksta ili njime ograničeni. Da bi softverski sistem uspešno radio, on treba da zadovolji te društvene i organizacijske zahteve. Ima slučajeva da se neki softverski sistem ne koristi, iako je razvijen u skladu sa zahtevima. U tim slučajevima, nisu uzeti u obzir društveni ili organizacijski zahtevi.

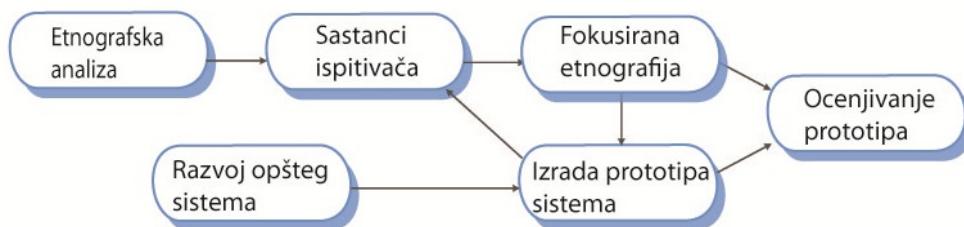
Etnografija je tehnika osmatranja koja se koristi radi razumevanja operacionih procesa. Analitičar svakodnevno osmatra i beleži stvarne zadatke koji učesnici rade. Na taj način se mogu otkriti i implicitni zahtevi koji odražavaju stvarni način rada, a ne formalni proces koja je definiše organizacija.

Neki ljudi rade posao, ali ne umiju da objasne kada i kako rade. Naročito, ne umiju da taj rad povežu sa društvenim i organizacijskim zahtevima. Osmatranje njihovog rada pomaže u otkrivanju ovih zahteva.

Etnografija je naročito korisna u otkrivanju dve vrste zahteva:

1. Zahtevi koji proističu iz načina kako ljudi stvarno rade (često i kršeći neke propise).
2. Zahtevi koji proističu iz kooperacije i aktivnosti drugih ljudi.
3. Etnografija se može kombinovati sa izradom prototipova (slika), jer ukazuje na vrstu prototipova koji su potrebni, te time i smanjuje njihov neophodan broj.

Etnografske studije mogu otkriti detalje kritičkih procesa koji često nedostaju primenom drugih tehnika nalaženja zahteva. Etnografija, se mora kombinovati sa drugim tehnikama otkrivanja zahteva, jer ona nije u stanju da otkrije organizacijske i domenske zahteve, a ponekad ne može da ukaže na neophodne nove zahteve.



Slika 5.3.1 Etnografija i izrada prototipova u izradi zahteva

## ✓ Poglavlje 6

### Validacija zahteva

#### ŠTA JE VALIDACIJA ZAHTEVA?

*Validacija zahteva je provera da li zahtevi stvarno definišu sistem koji kupac želi.*

Validacija zahteva provera da li zahtevi stvarno definišu sistem koji kupac želi. Validacija zahteva je važna zato što greške u dokumentu zahteva mogu da vode velikim troškovima prepravke, kada su otkriveni tokom razvoja ili nakon što je sistem pušten u rad. Trošak realizovanja promene sistema koji je uzrokovani problemom zahteva je mnogo veći od popravke projekta ili grešaka kodiranja. Razlog za ovo je da promena zahteva obično znači da projektovanje i implementacija sistema moraju da budu promenjeni i da sistem mora biti testiran ponovo.

Tokom procesa validacije zahteva, različiti tipovi provere treba da budu izvedeni na zahtevima u dokumentu zahteva. Ove provere uključuju:

