UNIVERZITET METROPOLITAN BEOGRAD

SE322 - INŽENJERSTVO ZAHTEVA

Dokumentovanje zahteva

Lekcija 08

PRIRUČNIK ZA STUDENTE

SE322 - INŽENJERSTVO ZAHTEVA

Lekcija 08

DOKUMENTOVANJE ZAHTEVA

- Dokumentovanje zahteva
- Poglavlje 1: Predstavljanje i označavanje softverskih zahteva
- Poglavlje 2: Uzorak specifikacije softverskih zahateva
- Poglavlje 3: Specifikacija zahteva za agilni razvoj softvera
- Poglavlje 4: Karakteristike odličnih zahteva
- Poglavlje 5: Preporuke za pisanje zahteva
- Poglavlje 6: Vežba
- Poglavlje 7: Domaći zadatak
- Poglavlje 8: Projektni zadatak
- Zaključak

UVOD

Uvodne napomene

Jasna i efektivna komunikacija je ključni princip razvoja zahteva. To je komunikacija između onih koji imaju potrebe za softverom i onih koji treba da ga razviju, primene i verifikuju. Vešt poslovni analitičar treba da izabere najefektniji način komunikacije informacije o svakom tipu zahteva korisnicima te informacije.

Rezultat razvoja zahteva je dokumentovan sporazum među akterima o proizvodu koji treba da se razvije i primeni. Kao što smo pokazali ranije, dokument o viziji i okviru sadrži poslovne zahteve, zahtevi korisnika se zahvataju primenom slučajeva korišćenja ili korisničkih priča. Funkcionalni i nefunkcionalni zahtevi se najčešće nalaze u specifikaciji softverskog zahteva (SRS - Software Requirement Specification). Ovaj dokument je namenjen onima koji treba da projektuju, konstruišu i verifikuju (potvrde) razvijeno softversko rešenje. Zapisivanje zahteva na jedan organizovan način na koji ključni akteri projekta mogu da izvrše njihovu recenziju, obezbeđuje da oni znaju o čemu su se dogovorili,

U ovoj lekciji izložićemo svrhu, strukturu i sadržaj Dokumenta sa specifikacijom softverskih zahteva (SRS)-Mada se izlaže kao dokument, SRS ne mora da bude u formi dokumenta. U formi dokumenta, SRS ima sledeća ograničenja:

- teško je uneti opise atributa zajedno sa zahtevima,
- upravljanje promenama je nezgrapno,
- teško je zadržati istoriju verzija zahteva,
- nije lako da se izdvoji deo zahteva koji su alocirani u određenoj iteraciji, ili da se zadrži trag sa onim koji su jednom dobile saglasnost, ali su kasnije povučene ili poništene,
- teško je povezati zahteve sa drugim artifaktima projektovanja sistema,
- dupliranje zahteva koji logički odgovaraju i više različitih delova sistema dovode do problema sa održavanjem.

Kao alternativa, SRS možete organizovati u Excel-u, kao Wiki, u bazi podataka, ili u softverskom alatu za upravljanje zahtevima (RM - Requirement Management). Međutim, bez obzira na formu skladištenja informacija u zahtevima sistema, vi imate potrebu za istim informacijama. Uzorak SRS dokumenta koji se ovde izlaže je zbog toga koristan, jer daje strukturu potrebnih informacija, što je podsetnik za njihovo prikupljanje i organizovanje

UVODNI VIDEO

Trajanje video snimka: 1min 9sek

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 1

Predstavljanje i označavanje softverskih zahteva

VIDEO PREDAVANJE ZA OBJEKAT "PREDSTAVLJANJE I OZNAČAVANJE SOFTVERSKIH ZAHTEVA"

Trajanje video snimka: 28min 10sek

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PREDSTAVLJANJE SOFTVERSKIH ZAHTEVA

Progresivno poboljšanje detalja je ključni princip efektivnog razvoja zahteva.

Neki misle da ne treba gubiti vreme na dokumentovanju zahteva. Smatraju da kreiranje i održavanje dokumentacije o zahtevima ne daje mnogo vrednosti proizvodu. Međutim, trošak zapisivanja i uređivanja zahteva je mali u odnosu na trošak prikupljanja znanja ili reinženjeringa sistema u nekom trenutku u budućnosti. Tada dokument sa specifikacijom zahteva i njihovo modelovanje može da pomogne učesnicima u projektu da usmere svoja razmišljanja i da precizno postave stanje stvari, što obično same diskusije ostavljaju dvosmislenim. Takođe, u budućim sličnim projektima, preneseno znanje o zahtevima iz prethodnih projekata je vredno nasleđe koje daje veliku prednost firmama koje ga imaju. To smanjuje nove napore, jer ima dosta ponavljanje kod projekata, i sprečava da se neka teško stečena znanja zaborave.

Nikada nećete dobiti idealne zahteve. Kao što znate, pišete zahteve za određene korisnike. Učešće detalja, vrstu informacija koju dajete, i način na koji organizujete informacije - sve je podređeno potrebama korisnika za koje pišete dokumenta i projektnu dokumentaciju, jer je pišete u skladu sa potrebama tih korisnika. Analitičari obično pišu dokumenta iz svoje perspektive, međutim, dobro bi bilo da je pišu iz perspektive tih korisnika, tako da dokumentacija bude ono što njima treba. Zato je dobro da predstavnik korisnika dokumenta o specifikaciji zahteva vrši njegovu recenziju. Progresivno poboljšanje detalja je ključni princip efektivnog razvoja zahteva. Najčešće nije potrebno u ranim fazama projekta ući u detalje svakog zahteva.

Umesto toga, razmišljajte o nivoima zahteva. Na početku, treba da znate zahteve u meri koja je potrebna da napravite prioritete među njima i da ih rasporedite po izdanjima softvera ili po iteracijama (ako primenjujete agilne metode razvoja). Kasnije, kada treba da date

inženjerima razvoja i programerima vaše zahteve, vi ćete ih dopuniti neophodnim detaljima. Time izbegavate ponovni rad na njima kada kasnije imate više saznanja, u odnosu na rane faze projekta.

Ni najbolji dokument o zahtevima ne može da zameni diskusije o zahtevima tokom trajanja projekta. Držite otvorene komunikacione linije između biznis analitičara, razvojnog tima, predstavnika kupaca, i drugih aktera projekta, kako bi brzo razrešili mnogobrojne probleme koji će se javiti tokom rada na projektu.

Možete predstaviti softverske zahteve na nekoliko načina:

- dobro strukturisan i pažljivo napisan **dokument u prirodnom jeziku**
- **primena vizuelnih modela** koji ilustruju procese transformacije, stanja sistema, i promene između njih, beza među podacima, logičke tokove, i sl.
- **formalne specifikacije** koje definišu zahteve upotrebom preciznijih matematičkih jezika za specifikaciju.

Ovde ćemo govoriti o praktičnom načinu dokumentovanja zahteva, primenom kombinacije prva dva načina.

