Tehnologije i sistemi eUprave

Predavanja 07

Specifikacija zahteva

- Nakon što je poznat problem koji se rešava (poglavlje Uvod) i načini na koje su drugi rešavali sličan problem (odeljak Srodna rešenja), kao i tehnologije kojima raspolažete (odeljak Pregled korišćenih tehnologija), prelazi se na opisivanje mogućnosti koje se očekuju od vašeg rešenja.
- Poglavlje Specifikacija zahteva objašnjava šta se sve očekuje od rešenja problema iz perspektive u kojoj taj problem još nije rešen. Zbog toga se u ovom poglavlju ne navode nikakve tehnologije, nikakvi programerski izazovi, nikakav GUI.
- Sve što neko softversko rešenje treba da omogući, da bi se problem smatrao rešenim, potrebno je dokumentovati u ovom poglavlju.

Specifikacija zahteva

- Specifikacija zahteva bi trebalo da dokumentuje sva očekivanja budućeg korisnika (ili naručioca) softverskog rešenja.
- Sve što je specificarno u zahtevima, mora biti realizovano u rešenju. Sa druge strane, sve što nije specificarno smatraće se nepotrebnim ili suvišnim za sistem.
- U opisivanju zahteva nema mesta subjektivnim procenama (brzo, sporo, malo, mnogo, dobro, loše, …), ni nepreciznim formulacijama (unos korisnika, izlazak iz aplikacije, skidanje podataka, gašenje računara, ...)

Specifikacija zahteva

- Potrebno je da specifikacija zahteva sadrži dovoljno informacija da neko drugi može da reši isti problem, na sličan način.
- Zahtevi koji se stavljaju pred softversko rešenje mogu biti funkcionalni i nefunkcionalni, pa tako ovo poglavlje ima dva odeljka: Specifikacija funkcionalnih zahteva i Specifikacija nefunkcionalnih zahteva.
- Potrebno je na početku poglavlja u barem jednoj rečenici navesti šta to poglavlje sadrži, pa tek onda napraviti odeljke. Na primer, “U ovom poglavlju se objašnjeni funkcionalni i nefunkcionalni zahtevi softverskog rešenja predstavljenog u ovom radu”.

Specifikacija funkcionalnih zahteva

- Na početku odeljka Specifikacija funkcionalnih zahteva je potrebno najaviti šta taj odeljak sadrži. Na primer, “U ovom odeljku su opisani funkcionalni zahtevi koje je potrebno da ispunjava softversko rešenje za …”.
- U softverskom inženjerstvu se za specifikaciju funkcionalnih zahteva koriste UML dijagrami slučajeva korišćenja. Zato u ovom odeljku treba spomenuti sliku, prikazati je, a zatim navesti opise svih slučajeva korišćenja. Na primer, “Funkcionalni zahtevi ovog softverskog rešenja su predstavljeni UML dijagramom slučajeva korišćenja, kao što je prikazano na slici X.”.
- Neophodno je da tekst koji je sadržan na dijagramima bude približno iste veličine kao i tekst u ostatku dokumenta, da bi bilo čitljivo sve što je na tim dijagramima ispisano.

Slučajevi korišćenja

- Svaki od slučajeva korišćenja je potrebno opisati kako bi se preciziralo ko su im učesnici, koji su im preduslovi, od kojih se koraka sastoje, do kakvog rezultata vode, koji su mogući izuzeci.
- Najpreglednije je ako se ovi opisi prikažu tabelarno. Pri tome tabele moraju biti spomenute u tekstu i ispod svake tabele se mora nalaziti redni broj tabele i njen naziv (kao i kod slika).

Primer opisa slučaja korišćenja

Tabela 1 prikazuje opis slučaja korišćenja “Prijavljivanje”.

Naziv	Prijavljivanje
Učesnici	Korisnik
Preduslovi	-
Koraci	1. Korisnik bira opciju za prijavu 2. Korisnik unosi korisničko ime i lozinku 3. Korisnik potvrđuje unos
Rezultat	Korisnik je prijavljen na sistem
Izuzeci	Pogrešno korisničko ime ili lozinka

Tabela 1 - Opis slučaja korišćenja “Prijavljivanje”

Primeri loše formulisanih funkcionalnih zahteva

- Koraci:
  - Administrator traži korisnika po id-ju i ima mogućnost da trajno obriše korisnika.
  - Administrator ulazi u određeni zahtev i odlučuje o njegovom odobravanju.
- Rezultat:
  - Ako sva polja prođu validaciju, knjiga je kreirana i unosi se nova knjiga u bazu.
  - Ako su unete vrednosti tačne, to jest korisnik ne pstoji sa tim podacima, korisnik je dodat u bazu i može da se prijavi na sistem.
  - Nakon završenog unosa artikla, korisnik može odmah da vidi artikal u katalogu. Ako želi korisnik moće izvršiti njegovu izmenu ili brisanje.