1. **Provera validacije** – Korisnik smatra da je neophodno da sistem ima neke funkcije. Međutim, buduća razmišljanja i analize mogu identifikovati dodatne ili različite funkcije koje su potrebne. Sistem ima različite korisnike sa različitim potrebama i neki skup zahteva je neizbežan kompromis u toku komunikacije sa korisnikom.
2. **Provera konzistencije** – Zahtevi u dokumentu ne treba da se sukobljavaju. To je zato što ne treba da bude kontradiktornih ograničenja ili različitih opisa iste funkcije sistema.
3. **Provera kompletnosti** – Dokument zahteva treba da uključi zahteve koji definišu sve funkcije i ograničenja planirana od strane korisnika.
4. **Provera realizma** – Korišćenjem znanja postojećih tehnologija, zahtevi treba da budu provereni da bi se obezbedilo da li mogu stvarno da budu implementirani. Ove provere takođe treba da uzmu u obzir raspoloživi budžet i vreme potrebno za razvoj sistema.
5. **Verifikacija** – Da bi smanjili potencijalne rasprave između korisnika-nručioca softvera i izvođača, zahtevi sistema treba uvek da budu pisani tako da mogu da budu provereni. Ovo znači da može biti projektovan skup provera koji može da demonstrira da li je isporučeni sistem zadovoljio zahteve.

## TEHNIKE VALIDACIJE

*Tehnike validacije su pregledi zahteva od strane recenzenta, korišćenje prototipa kao modela za proveru definisanih zahteva, i korišćenje testova povezanih sa svakim zahtevom.*

Postoji određen broj tehnika validacije zahteva koje mogu biti korišćene zajedno ili pojedinačno:

1. **Pregledi zahteva** – Zahtevi su sistematski analizirani od strane tima ocenjivača.
2. **Prototip** – U ovom pristupu validaciji, izvršni model sistema prikazan je krajnjim korisnicima i mušterijama. Oni mogu eksperimentisati sa ovim modelom da bi videli da li zadovoljava njihove realne potrebe.
3. **Stvaranje test slučaja** – Idealno, zahtevi treba da budu tako urađeni tako da mogu da budu testirani. Ako su testovi za zahteve smišljeni kao deo procesa validacije, ovo često otkriva probleme zahteva. Ako je test težak ili nemoguć za projektovanje, ovo obično znači da će biti teško implementirati zahteve i da treba da budu ponovo razmatrani.

Ne treba potceniti značaj validacije zahteva. Teško je pokazati da neki skup zahteva zadovoljava potrebe korisnika. Korisnici moraju da zamisle sliku sistema u radu i da vide kako će sistem odgovarati njihovom poslu. Za najveći broj korisnika sistema, pretežak je zadatak vršenje takve apstraktne analize. Zato se teško i otkrivaju greške u fazi procesa validacije zahteva. Zahtevi za promenu zahteva obično dolaze tek posle izdavanja dokumenta ta zahtevima.

## ZADATAK 9

### *Recenzija zahteva*

Ko treba da učestvuje u recenziji zahteva? Nacrtajte moguć model procesa recenzije zahteva.

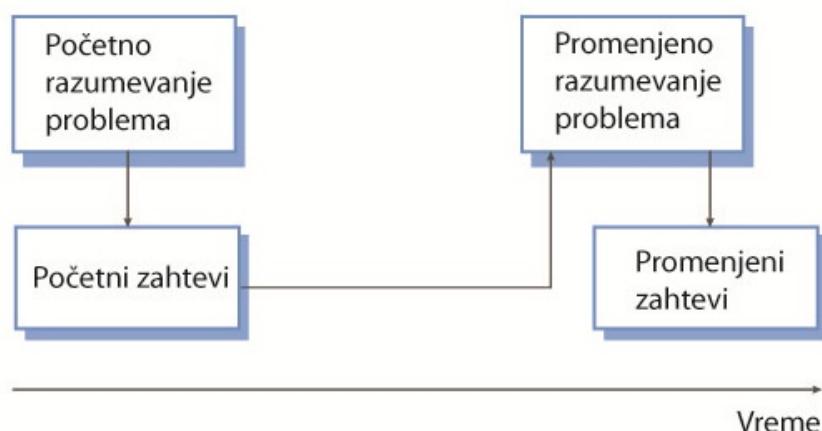
## ✓ Poglavlje 7

# Upravljanje zahtevima

## ŠTA JE UPRAVLJANJE ZAHTEVIMA?

*Upravljanje zahtevima je proces razumevanja i kontrole promene u zahtevima sistema*

Zahtevi kod velikih softverskih sistema se stalno menjaju. Jedan od razloga za ovo je u nemogućnost sagledavanja svih zahteva kod ovih sistema. Po pravilu, zahtevi za ove sisteme nisu potpuno definisani, te su nekompletni. Za vreme softverskog procesa (tj. procesa razvoja softvera), razumevanje problema kod aktera sistema se stalno menja (slika 1.). Zbog toga, i zahtevi sistema mora stalno da evoluiraju, da bi odražavali promene u pogledu na problem.