SPECIFIKACIJA SOFTVERSKIH ZAHTEVA

Specifikacija softverskih zahteva sadrži funkcije i sposobnosti koje softverski sistem mora da obezbedi, njegove karakteristike, i ograničenja koja mora da poštuje.

Specifikacija softverskih zahteva (SRS -Software Requirement Specification) sadrži funkcije i sposobnosti koje softverski sistem mora da obezbedi, njegove karakteristike, i ograničenja koja mora da poštuje. On treba da opiše u potrebnoj meri, ponašanje sistema pod određenim uslovima, kao i poželjne kvalitete sistema, kao što su performanse, bezbednost i upotrebljivost. SRS je osnova za izradu drugih dokumenta, kao što je planiranje sistema, projektovanje, i kodiranje. On je takođe temelj za pripremu testiranja softvera, i za izradu korisničke dokumentacije. Međutim, SRS ne bi trebalo da sadrži detalje o projektnom rešenju softvera, njegovoj konstrukciji, testiranju i o upravljanju projektom, sem ograničenja koje projektovanje i implementacija može da zahteva. I kod primene agilnih metoda razvoja softvera, potrebne su informacije koje sadrži jedan dobar SRS dokument. Oni ne prikupljaju na jedan organizovan način informacije kao što se radi sa SRS dokumentom, ali SRS dokument ih podseća na vrstu znanja koje treba da istražuju.

Kako različiti akteri imaju potrebe za različitim informacijama, mi na predmetu predlažemo tri vrste dokumenta:

- Dokument o viziji i okviru projekta,
- Dokument o zahtevima korisnika, i
- Dokument o specifikaciji softverskih zahteva (SRS)

Korisnici SRS dokumenta su:

- kupci, marketing, i prodavci, koji treba da znaju kakav proizvod treba da dobiju, odn. da isporuče,
- Menadžeri projekta zasnivaju na zahtevima svoje procene rokova, rada, i resursa
- Razvojni tim softvera saznaje šta treba da razvije
- Testeri, koji ga koriste da bi razvili testove bazirane na zahtevima, planove testiranja i test procedure.
- Osoblje za održavanje i podršku ga koriste da razumevanje šta koji deo softvera radi.
- Pisci dokumenata baziraju korisnička uputstva na SRS dokumentu
- Zaduženi za obuke upotrebljavaju SRS i korisničku dokumentaciju da bi razvila materijal obuke
- Pravnici da zahtevi zadovoljavaju zakone i propise.
- Proizvođači baziraju svoj rad i pravno su vezana, na SRS

Ako se nešto ne nalazi u SRS, niko to ne treba da očekuje u svojstvima proizvoda.

OBELEŽAVANJE ZAHTEVA

Svaki zahtev treba da ima jedinstveni identifikator. On se koristi kod svakog pozivanja na odgovarajući zahtev.

Važno je da pišete i organizujete dokument sa specifikacijom softverskih zahteva (SRS) tako da ga razumeju svi akteri projekta. Da bi to postigli, ovde se daju sledeći saveti:

- Koristiti odgovarajući uzorak (sadržaj dokumenta) da bi organizovali sve potrebne informacije.
- Obeležite i uredite konsistentno sekcije, podsekcije, i pojedinačne zahteve.
- Koristite vizuelna sredstva (boldiran, italik, podvučena i obojena slova, kao i odgovarajuće tipove slova - fontove) radi dobijanja uređenijeg i jasnijeg teksta.
- Kreirajte sadržaj dokumenta
- Dajte brojeve svim tabelama i slikama u dokumentu, upišite njihove naslove.
- Koristite svojstvo povezivanja referenci, koje nude savremeni sistemi za pripremu elektronskih dokumenata (kao MS Word).
- Koristite linove ka pojedinim delovima SRS dokumenta
- Uključite vizualno predstavljanje informacije uvek kada to možete.

Obeležavanje zahteva

Svaki zahtev treba da ima jedinstveni identifikator. On se koristi kod svakog pozivanja na odgovarajući zahtev. To olakšava i kolaboraciju unutar tima.

Broj redosleda

Najjednostavniji način obeležavanja je davanje broja po redosledu pojavljivanja. Na primer, UV-9 (za slučaje korišćenja) ili DR-29 /za funkcionalne zahteve), Prefiks označava tip zahteva. Softverski alati za uređivanje zahteva automatski dodeljuju brojne oznake, koje su neponovljive.

Hijerarhijsko dodeljivanje brojeva

Ovo se najčešće koristi. Ako je broj nekog funkcionalnog zahtevaFR3.2, onda će svi zahtevi koji iz njega proizilaze dobiti istu oznaku (3.2), ali će nadalje dodavati, iza tačke, nove brojeve (npr. 3.2.1, 3.2.2, 3.2.2.1, itd.).

Svi editori teksta omogućavaju automatsku hijerarhijsku numeraciju teksta. Međutim, nastaje problem kod brisanja nekog zahteva. Kada se obriše zahtev, obriše se i njegova oznaka, sve naredne brojčane oznake se smanjuju za 1, a to se kosi sa našim zahtevom da se brojevi zahteva, kada se jedanput definišu, više ne menjaju, da se ne bi poremetile reference na njega u drugim tekstovima i dokumentima.

Problem se ne rešava, ali se lakše radi, kada se vrši kombinacija oznaka i rednih brojeva, pri označavanju zahteva. Na primer, ED-1, Ed-2, itd..

HIJERARHIJSKI TEKSTUALNI TAGOVI

Hijerarhijski tekstualni tagovi su strukturisani, ukazuju na značenje i ne menjaju se pri dodavanju, brisanju ili pomeranju drugih zahteva

Hijerarhijski tekstualni tagovi

Hijerarhijski tekstualni tagovi su strukturisani, ukazuju na značenje , i ne menjaju se pri dodavanju, brisanju ili pomeranju drugih zahteva. Ovaj metod isto zgodan kod obeležavanje poslovnih pravila. Primer:

Print.ConfirmeCompies

Ova oznaka se odnosi na sledeći zahtev: "Sistem će zahtevati od korisnika da potvrdi bilo koji zahtev da štampa više od 10 kopija. " Tako zvana oznaka ukazuje da se radi o funkciji štampanja, i da je povezan sa brojem kopija koji treba štampati.

Na primer, oznaka:

Product.Cart.Discount.Error.Shipping

Ovo može da znači da se proizvod (Product) kupuje preko veb sajta koji koristi karte za kupovinu (Cart), sa određenim popustom (Discount), pri čemu sistem daje i oznaku greške (Error), ako kupac unese pogrešnu informaciju, a pri kupovini se dodaju troškovi isporuke proizvoda (Shipping). Ovakvim oznaka se eliminiše problem sa održavanjem, koje ima hijerarhijsko označavanje , ali su tagovi duži i treba im davati smislena imena. Ove oznake se mogu kombinovati i sa brojevima, na primer:
Product.Card.01, Product.Crad.02, ...

Rad sa nekompletnim informacijama

Ponekad vi znate da vam privremeno nedostaje neka informacija. Onda je možete obeležiti sa TBD (to be determined - na engleskom), da bi označili ove rupe u znanju. Potrebno je planirati otklanjanje svih ovakvih "rupa" pre nego što dođe do primene skupa zahteva u kojima se pominju.