Specifikacija nefunkcionalnih zahteva

- Pomoću specifikacije nefunkcionalnih zahteva se definišu svojstva softverskog rešenja koja su potrebna da bi se dati problem rešio, ali ne predstavljaju njegove funkcionalnosti.
- Najčešće se tiču nekih ograničenja, performansi, dizajna i slično.
- Nefunkcionalni zahtevi, kao i funkcionalni, moraju biti realizovani u softverskom rešenju da bi se konkretan problem rešio.

Specifikacija nefunkcionalnih zahteva

- Neke vrste nefunkcionalnih zahteva:
  - Performanse - brzina odziva sistema
  - Skalabilnost - promena performansi sa opterećenjem resursa
  - Portabilnost - ciljne platforme, prilagodljiv dizajn
  - Kompatibilnost - usklađenost sa standardima, formatima, sistemima (interoperabilnost), propisima
  - Raspolaganje resursima - ograničenja u korišćenju procesora i memorije
  - Pristupačnost - podrška za korisnike sa slabijim vidom, sluhom
  - Pouzdanost - otpornost na greške
  - Održavanje - otklanjanje grešaka i upravljanje sistemom
  - Bezbednost - zaštita podataka
  - Lokalizacija - višejezičnost
  - Upotrebljivost - lakoća korišćenja

Primeri loše sastavljenih nefunkcionalnih zahteva

- Veb sajt treba da bude postavljen na server koji će omogućiti efikasne i brze performanse. Što znači da korisnik tokom korišćenja aplikacije treba dobije brzo obrađene odgovore na akcju koju je izvršio.
- Neophodno je primenjivati standarde koji podatke čuvaju na konzistentan i siguran način. Što znači da podaci u sistemu treba da budu precizni.
- Koristiti Java programski jezik jer pored Pajton programskog jezika daje najbolju podršku preuzimanju resursa sa veba. Java uključuje biblioteke koje efikasno preuzimaju podatke i pri tome zahtevaju minimalno vremena i resursa.
- Prvo što korisnik vidi i može da uradi kada uđe na aplikaciju je pretraga. Pretraga se vrši tako što unesemo neophodne parametre i program će sam naći relevantne podatke.

Specifikacija dizajna

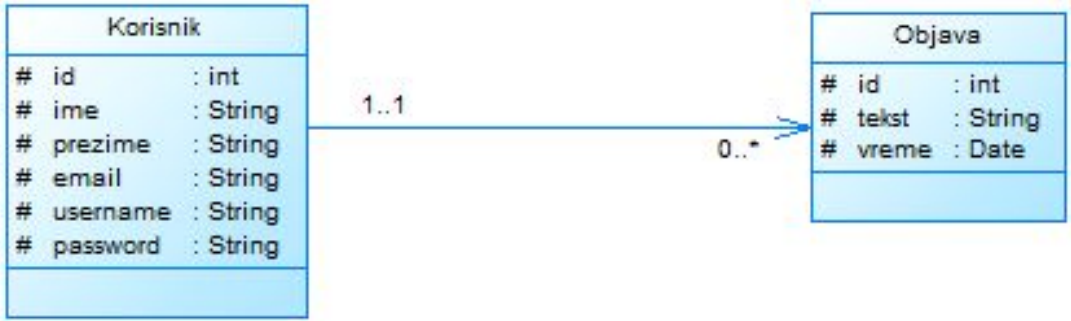
- U specifikaciji dizajna treba objasniti organizaciju sistema i kako bi trebalo da on bude izgrađen.
- Na početku poglavlja treba najaviti šta to poglavlje sadrži. Na primer, “Ovo poglavlje objašnjava dizajn softverskog rešenja za …”.
- Zatim se može preći na opis arhitekture odnosno objašnjenja od kojih celina (komponenti) se sastoji rešenje i kako one međusobno komuniciraju. U tome UML dijagrami komponenti (ili rasporeda) mogu biti korisni.
- Iz opisa arhitekture čitalac treba da dobije predstavu o tome kako je naše rešenje koncipirano (ali ne i pomoću kojih tehnologija i na koji način je implementirano)

Specifikacija dizajna

- Zatim treba objasniti pojedinosti dizajna sistema odnosno ideje kako kreirati komponente sistema.
- Za to su pogodni UML dijagrami klasa, aktivnosti i sekvenci.
- Na primer, treba objasniti:
  - kako je organizovan objektni model podataka (class diagram),
  - od kojih akcija se sastoje pojedine funkcije sistema (activity diagram),
  - kakve poruke i u kojem redosledu razmenjuju komponente (sequence diagram)
- Svaki od dijagrama je potrebno spomenuti u tekstu, a ispod svakog dijagrama je potrebno navesti redni broj slike i naziv slike.

Primer objašnjenja dijagrama klasa

Na slici 1 je pomoću UML dijagrama klasa predstavljen objektni model sistema.



Slika 1 - Dijagram klasa

Klasa Korisnik reprezentuje korisnike sistema i sadrži njihove lične podatke i podatke za autentifikaciju na sistem. Podaci o tekstovima koje korisnici objavljuju su modelovani klasom Objava. Kardinalitet relacije između ovih klasa omogućava korisniku da može imati više objava, dok svakoj objavi odgovara tačno jedan korisnik, odnosno njen autor.