Slika 7.1 Evolucija zahteva

Kada se novi sistem pusti u upotrebu, neminovno se javi novi zahtevi. Teško je krajnji korisnici i akteri sistema mogu da sagledaju efekte novog

sistema na njihove poslovne procese i na način obavljanja posla. Posle sticanja prvih iskustava sa novim sistemom, krajnji korisnici sistema počinju da otkrivaju nove potrebe i prioritete. Za to postoji nekoliko razloga:

1. Poslovno i tehničko okruženje sistema se uvek menja posle instalacije sistema (novi hardver, novi interfejsi, novi poslovni prioriteti, novi propisi).
2. Ljudi koji naručuju ili biraju sistem i ljudi koji koriste sistem, najčešće nisu isti. Kupci sistema postavljaju zahteve zbog organizacijskih i finansijskih ograničenja. To može da bude u sukobu sa zahtevima korisnika sistema, te se pokaže potreba da se posle instalacije moraju da dodaju novi korisnički zahtevi.

3. Veliki sistemi obično imaju šaroliku zajednicu korisnika, sa različitim zahtevima i prioritetima, koji mogu biti i međusobno suprotstavljeni. Konačan sistem je onda obično neki kompromis između tih zahteva, a praksa pokaže da ipak mora pomoći nekim drugim korisnicima, tj. da treba udovoljiti njihovim zahtevima.

Upravljanje zahtevima je proces razumevanja i kontrole promene u zahtevima sistema.

## PLANIRANJE UPRAVLJANJA ZAHTEVIMA

*Planiranje je bitna faza procesa upravljanja zahtevima u kojoj se definišu detalji neophodni za donošenje određenih odluka.*

Neophodno je beležiti pojedinačne zahteve i održavati vezu između zavisnih zahteva, kako bi se realnije procenili efekti promena u zahtevima. Neophodni je za ovo da se definišu formalni procesi za upravljanje zahtevima. Formalan proces treba da počne što ranije, čim verzija dokumenta sa zahtevima bude u nacrtu,

Planiranje je bitna prva faza u procesu upravljanja zahtevima. Tada se uspostavlja nivo upravljačkih detalja koji je potreban. U ovoj fazi treba da se donešu sledeće odluke:

1. **Identifikacija zahteva.** Svaki zahtev mora da ima jedinstvenu identifikaciju tako da se može pomešati sa drugim zahtevima, kao i da se omogući ocena primene zahteva.
2. **Proces upravljanja promenama.** Ovo je skup aktivnosti koje ocenjuju efekat i troškove vezan za promene.
3. **Pristupi praćenju primene zahteva.** Oni definišu odnose između svakog zahteva i između zahteva i projektovanja sistema koji se treba pratiti i zapisivati. Ovi pristupi određuju i način održavanja ovih zapisa.
4. **Podrška alata.** Upravljanje zahtevima zahteva obradu velike količine informacija o zahtevima. Zato je neophodno korišćenje alata za upravljanje zahtevima.

Upravljanje zahtevima treba automatizovati i obezbediti softverske alate koji se moraju izabrati u fazi planiranja. Ta podrška je potrebna radi realizacije sledećih aktivnosti:

1. **Skladištenje zahteva.** Zahtevi treba da se održavaju na sigurnom, upravljivom skladištu podataka, tako da svako ko je u uključen u procese inženjeringu zahteva može da pristupi tom skladištu.
2. **Upravljanje promenama.** Proses upravljanje promenama (slika) je uprošćen ako se koristi odgovarajući alat.
3. **Upravljanje primenom zahteva.** Primenom odgovarajućeg alata otkrivaju se pojedini zahtevi, a i prati primena svih zahteva, kao i analiza njihovih veza.