Poglavlje 2

Uzorak specifikacije softverskih zahateva

SADRŽAJ SRS DOKUMENTA

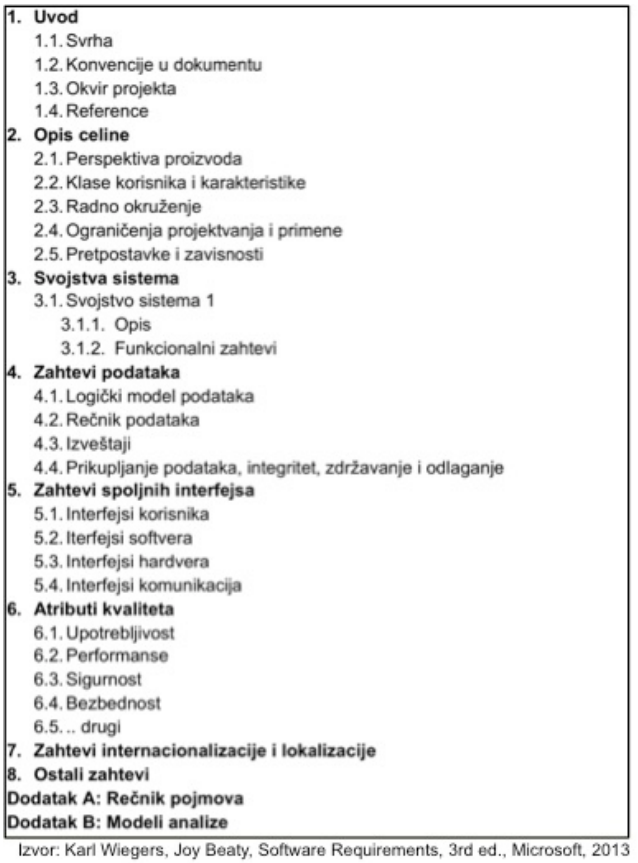
Postoje različiti uzorci SRS dokumenta, jer postoje i različite veličine projekata razvoj softvera. Na vama je da izaberete onaj koji najviše odgovara vašoj organizaciji i projektu.

Postoje različiti uzorci SRS dokumenta, jer postoje i različite veličine projekata razvoja softvera. Na vama je da izaberete onaj koji najviše odgovara vašoj organizaciji i projektu. Na slici 1 prikazan je uzorak SRS dokumenta koji se pokazao dobrim u mnogim tipovima projekata.

Izbegnite duple informacije u dokumentu, tj. jednu informaciju upišite samo u jedan deo dokumenta. Ako je potrebna i u drugim delovima, dajte samo referencu ka informaciji u dokumentu. Koristite reference i linkove u dokumentu, da bi se lakše nalazile potrebne informacije.

Pri razvoju dokumenta, koristite verzije i vodite računa o njihovim oznakama i ažuriranju dokumenata. Posle internog objavljivanja verzije, svaka izmena u toj verziji mora da se sadrži u sledećoj verziji, tj. posle prve izmene, odmah promenite verziju dokumenta na kojoj radite, i koja će biti sledeća javna verzija dokumenta. Uključite i istoriju verzija u dokument.

Nadalje ćemo ukratko opisati svaku sekciju podsekciju dokumenta



Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 2.1 Sadržaj dokumenta sa specifikacijom softverskih zahteva (SRS)

1. UVOD

Uvod predstavlja pregled koji će pomoći čitaocu da razume kako je SRS organizovan i kako da ga koristi

1. UVOD

Uvod predstavlja pregled koji će pomoći čitaocu da razume kako je SRS organizovan i kako da ga koristi.

1.1 Svrha

Identifikujte proizvod čiji su zahtevi za softver navedeni u ovom dokumentu, uključujući broj revizije ili izdanja. Opišite različite vrste korisnika kojima je dokument namenjen, kao što su programeri, rukovodioci projekata, marketinško osoblje, korisnici, testeri i pisci dokumentacije.

1.2 Konvencije o dokumentima

Opišite sve korištene standarde ili tipografske konvencije, uključujući značenje određenih stilova teksta, isticanja ili notacija. Ako ručno označavate jedinstvene identifikatore zahteva, ovde možete odrediti format za svakoga ko ga treba kasnije dodati.

1.3 Obim projekta

Navedite kratak opis softvera koji se specificira i njegovu svrhu. Povežite softver sa korisničkim ili korporativnim ciljevima i sa poslovnim ciljevima i strategijama. Ako su na raspolaganju zasebna vizija i opseg ili sličan dokument, obratite se njemu, a ne da duplirate njegov sadržaj ovde. SRS koji određuje postepeno puštanje proizvoda koji se razvija treba da

sadrži sopstvenu izjavu o obimu kao podskup dugoročne strateške vizije proizvoda. Možete da date rezime visokih nivoa glavnih funkcija koje izdanje sadrži ili značajnih funkcija koje obavlja.

1.4 Reference

Lista svih dokumenata ili drugih izvora na koje se odnosi ovaj SRS. Uključite hiperveze ako su na postojanoj lokaciji. Oni mogu uključivati vodiče za stil korisničkog interfejsa, ugovore, standarde, specifikacije sistemskih zahteva, specifikacije interfejsa ili SRS za srodni proizvod. Dajte dovoljno informacija kako bi čitalac mogao pristupiti svakoj referenci, uključujući njen naslov, autora, broj verzije, datum i izvor, lokaciju za skladištenje ili URL.

2. OPŠTI OPIS

Opšti opis predstavlja pregled proizvoda i okruženja na visokom nivou apstrakcije, očekivanih korisnika i poznata ograničenja, pretpostavke i zavisnosti.

2. OPŠTI OPIS

Ovaj odeljak predstavlja pregled proizvoda i okruženja na visokom nivou apstrakcije, očekivanih korisnika i poznata ograničenja, pretpostavke i zavisnosti.

2.1 Perspektiva proizvoda

Opišite kontekst i poreklo proizvoda. Da li je to sledeći član rastuće linije proizvoda, sledeća verzija zrelog sistema, zamena za postojeću aplikaciju ili potpuno novi proizvod? Ako ovaj SRS definiše komponentu većeg sistema, navedite kako se ovaj softver odnosi na celokupni sistem i identifikujte glavne interfejse između ova dva. Razmotrite uključivanje vizuelnih modela poput kontekstnog dijagrama ili mape ekosistema da biste pokazali odnos proizvoda prema drugim sistemima

2.2 Klase i karakteristike korisnika

Identifikujte različite klase korisnika za koje očekujete da će koristiti ovaj proizvod i opišite njihove relevantne karakteristike. Neki zahtevi mogu se odnositi samo na određene klase korisnika. Identificirajte omiljene klase korisnika. Korisničke klase predstavljaju podskup zainteresovanih strana opisan u dokumentu o viziji i opsegu. Opisi korisničkih klasa su izvor koji se može ponovo koristiti. Ako su dostupni, možete uključiti opise korisničkih klasa tako što ćete ih jednostavno pokazati u katalogu matične klase umesto dupliranja podataka ovde.

2.3 Operativno okruženje

Opišite okruženje u kojem će softver raditi, uključujući hardversku platformu; operativni sistemi i verzije; geografske lokacije korisnika, servera i baza podataka; i organizacije koje nude odgovarajuće baze podataka, servere i veb lokacije. Navedi sve ostale softverske komponente ili aplikacije sa kojima sistem mora mirno koegzistirati. Ako je potrebno razviti opsežne radove na tehničkoj infrastrukturi zajedno sa razvojem novog sistema, razmislite o stvaranju zasebne specifikacije za infrastrukturu da biste je detaljno opisali.

2. OPŠTI OPIS (NASTAVAK)

Nastavak uputstva za pripremu poglavlja Opšti opis

2.4 Ograničenja u projektovanju i primeni

Opišite sve faktore koji će ograničiti mogućnosti dostupne programerima. Oni mogu uključivati: korporativne ili regulatorne politike; ograničenja hardvera (vreme i memorijski zahtevi); interfejs do drugih aplikacija; specifične tehnologije, alate i baze podataka koje se koriste; zahtevi ili ograničenja programskog jezika.

2.5 Pretpostavke i zavisnosti

Navedi sve pretpostavljene faktore (za razliku od poznatih činjenica) koji bi mogli uticati na zahteve koji su navedeni u SRS. Oni mogu da uključuju treće ili komercijalne komponente koje planirate da koristite, ponovo koristite očekivanja, probleme oko razvojnog ili operativnog okruženja ili ograničenja. Na projekat bi moglo uticati ako su te pretpostavke netačne, ne dele se ili se menjaju. Takođe identifikujte sve zavisnosti koje projekat ima od spoljnih faktora van njegove kontrole

3. FUNKCIJE SISTEMA

Ovaj obrazac ilustruje organizovanje funkcionalnih zahteva za proizvod prema karakteristikama sistema, glavnim uslugama koje proizvod pruža.

3. FUNKCIJE SISTEMA

Ovaj obrazac ilustruje organizovanje funkcionalnih zahteva za proizvod prema karakteristikama sistema, glavnim uslugama koje proizvod pruža. Ovaj odjeljak možete radije organizovati prema slučaju, načinu rada, korisničkoj klasi, klasi predmeta, funkcionalnoj hijerarhiji, stimulaciji, odgovoru ili kombinacijama istih, bez obzira što ima najlogičniji smisao za vaš proizvod.

3.1 Funkcija sistema X

Nemojte zaista reći „Funkcija sistema X.“ Navedite ime funkcije u samo nekoliko reči.

3.1.1 Opis

Navedite kratak opis funkcije i navedite da li je ona visokog, srednjeg ili niskog prioriteta.

3.1.2 Sekvence stimulusa / odgovora

Lista sekvencija korisničkih radnji i sistemskih odgovora koji podstiču ponašanje definisano za ovu značajku. Oni će odgovarati elementima dijaloga koji su povezani sa slučajevima upotrebe.

3.1.3 Funkcionalni zahtevi

Podesite specifične funkcionalne zahteve povezane sa ovoim svojstvom. Ovo su softverske mogućnosti koje moraju biti implementirane da korisnik izvrši usluge ove funkcije ili da izvrši slučaj upotrebe. Opišite kako proizvod treba da reaguje na očekivane uslove greške. Koristite „TBD“ kao rezervirano mesto da naznačite kada potrebne informacije još uvek nisu dostupne

3.2 Funkcija sistema 2 (i tako dalje)

.....

4. ZAHTEVI ZA PODATKE

Ovaj odeljak opisuje različite aspekte podataka koje će sistem koristiti kao ulaze, obrađivati na neki način ili kreirati kao izlaze.

4. ZAHTEVI ZA PODATKE Ovaj odeljak opisuje različite aspekte podataka koje će sistem koristiti kao ulaze, obrađivati na neki način ili kreirati kao izlaze.

4.1 Logički model podataka

Model podataka je vizuelni prikaz objekata podataka i zbirki koje će sistem obraditi i odnosa između njih. Uključite model podataka za poslovne operacije kojima se bavi sistem ili logičku reprezentaciju za podatke kojima će sam sistem manipulisati. Modeli podataka najčešće se kreiraju kao dijagram odnosa entiteta.

4.2 Rečnik podataka

Rečnik podataka definiše sastav podaktovnih struktura i značenje, vrstu podataka, dužinu, format i dopuštene vrednosti za elemente podataka koji čine te strukture. U mnogim slučajevima je bolje da skladištite rečnik podataka kao poseban artefakt, umesto da ga ugrađujete u sredinu SRS-a. To takođe povećava njegov potencijal ponovne upotrebe u drugim projektima.

4.3 Izveštaji

Ako će vaša aplikacija generisati bilo koji izveštaj, identifikujte ih ovde i opišite njihove karakteristike. Ako izveštaj mora biti u skladu s određenim unapred definisanim izgledom, ovde možete to specificirati kao ograničenje, možda primerom. U suprotnom, usredsredite se na logičke opise sadržaja izveštaja, redosled sortiranja, ukupnog nivoa i tako dalje, odlažući detaljan izgled izveštaja u fazi projektovanja.

4.4 Prikupljanje podataka, integritet, zadržavanje i odlaganje

Ako je relevantno, opišite kako se podaci prikupljaju i održavaju. Navedite sve zahteve koji se odnose na potrebu zaštite integriteta podataka sistema. Identifikujte bilo koje posebne tehnike koje su neophodne, kao što su rezervne kopije, kontrolne tačke, dupliranje (mirroring) ili verifikacija tačnosti podataka. Državne politike koje sistem mora primenjivati ili za čuvanje ili za uklanjanje podataka, uključujući privremene podatke, metapodatke, preostale podatke (kao što su izbrisani zapisi), keširane podatke, lokalne kopije, arhive i privremene sigurnosne kopije.

5. ZAHTEVI ZA SPOLJNE INTERFEJSE

Ovaj odeljak sadrži informacije kojima se osigurava da će sistem pravilno komunicirati sa korisnicima i sa spoljnim hardverskim ili softverskim elementima.

5. ZAHTEVI ZA SPOLJNE INTERFEJSE

Ovaj odeljak sadrži informacije kojima se osigurava da će sistem pravilno komunicirati sa korisnicima i sa spoljnim hardverskim ili softverskim elementima.

5.1 Korisnički interfejsi

Opišite logičke karakteristike svakog interfejsa između softverskog proizvoda i korisnika. Ovo

može da uključuje uzorke slika na ekranu, bilo koje GUI standarde ili vodiče za porodični stil proizvoda koji se moraju pridržavati, ograničenja izgleda ekrana, standardne tastere i funkcije (npr. Pomoć) koje će se pojavljivati na svakom ekranu, prečice na tastaturi, standardi prikazivanja greške i uskoro. Definišite softverske komponente za koje je potreban korisnički interfejs. Pojedinosti o dizajnu korisničkog interfejsa treba da budu dokumentovane u posebnoj specifikaciji korisničkog interfejsa.