Pri razvoju malih sistema, nije nužno koristiti alate za upravljanje zahtevima.



Slika 7.2 Planiranje u procesu upravljanja zahtevima

## ZADATAK 10

### *Hitna promena zahteva*

U sistem treba ugraditi mogućnost hitne promene zahteva. Softver se može promeniti i pre nego što su novi zahtevi verifikovani. Predloži model procesa za realizaciju ovih promena koji treba da obezbedi da dokument sa zahtevima ne bude neusaglašen sa implementacijom softverskog sistema

## ✓ Poglavlje 8

# Modelovanje zahteva

## UVOD U VEŽBE

*U ovim vežbama biće prikazan Power Designer model zahteva kao i UML UseCase model u svrhu prikupljanja i definisanja zahteva korisnika*

Najbitniji deo razvoja softverskog inženjerstva jeste ispuniti zahteve korisnika i udovoljiti njegovim željama i potrebama. Odатле je i nastala izreka "**Ukoliko korisnik nije zadovoljan, niko nije zadovoljan**". Svaki uspešan deo softvera nastao je kao korisnikova velika ideja. Vaš posao kao profesionalnog softverskog inženjera jeste da njegove ideje sprovedete u delo i jednostavno ih oživite. Uzimajući nejasnu ideju i sporvesti je u rad - zadovoljivši klijenta / nije tako lako. U ovim vežbama biće demonstriran način kreiranja zahteva korišćenjem Power Designer modela zahteva i njegovo povezivanje sa drugim delovima sistema.

Takođe biće objašnjen način modelovanja dijagrama slučajeva korišćenja (DSK) i njegova primena korišćenjem PowerDesigner alata.

Dodatni nastavni materijal koji mogu da budu od pomoći dati su na sistemu za e-učenje, a u okviru lekcije KI206-L04, kao što su:

1. Requirements Modeling sa PowerDesigner® 16.1 (originalno uputstvo za modeliranje zahteva sa Power Designer sistemom)
2. Use Case Description (uzorak formulara za definisanje slučaja korišćenja)
3. Više primera koji opisuju utvrđivanje i definisanje zahteva pojedinih softverskih sistema

## ✓ 8.1 Modeliranje zahteva sa PowerDesigner-om

## MODEL ZAHTEVA POWER DESIGNER ALATA

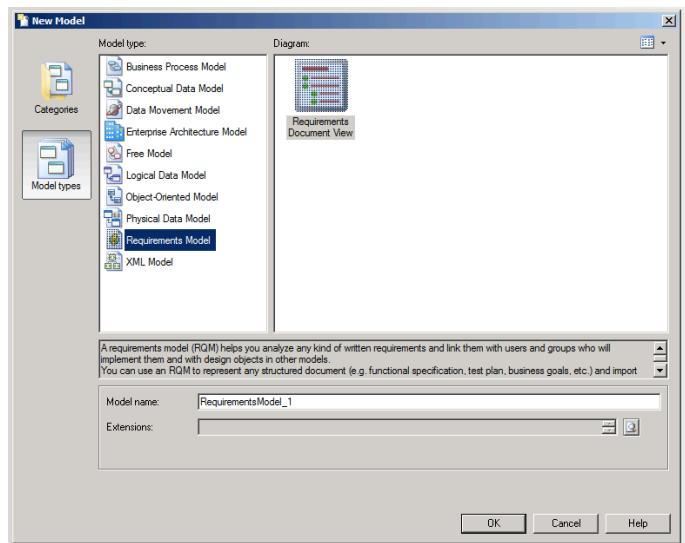
*Model zahteva (Requirements Model) se u PowerDesigner-u koristi za dokumentovanje zahteva u fazi analize zahteva softverskog sistema.*

Model zahteva se kreira odabirom stavke **File → New** iz glavnog menija ili New ikonice na standardnom toolbar-u. U dijalogu (slika 1) je potrebno izabrati Requirements Model sa leve strane, te zadati naziv sa desne strane (za First diagram odabratи Requirements Document View).