5.2 Softverski interfejsi

Opišite veze između ovog proizvoda i drugih softverskih komponenti (identifikovanih imenom i verzijom), uključujući ostale aplikacije, baze podataka, operativne sisteme, alate, biblioteke, veb lokacije i integrisane komercijalne komponente. Navedite svrhu, formate i sadržaj poruka, podataka i kontrolnih vrednosti koje se razmenjuju između softverskih komponenti. Navedite preslikavanja ulaznih i izlaznih podataka između sistema i sve prevode koji su potrebni da bi podaci prešli iz jednog sistema u drugi. Opišite usluge potrebne od ili od spoljnih softverskih komponenti i prirodu komunikacije interkomponenata. Identifikujte podatke koji će se razmenjivati između ili deliti između komponenti softvera. Navedite nefunkcionalne zahteve koji utiču na interfejs, kao što su nivoi usluga za vreme i frekvencije odgovora ili sigurnosne kontrole i ograničenja.

5. ZAHTEVI ZA SPOLJNE INTERFEJSE (NASTAVAK)

Ovo je nastavak uputstva za popunjavanje odeljka 5. Zahtevi za spoljne interfejse

5.3 Hardverski interfejsi

Opišite karakteristike svakog interfejsa između softverske i hardverske (ako postoje) komponente sistema. Ovaj opis može uključivati podržane tipove uređaja, podatke i kontrolne interakcije softvera i hardvera i komunikacijske protokole koji će se koristiti. Navedite ulaze i izlaze, njihove formate, njihove važeće vrednosti ili raspone i sve probleme sa vremenom koji programeri moraju biti svesni. Ako su ove informacije opsežne, razmislite o kreiranju posebnog dokumenta specifikacije interfejsa.

5.4 Komunikacijski interfejsi

Navedite zahteve za sve komunikacione funkcije koje će proizvod koristiti, uključujući e-poštu, veb pregledač, mrežne protokole i elektronske obrasce. Definišite bilo koje relevantno formatiranje poruke. Navedite probleme sigurnosti ili šifrovanja komunikacije, brzine prenosa podataka, rukovanje rukama i mehanizme sinhronizacije. Navedite bilo kakva ograničenja oko ovih interfejsa, kao što su da li su prilozi e-pošte prihvatljivi ili ne.

6. ATRIBUTI KVALITETA

Ovde se definišu nefukcionalni zahtevi

6. ATRIBUTI KVALITETA

6.1 Upotrebljivost Navedite sve zahteve u vezi sa karakteristikama zbog kojih će softver izgledati kao "user-friendli". Upotrebljivost obuhvata jednostavnost korišćenja, jednostavnost

učenja; pamćenje; izbegavanje grešaka, rukovanje i oporavak; efikasnost interakcija; pristupačnost; i ergonomija. Ponekad se mogu sukobiti jedno sa drugim, kao i sa lakoćom korišćenja u odnosu na lakoću učenja. Navedite sve standarde ili smernice za dizajn korisničkog interfejsa kojima se aplikacija mora uskladiti.

6.2 Performanse

Zahtevi performansi za različite operacije sistema. Ako različiti funkcionalni zahtevi ili karakteristike imaju različite zahteve za performansama, primereno je da se ciljevi performansi tačno odrede odgovarajućim funkcionalnim zahtevima, a ne da se prikupljaju u ovom odeljku.

6.3 Bezbednost

Navedite sve zahteve u vezi sa pitanjima bezbednosti ili privatnosti koji ograničavaju pristup ili upotrebu proizvoda. Oni se mogu odnositi na fizičku, bezbednost podataka ili softver. Sigurnosni zahtevi često potiču iz poslovnih pravila, pa identifikujte sve sigurnosne ili privatne politike ili propise kojima se proizvod mora pridržavati. Ako su oni dokumentovani u skladištu poslovnih pravila, samo ih pogledajte.

6.4 Bezbednost

Navedite zahteve koji se odnose na mogući gubitak, oštećenje ili štetu koja može proizaći iz upotrebe proizvoda. Definišite sve zaštitne mere ili radnje koje se moraju preduzeti, kao i potencijalno opasne radnje koje se moraju sprečiti. Identifikujte sve sigurnosne sertifikate, politike ili propise kojima se proizvod mora uskladiti.

6.5 [Ostali po potrebi]

Kreirajte poseban odeljak u SRS-u za svaki dodatni atribut kvaliteta proizvoda da biste opisali karakteristike koje će biti važne bilo kupcima ili programerima. Mogućnosti uključuju dostupnost, efikasnost, instalabilnost, integritet, interoperabilnost, izmenljivost, prenosivost, pouzdanost, upotrebljivost, robusnost, skalabilnost i proverljivost. Napišite ove da budu specifične, kvantitativne i proverljive. Razjasnite relativne prioritete za različite atribute, kao što je sigurnost nad performansama.

7. USLOVI INTERNACIONALIZACIJE I LOKALIZACIJE

Odeljak daje uputstvo za primenu internacionalizaciju i lokalizaciju

7. USLOVI INTERNACIONALIZACIJE I LOKALIZACIJE

Zahtevi internacionalizacije i lokalizacije osiguravaju da će proizvod biti pogodan za upotrebu u nacijama, kulturama i geografskim lokacijama koje nisu one u kojima je stvoren. Takvi zahtevi mogu da reše razlike u: valuti; formatiranje datuma, brojeva, adresa i telefonskih brojeva; jezik, uključujući nacionalne pravopisne konvencije na istom jeziku (poput američkog naspram britanskog engleskog), korišćene simbole i skupove znakova; određeno ime i prezime; vremenske zone; međunarodni propisi i zakoni; kulturna i politička pitanja; korišćene veličine papira; utezi i mere; električni naponi i oblici utikača; i mnogi drugi

8. OSTALI USLOVI

Ovaj odeljak sadrži informacije o zakonskim, regulatornim ili finansijskim uslovima za rad softvera, kao i informacije o primenjenim industrijskim stan

8. OSTALI USLOVI

Primeri su: zakonski, regulatorni ili finansijski uslovi i zahtevi za standardi; zahtevi za instalaciju proizvoda, konfiguraciju, pokretanje i gašenje; i zahteve za evidentiranje, nadgledanje i reviziju. Umesto da sve to kombinujete pod „Ostalo“, dodajte sve nove odeljke u predložak koji se odnose na vaš projekat. Propustite ovaj odeljak ako su svi vaši zahtevi smešteni u drugim delovima.

DODACI

Dodatak A: Rečnik

Definišite sve specijalizovane pojmove koje čitalac mora da zna da bi shvatio SRS, uključujući skraćenice i skraćenice. Prepišite svaki akronim i navedite njegovu definiciju. Razmislite o stvaranju višestrukog upotrebnog pojma na nivou preduzeća koji obuhvata više projekata i koji sadrži reference po bilo kojim uslovima koji se odnose na ovaj projekat.

Dodatak B: Modeli analize

Ovaj opcioni odeljak uključuje ili ukazuje na relevantne modele analize kao što su dijagrami protoka podataka, stabla funkcija, dijagrami stanja i dijagrami odnosa entiteta. Možda biste radije da umetnete određene modele u relevantne odeljke specifikacije umesto da ih prikupljate na kraju.

Poglavlje 3

Specifikacija zahteva za agilni razvoj softvera

VIDEO PREDAVANJE ZA OBJEKAT "SPECIFIKACIJA ZAHTEVA ZA AGILNI RAZVOJ SOFTVERA"

Trajanje video snimka: 40min 3sek

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

SPECIFIKACIJA ZAHTEVA PRI AGILNOM RAZVOJU SOFTEVERA

Umesto funkcionalnih zahteva, najčešće se razvoj upravlja prema postavljenim testovima prihvatanja nastalim iz korisničkih priča.

Pri agilnim razvoju softvera najviše se koriste priče korisnika radi utvrđivanja zahteva. Svaka priča je jedan iskaz o potrebama korisnik ili o funkciji koju celi sistem treba da ima. Kada se prikupi dovoljan skup priča, pravi se njihov prioritet i raspoređivanja po iteracijama. U slučaju većih korisničkih priča koje se ne mogu realizovati u jednoj iteraciji, dele se na više manjih priča i raspoređuju po iteracijama, zapisivanje priča korisnika se mogu koristiti kartice ili softverski alati za rad sa pričama korisnika.

Kod svake iteracije akteri projekta diskutuju i definišu nove detalje priča korisnika. Znači, zahtevi se ažuriraju i detaljišu u vreme kada se primenjuju. Testiranje se vrši i u budućim iteracijama, radi provere.

U tom slučaju umesto funkcionalnih zahteva, najčešće se razvoj upravlja prema postavljenim testovima prihvatanja nastalim iz korisničkih priča. Testovi se rade u istoj iteraciji u kojoj se implementira priča korisnika. Treba da pokriju i očekivano ponašanje i izuzetke, tj. reakciju sistema u izuzetnim slučajevima, pokrenuti nekim događajima. Ako tim odstupi od početnih priča korisnika, onda od dokumentacije ostaju samo testovi prihvatanja.

Testovi se mogu pisati na kartama, ali se mogu unositi i preko alata za testiranje. Testovi se automatski izvode radi primene regresionog testiranja.

Nefunkcionalni zahtevi se ne unose kroz priče korisnika, već kao ograničenja. Alternativno, mogu se zapisati kao nefunkcionalni zahtevi koji se povezuju sa odgovarajućom pričom korisnika u vidu testa prihvatanja i kriterijuma za ocenjivanje testa. Razvojni tim može da

koristi i modele analize za predstavljanje znanja o radu sistema. Samo od razvojnog tima zavisi koji će metod izabrati za specifikaciju zahteva. Važno je da je izabran metod dovoljno dobar za konstrukciju sledećeg dela softvera pri čemu se radi sa prihvatljivom dozom rizika. Formalni nivo i nivo detaljisanja koji će dokument o zahtevima imati zavisi od sledećeg:

- od proširenja neformalne, just-in-time verbalne i vizuelne komunikacije između kupaca i razvojnog tima, koji treba da omogući dovoljno informacija za razvoj proizvoda u novoj iteraciji.
- od proširenje sinhronizovane neformalne komunikacije među članovima tima
- zavisno od potrebe da se za buduće projekte zadrži stečeno znanje, radi održavanja, reinženjering aplikacija, verifikaciju, iili radi ugovornih obaveza i provera ispunjenja ugovora.
- od mere kojom testovi prihvatanja mogu da budu efektivna zamena opisa očekivanog ponašanja sistema
- od stepena do kog ljudska memorija može da zameni pisano predstavljanje.

Poglavlje 4

Karakteristike odličnih zahteva

KARAKTERISTIKE ISKAZA O ZAHTEVIMA

Kompletnost, tačnost, ostvarljivost, potreba, usklađenost sa prioritetima, nedvosmislenost, proverljivost - to su karakteristike dobro definisanih zahteva.

Kompletnost Svaki zahtev treba da sadrži dovoljno informacija potrebnih da ih korisnik razume. U slučaju funkcionalnih zahteva, to znači da informacija mora da zadovolji potrebe programera kako bi pravilo primenio zahtev. Nedostajuće informacije se moraju uneti pre početka konstrukcije softvera.

Tačnost Svake mogućnosti kroz koje zahtev može da opiše sposobnosti sistema da tačno zadovolje neke potrebe aktera uz jasan opis funkcionalnosti koja se mora ostvariti. Tačnost proverava onaj koji je i zahtevao određeni zahtev. To može biti i slučaj korišćenja, poslovno pravilo, inicijalni dokument, lista zahteva sa visokim nivoom apstrakcije ili neki drugi dokument. Predstavnici korisnika bi trebalo da recenziraju zahteve.

Ostvarljivost Zahtev treba da bude ostvarljiv sa poznatim sposobnostima i ograničenjima sistema i njegovog radnog okruženja, kao i u okviru ograničenja projekta: rokovi, budžet i ljudi. Inženjer razvoja koji učestvuje u utvrđivanju zahteva treba da vrši proveru regularnosti zahteva. Ostvarljivost zahteva se vrši na dva načina: primenom prototipova za proveru koncepta i primenom inkrementalnog razvoja. Ako se zahtev odbaci kao neostvarljiv, onda treba proveriti posledicu na viziju i okvir projekta.

Potreba Svaki zahtev treba da opisuje sposobnost sistema koje donosi poslovnu vrednost akterima projekta, da se proizvod razlikuje od drugih na tržištu, ili ostvaruje usaglašenost sa standardima i drugom regulativom. Svaki zahtev treba da potiče iz izvora koji je ovlašćen da postavlja i definiše zahteve. Povežite sve funkcionalne i nefunkcionalne zahteve sa njihovim izvorima, da bi ovo proverili. Mora se povezati svaki zahtev sa poslovnim ciljevima kako bi se pokazalo da je on potreban.

Usklađenost sa prioritetima Dajte prioritet zahtevima koji su najvažniji za ostvarenje željene vrednosti. Dodelite prioritet svakom funkcionalnom zahtevu, zahtevu korisnika, scenariju slučaja korišćenja, ili svojstvu, kako bi se pokazalo koliko je bitan za određeno izdanje softvera. Određivanje prioriteta se kolektivno radi.

Nedvosmislenost Svi moraju da razumeju značenje zahteva na isti način, to se proverava inspekcijom.

Proverljivost Zahtevi moraju biti proverljivi i to se utvrđuje testiranjem. Zato treba testere uključiti u proces utvrđivanja zahteva.

KARAKTERISTIKA KOLEKCIJA ZAHTEVA

Kompletnost, konzistentnost, promenljivost i sledljivost su karakteristike skupa zahteva izdanja.