Model zahteva nema dijagrame, kao ostali PowerDesigner modeli, već se zahtevi definišu kroz tabelu (Requirements Document, slika 2). Ova tabela sadrži spisak zahteva sistema raspoređenih u hijerarhiju. Jedan model zahteva može da sadrži više ovakvih tabela (što može da olakša čitljivost ako postoji veliki broj zahteva), koje mogu da se rasporede i u pakete.

Sem ove tabele, model zahteva može da sadrži

- **matricu preslikavanja** (Traceability Matrix, v. odeljak Povezivanje zahteva sa elementima dizajna),
- **matricu za raspoređivanje** zahteva po korisnicima (User Allocation Matrix), kao i
- **rečnik pojmoveva** (v. odeljak Glossary).



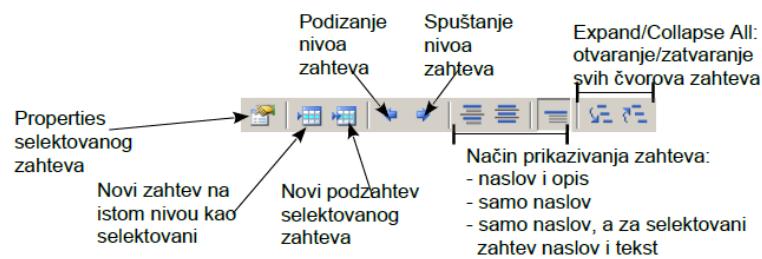
Slika 8.1.1 Kreiranje novog modela zahteva

#	Title (ID)	Full Description	Code	Creation Date	Priority
1.	1. Overview	Overview	Overview	Friay, October 25,	1
2.	2. Use Cases	Use_Cases	Use_Cases	Friay, October 25,	1
3.	2.1 Identified Users	Identified_Users	Identified_Users	Friay, October 25,	1
4.	2.1. Customer	Customer	Customer	Friay, October 25,	1
5.	2.1. Administrator	Administrator	Administrator	Friay, October 25,	1
6.	2.2 Identified Use Cases	Identified_Use_Cases	Identified_Use_Cases	Friay, October 25,	1
7.	2.2 Registration	Registration	Registration	Friay, October 25,	1
8.	2.2. Log On	Log_On	Log_On	Friay, October 25,	1
9.	2.2. Consultation	Consultation	Consultation	Friay, October 25,	1
10.	2.2. Administration	Administration	Administration	Friay, October 25,	1
11.	2.2 Adding new publication	Adding_new_publication	Adding_new_publication	Friay, October 25,	1
12.	2.2 Removing an existing publication	Removing_an_existing_publication	Removing_an_existing_publication	Friay, October 25,	1

Slika 8.1.2 Pogled na dokument sa zahtevima

## TRAKA SA ALATIMA ZA RAD SA ZAHTEVIMA - REQUIREMENTS TOOLBAR

*Traka sa alatima za rad sa zahtevima (requirements) nalazi se direktno iznad tabele sa zahtevima. Prikazana slika demonstrira glavne osobine trake sa alatima..*

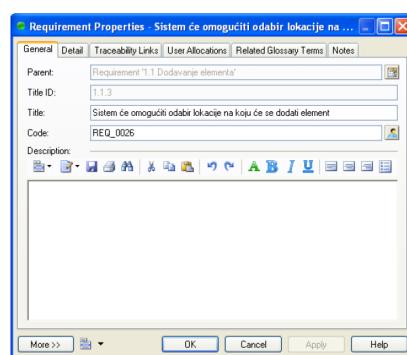


Slika 8.1.3 Traka alata u okviru dijagrama zahteva

## OSOBINE ZAHTEVA

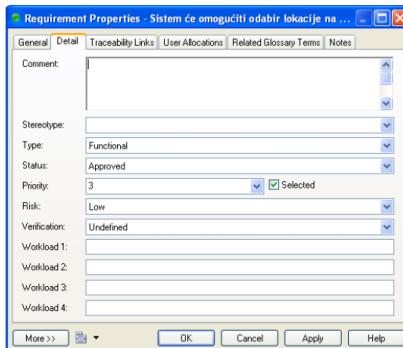
### *Definisanje zahteva u okviru modela zahteva*