Nije dovoljno imati odlične iskaze o zahtevima. Važno je i da skup svih zahteva koji su planirani za određeno izdanje softvera ili iteraciju ima dobre karakteristike, koje se ovde navode:

Kompletnost (Complete**)** Ne bi trebalo da nedostaje nijedan zahtev ili važna informacija. Njih je teško otkriti, jer se ne pominju, a često se implicitno podrazumevaju. Treba otkriti da neki zahtev nije eksplicitno naveden, i treba ga onda dodati.

Konzistentnost (Consistent) Konsistentni zahtevi su oni koji nisu u sukobu sa drugima zahtevima istog tipa, ili sa zahtevima poslovanja u širem smislu, zahtevima korisnika, ili sa sistemskim zahtevima. Ako ne otklonite takve konfliktne situacije, programeri će kasnije te probleme morati da rešavaju. Zato je važno da znate izvor svakog zahteva da bi sa izvorom otklonili problem.

Promenljivost (Modifiable) Neophodno je da zapisujete i održavate istoriju promena svakog zahteva. Morate znati veze i zavisnosti među zahtevima da bi mogli da ih sve promenite, kada menjate jedan od njih. Promenljivost znači da je svaki zahtev jedinstveno obeležen i nezavisno iskazan, tako da možete nedvosmisleno tražiti i naći. Zato, izbegnite redudantnost podataka, tj. da se isti zahtev navodi na različitim mestima. Na drugim mestima se može koristiti samo referenca na zahtev koji se definiše samo na jednom mestu.

Sledljivost (Traceable) Sledljiv zahtev može da se poveže sa svojim izvorima, ali i sa zahtevima koji su iz njega izvedeni, sa elementima projektnog rešenja, sa kodom koji ga primenjuje, i sa testom koji potvrđuje njegovu primenu. Ne morate da definišete sve te veze za praćenje da bi omogućili da zahtev bude sledljiv. Sledljiv zahtev se jedinstveno obeležavaju sa trajno memorisanim identifikatorom. Oni se pišu na strukturisan način, sa potrebnim nivoom detaljnosti, a ne u vidu dugačkih paragrafa. Izbegnite navođenje više zahteva zajedno u okviru jednog iskaza, jer pojedini iskazi mogu da vide do različitih komponenti razvoja, a njihovo grupisanje to sprečava (jer su u "istom košu").

U praksi, nikada nećete imati slučaj da svi zahtevi imaju sve ove karakteristike. Ako o njima vodite računa, imaćete bolje definisane zahteve, te samim tim, i bolji softver.

Poglavlje 5

Preporuke za pisanje zahteva

PISANJE ZAHTEVA IZ PERSPEKTIVE SISTEMA ILI KORISNIKA

Pri pisanju zahteva, možete ga pisati tako da kažete šta sistem treba da uradi (perspektiva sistema) ili šta korisnik treba da uradi (perspektiva korisnika).

Ne postoji formalan način pisanja odličnih zahteva. Do toga se dolazi iskustvom i praćenjem reakcija korisnika vaših specifikacija zahteva. Zato su recenzije bitne. Razmenjujte specifikacije zahteva sa vašim kolegama, analitičarima. Učićete jedni od drugih. Poboljšaćete vašu kolektivnu sposobnost pisanja dobrih zahteva i otkrićete greške i mogućnosti za njihovo otklanjanje u ranim fazama projekta. Pri pisanju zahteva, imate dva cilja:

- Svako ko čita zahtev, treba da ga razume ga na isti način,
- Razumevanje svakog čitaoca nekog zahteva je isto sa autorovim razumevanjem i namerom u vezi zahteva.

Perspektiva sistema ili korisnika

Pri pisanju zahteva, možete ga pisati tako da kažete šta sistem treba d a uradi (perspektiva sistema) ili šta korisnik treba da uradi (perspektiva korisnika). Izaberite onu koja najjasnije definiše određeni zahtev. Pišite zahteve na konsistentan način: "Sistem treba..." ili "Korisnik treba...". Specificirajte akciju pokretača i uslove za to, Ovde se daje opšti uzorak za pisanje zahteva:
[**opcioni uslov] [opcioni događaj pokretanja] [očekivan odgovor sistema]**
sistem treba [odgovor sistema]
Evo primera jednog funkcionalnog zahteva:
Ako je zahtevana hemikalija pronađena u skladištu hemikalija, sistem treba da prikaže listu svih kontejnera sa hemikalijom koji se trenutno nalaze u skladištu.
Ovaj zahtev sadrži preduslov, ali nem i pokretač akcije (događaj). Da bi se izbeglo ponavljanje, može se i izostaviti fraza, "sistem treba...". Trebalo bi da navedete akciju opkretača ili uslova koji dovodi do reakcije sistema na opisani način.
Ponekad, jasnije je ako se zahtev napiše iz perspektive korisnika, uz korišćenje aktivnih glagola, jer čini traženu akciju jasnijom.

[klasa korisnika ili ime aktera] treba da [uradi nešto] [nekom objektu] [uslovi prihvatanja, vreme odgovora, ili iskaz kvaliteta]

Evo primera:

Hemičar treba da ponovo zahteva hemikalije koje ja zahtevao u prošlosti pregledajući i menjajući detalje u ranijim zahtevima.
Umesto "korisnik", ovde se navodi konkretna klasa korisnika "Hemičar". To je dobro, jer treba da zahtev uvek kada je moguće, bude što konkretniji.

STIL PISANJA

Prvo navedite iskaz o potrebi ili o funkcionalnosti, a onda detalje podrške, tj.razloge, izvor, prioritet, i druge atributie zahteva. Važno je pisati jednostavno, jasno i koncizno.

Prvo navedite iskaz o potrebi ili o funkcionalnosti, a onda detalje podrške *razloge, izvor, prioritet, i druge atribute zahteva.

Ne mešajte pasivno i aktivno pisanje. Konsistentno primenjujte jedno od njih. Ne koristite više izraza za isti koncept. Važno je pisati jednostavno i razumljivo. Ovde se daju preporuke za pisanje jasnih zahteva:

Jasno i koncizno: Pišite zahteve sa kompletnom rečenicom, sa odgovarajućom gramatikom. Neka rečenice budu kratke i direktne. Izbegnite žargone i pišite jezikom domena. Dajte objašnjenje termina u rečniku termina. Pored jasnoće, važna je (mada manje) i konciznost. Uvek se pitajte šta će korisnik raditi sa informacijom koju mu dajete. Ako to nije jasno, onda je izbacite. Budite precizni i konkretni.

Ključna reč "treba": Ona označava sposobnost sistema koju želite. Ne mešajte termine u zahtevima. Ponavljajte istu ključnu reč.

Aktivni govor: Pišite aktivnim rečenicama tako da bude jasno šta entitet treba da uradi. Umesto da pišete šta se nekom treba da uradi (pasivni način), pišite ko i šta treba da uradi (aktivni način pisanja). Evo primera:

Kada odeljenje isporuke potvrdi da je pošiljka sa poboljšanim proizvodom poslata, sistem će promeniti serijski broj proizvoda u ugovoru sa novim serijskim brojem.

Pojedinačni zahtevi: Umesto jednog složenog sa više zahteva u sebi, pišite jednostavne, posebne i pojedinačne zahteve. U takvim slučajevima, napišite više pojedinačnih zahteva.