- Deo osobina zahteva je prikazan u tabeli i može se tu direktno editovati. Prikazane kolone se podešavaju odabirom ikonice Customize Columns and Filter na Requirements toolbar-u.
- Kompletan pregled osobina zahteva se nalazi u njegovom Properties dijalogu, do kojeg se može doći odabirom Properties ikonice na Requirements toolbar-u ili dvoklikom na sivo polje u tabeli ispred zahteva.
- Na kartici **General** definišu se osnovne osobine zahteva kao što je prikazano na slici 4 .
- Na kartici **Related Glossary Terms** prikazani su pojmovi koji su važni za razumevanje datog zahteva (na primer, pominju se u njegovom opisu). Oni se mogu odabratи iz prethodno unetih pojmoveva (dugme Add Objects) ili se mogu kreirati novi pojmovi i automatski povezati sa datim zahtevom (dugme Create an Object).



Slika 8.1.4 Osobine zahteva - kartica General

Parent	Roditeljski zahtev
Title ID	Broj zahteva koji odražava njegovo mesto u hijerarhiji (automatski se generiše i ne može se menjati)
Title	Naziv zahteva
Code	Kód zahteva. Generiše se automatski, a može se izmeniti.
Description	Detaljan opis zahteva. Ovaj isti tekst se nalazi i na Notes kartici.



Slika 8.1.5 Osobine zahtev, kartica Details

Type	Tip zahteva: funkcional (funkcionalni), technical (nefunkcionalni), design (podskup nefunkcionalnih zahteva koji može da ima uticaja na njegov dizajn).
Status	Status validacije zahteva: draft, defined, verified, to be reviewed, approved.
Priority	Priortitet zahteva: brojna pozitivna vrednost sa max jednom decimalom. Preporuka je da se konsti tr do četiri nivoa prioriteta: 1, 2, 3, 4.

## REČNIK POJOMOVA

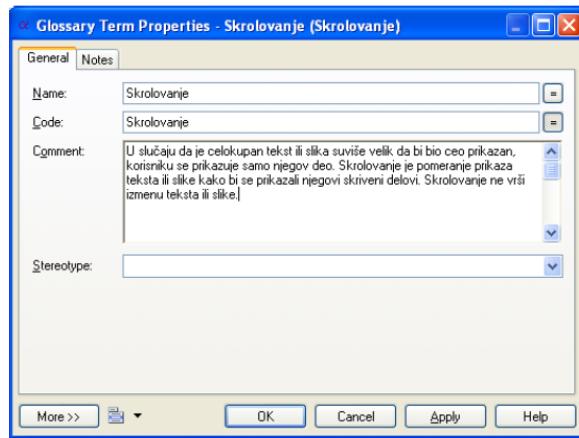
*Rečnik pojmove služi za precizno definisanje pojmove koji se koriste u opis zahteva. Power Designer nema poseban dijagram za reči pojmove, već se sve definiše u okviru istog modela*

Specifikacija zahteva uvek uključuje glossary (rečnik pojmove), koji služi za precizno definisanje pojmove koji se koriste u opisu zahteva, kako bi se izbegle nedoumice i nesporazumi. U Requirements modelu ne postoji poseban dijagram za definisanje ovih pojmove, već se oni mogu dodati na neki od sledećih načina:

- Model → Glossary Terms – prikazuje spisak svih pojmove, sa mogućnošću dodavanja novih
- desni klik na model u browser-u → New → Glossary Term.

Pojam se takođe može dodati u properties dijalogu svakog zahteva, na kartici Related Glossary Terms, kako je to opisano u prethodnom odeljku.

Za pojam je potrebno navesti naziv (Name) i objašnjenje (Comment).



Slika 8.1.6 Unos termina u Glossary

## GENERISANJE ELEMENATA DIZAJNA I ZAHTEVA

*Power designer omogućava kreiranje veze između modela zahteva i drugih modela kako bi se omogućilo njihovo jednostavno praćenje*

PowerDesigner omogućava da se zahtev eksportuje u neki drugi model, pri čemu se definiše tip elementa u koji zahtev treba da se pretvori na tom modelu, a kreirani element će imati isti naziv kao zahtev od koga potiče. Takođe se automatski kreira veza između tog elementa i zahteva iz kojeg je kreiran.

Operacija se pokreće iz modela zahteva odabriom opcije Requirements → Export Requirements as Design Objects. U prvom koraku se odabira zahtev koji se eksportuje, zatim se odabira model u koji se vrši eksport, te tip elementa.

Napomene:

- Ukoliko je za odabrani zahtev već vezan element odabranog tipa, operacija se neće moći izvršiti.
- Kreirani elementi nemaju svoj simbol, ali se mogu pronaći u browser-u, te prevući na odgovarajući dijagram.

## POVEZIVANJE ZAHTEVA SA ELEMENTIMA DIZAJNA

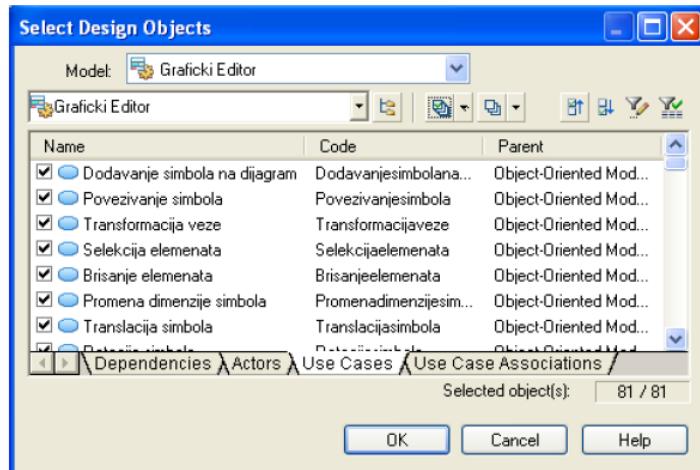
*Zahteve je moguće povezati sa elementima drugih dijagrama*

Zahteve je moguće povezati sa elementima drugih dijagrama (npr. elementima UML dijagrama: slučajevima korišćenja, klasama i slično), kako bi se omogućilo njihovo jednostavno praćenje tokom svih faza razvoja softvera. Takođe, zahtevi mogu biti međusobno povezani, tj. jedan zahtev može da se referiše na drugi zahtev u okviru istog modela zahteva. Za ovo se u modelu zahteva koristi matrica preslikavanja (Traceability Matrix), koja se u model (paket) dodaje opcijom Create a Traceability Matrix View iz Requirements toolbar-a (direktno iznad tabele zahteva).

U pitanju je tabela u čijim su redovima navedeni zahtevi, a u kolonama elementi sa kojima se oni povezuju. Za međusobno povezivanje zahteva potrebno je u properties dijalogu matrice u polju Traceability Matrix Type odabratи opciju Requirement, a za povezivanje sa elementima nekog drugog modela odabratи opciju Design (taj drugi model treba da je otvoren u browser-u).

	1 Overview	2 Use Cases	2.1 Identified Users	2.1.1 Customer	2.1.2 Administrator	2.2 Identified Use Cases	2.2.1 Registration	2.2.2 Log On	2.2.3 Consultation	2.2.4 Administration	2.2.4.1 Adding new publication	2.2.4.2 Removing an existing	2.2.4.3 Updating an existing publi	2.2.4.4 Updating the list of	3 Technical Requirements	3.1 Administration	3.1.1 Administrator accounts	3.1.1.1 First Administrator	3.1.1.2 Last Administrator
							✓	✓											
									✓										
										✓									
											✓								
												✓							
													✓						
														✓					
															✓				
																✓			
																	✓		
																		✓	
																			✓

Slika 8.1.7 Međusobno povezivanje zahteva



Slika 8.1.8 Odabir elemenata za kolone matrice

## POVEZIVANJE ZAHTEVA SA ELEMENTIMA DIZAJNA (NASTAVAK)

*Power designer omogućava kreiranje veze između modela zahteva i drugih modela kako bi se omogućilo njihovo jednostavno praćenje II*

Kolone i redovi se definišu odabirom opcije Select Rows/Columns iz Requirements toolbar-a. Na kartici Column Object Selection odabratи dugme Add New Column Object, te u dijalogu odabratи elemente koji treba da se prikažu u kolonama.

**Savet:** kada se odabere element za prikazivanje u koloni, PowerDesigner automatski selektuje i sve njegove veze, te je potrebno u istom koraku deselektovati te veze na njihovoj kartici. Na primer, nakon selekcije slučajeva korišćenja, na kartici Dependencies deselektovati sve veze.

Veza između zahteva i elementa modela dodaje se tako što se klikne u polje u preseku odgovarajuće vrste i kolone, te se u tom polju pritisne taster Space. Brisanje veze može se obaviti odabirom polja, te pritiskom na taster Space ili Delete. Ovako dodate veze mogu se videti i u Properties dijalogu zahteva, na kartici Traceability Links, gde ih je moguće i dodati.

**Savet:** dugme Display Only Empty Rows prikazuje samo prazne redove, što može da bude korisno za nalaženje zahteva koji nisu povezani ni sa jednim elementom.

Uspostavljene veze između zahteva i drugih elemenata se mogu videti i:

- u properties dijalogu zahteva, na kartici Traceability Links
- u properties dijalogu elementa povezanog sa zahtevom, na kartici Requirements.

Iz oba ova dijaloga je takođe moguće uspostaviti vezu između zahteva i datog elementa.

## ZADACI 11-14

### *Crtanje dijagrama slučajeva korišćenja (DSK) za dati slučaj primenom PowerDesigner alata*

1. Nacrtati dijagram slučajeva korišćenja koji će omogućiti korisniku da preko interneta rezerviše bioskopske ulaznice za željene filmove. Takođe je potrebno omogućiti korisniku stalan uvid u bioskopski repertoar i informacije o bioskopu u kojem željeni film igra. Nacrtati još jedan dijagram slučajeva korišćenja za proces rezervacije. Potrebno je uraditi kratak opis svakog učesnika i svakog slučaja korišćenja.
2. Nacrtati dijagram slučajeva korišćenja koji će prikazati funkcionisanje video kluba, posmatramo učesnike Član i Radnik u video klubu.
3. Nacrtati dijagrame slučajeva korišćenja za sledeće scenarije: a) kupovina u prodavnici, b) rad studentske službe, c) rad poštanskog šaltera, snimanje rada gradskog saobraćaja
4. Nacrtati dijagram slučajeva upotrebe za aktivnost po izboru. Potrebno je uraditi kratak opis svakog učesnika i svakog slučaja upotrebe. Minimum 2 učesnika i minimum 4-5 slučajeva upotrebe.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

*Dobro definisni zahtevi su osnovni preduslov za uspešan razvoj softverskog sistema.*

1. Zahtevi za softverskim sistemom određuju šta sistem treba da radi i definišu ograničenja njegovog rada i implementacije.
2. Funkcionalni zahtevi su iskazi o servisima koje sistem mora da obezbedi ili su opisi kakvo računanje mora da se izvrši.
3. Nefunkcionalni zahtevi često ograničavaju sistem u razvoju i upotrebljen proces razvoja. To mogu biti i zahtevi za proizvodom, organizacijski zahtevi, ili spoljni zahtevi. Oni se obično odnose na nova svojstva sistema, te se zbog toga odnose na sistem u celini.
4. Dokument sa zahtevima za softverom je usaglašen iskaz o zahtevima sistema. On treba da bude tako organizovan, da ga mogu da koriste i korisnici sistema, i inženjeri razvoja softvera.
5. Otkrivanje i analiza zahteva je jedan iterativan proces koji se može predstaviti spiralom aktivnosti: otkrivanje zahteva, klasifikacija i organizacija zahteva, pregovaranje oko zahteva, i dokumentovanje zahteva.
6. Proces validacije zahteva je proces provere validacije (ispravnosti) zahteva, njihove konzistentnosti, kompletnosti, realizma i proverljivosti.
7. Poslovne, organizacione i tehničke promene nepobitno vode ka promeni zahteva softverskog sistema. Upravljanje zahtevima je proces upravljanja i kontrolisanja ovih promena