NIVO DETALJA

Zahteve treba pisati na onom nivou detalja koliko je minimalno potrebno za njihovu primenu

Zahteve treba pisati na onom nivou detalja koliko je minimalno potrebno inženjerima razvoja da na osnovu toga, ih pravilno primene.

Odgovarajući detalji: Uopštene zahteve treba razbiti na više jednostavnih na potrebnom nivou detaljnosti informacija. Koliko detaljno? Onoliko koliko je minimalno potrebno da

minimizirate rizik od različitih razumevanja zahteva. Treba da uključite više detalja u sledećim slučajevima:

- Radi se posao za odličnog kupca.
- Razvoj ili testiranje biće dato u outsourcing, tj, dato kooperantu.
- Članovi projektnog tima su na različitim lokacijama.
- Testiranje sistema biće zasnovano na zahtevima.
- Potrebne su tačne procene.
- Potrebno je trasirati zahteve, tj. imati veze ka njihovim izvorima i ka izvedenim zahtevima.

Može se dati manje detalja kada:

- Kada radite interni projekt za vašu kompaniju.
- Kupci su vrlo uključeni u projekat.
- Programeri i inženjeri imaju puno iskustva.
- Dozvoljeni su presedani, kao u slučaju zamene prethodne aplikacije.
- Koristi se paketno rešenje (softver)

Konsistentna granularnost: Pisci zahteva se stalno pitaju sa kojom granularnošću da pišu funkcionalne zahteve? Ne morate sve zahteve pisati na istom nivou detalja. U delovima koji nose veće rizike možete više u dubinu (veći novo detalja) .Ipak, zgodno je ceo skup funkcionalnih zahteva pisati sa konzistentnim nivoom granularnosti. Postoji preporuka da granularnost bude takva da zahtev može da se verifikuje jednim testom.

TEHNIKE PRESTAVLJANJA ZAHTEVA

Izaberite najefektniji način predstavljanja zahteva. Izbegavajte dvosmislenosti.

Predstavljanje zahteva:

Koristite najefektniji oblik predstavljanja svakog zahteva u skladu sa odgovarajućim auditorijumom. Mogu se koristiti liste, tabele, vizuelni modeli analize, dijagrami, matematičke formule, zvučni i video zapisi. Sve ovo ne može biti zamena i za primenu tekstualnog opisa zahteva. ali služi kao odličan dodatak koji povećava čitaocu razumljivost zahteva.

Kada imamo grupu sličnih zahteva, koji se razlikuju po nekoj činjenici ili podatku, onda je najbolje koristiti tabelarni prikaz takvih zahteva. Ako neka kombinacija ne dovodi do nekog funkcionalnog zahteva, onda je bolje je zadržati u tabeli, ali označiti je "nije primenljivo", umesto da je izostavite. To bi moglo da izazove nedoumicu kod čitaoca, i da se pitaju da li je nekom greškom ta kombinacija izostavljena.

Izbegavanje dvosmislenosti

Recenzija zahteva može najbolje da ukaže na nejasne ili na dvosmislene zahteve. Ovde se navode nekoliko izvora dvosmislenosti u zahtevima.

Nejasne reči: Koristite konzistentno pojmove i definišite ih u vašem rečniku projekta. Ne mešajte sinonime. Usvojite samo jedan naziv nekog pojma (koncepta) za pisanje zahteva. Izbegnite korišćenje dvosmislenih ili nepreciznih termina i dvosmisleni jezik.

Oblik A/B: Mnogi zahtevi sadrže razlomke tipa A/B u kojima se nalaze sinonimi illi neki drugi termini. Često su oni dvosmisleni. Autor možda nije bio siguran koji od dva termina da koristi, pa je na ovaj način naveo oba. Čitalac može svaki od njih da razume različito, i to onda dovodi do dvosmislenosti,

Granične vrednosti: Ako granične vrednosti nisu precizno i eksplicitno navedene, onda one unose dvosmislenost u zahteve i prave nejasnoće. Na primer, "Student dobija 8, ako ima 80 do 90 poena, a 10, ako ima od 90 do 100": Nije jasno koju ocenu student dobija ako ima 90 poena.

Negativni zahtevi: Zahtev govori šta sistem ne treba da radi. To može da unese nejasnoće. Napišite takve zahteve u pozitivnom smislu, koji bi jasno ukazao na željeno ponašanje sistema,

IZBEGAVANJE NEKOMPLETNOSTI

Postoje nekoliko tehnika da pronađete nedostajuće zahteve: Simetrija, složena logika i nedostajući izuzeci.

Postoje nekoliko tehnika da pronađete nedostajuće zahteve. Usmerite se više na zadatke korisnika umesto na svojstva sistem da bi pronašli nezahtevanu, a potrebnu funkcionalnost.

Simetrija: Simetrične operacije su česti izvor nedostajućih zahteva. Na primer: "*Korisnik mora da memoriše ugovor u bilo kom trenutku tokom manuelnog postavljanja ugovora*". Ne govori se o tome da korisnik mora da ima mogućnost da izvadi nekompletan zahtev koji je memorisan, da bi nastavio rad na njemu. Programer mora to da zna.

Složena logika: Obično zahtevi u kojima se koriste složeni logički iskazi ostavljaju neke kombinacije bez odgovora, tj. nisu kompletne. Programer kada analizira takve zahteve dođe u nedoumicu kako da ih tumači.

Nedostajući izuzeci: Svaki zahtev ima deo koji govori šta sistem treba da radi, ali bi trebalo da ima i deo šta treba da radi u slučaju izuzetnih situacija (tzv. izuzeci). Ako oni nedostaju u opisu zahteva, može u praksi da dođe do nepredviđene situacije, i sistem se blokira. Programer to može da uoči, ali mora da stvar raspravi sa autorom zahteva ili sa analitičarem.

VIDEO 24 - CHARACTERISTICS OF EXCELLENT REQUIREMENTS - WIEGERS (VIDEO)

Trajanje: 5:14 minuta

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO 25 - TIPS FOR WRITING CLEAR REQUIREMENTS - WIEGERS (VIDEO)

Trajanje: 4:02 minuta

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO 26 - TIPS FOR WRITING CLEAR REQUIREMENTS - 2 - WIEGERS (VIDEO)

Trajanje: 5 minuta

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO 27 - WORDS TO AVOID IN REQUIREMENTS - WIEGERS (VIDEO)

Trajanje: 4:31 minuta

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 6

Vežba

PISANJE I IZRAŽAVANJE ZAHTEVA

Primeri izražavanja zahteva iz perspektive korisnika ili perspektive sistema

Postoji više tehnika i stilova pisanja zahteva. Zahtevi se, takođe, mogu izražavati na različitim nivoima detaljnosti. Specifični po tom pitanju su projekti razvoja softvera na kojima se primenjuju agilne metode razvoja.

Dve perspektive koje se obično primenjuju u pisanju zahtev su perspektiva korisnika ili perspektiva sistema. Tradicionalno, zahtevi se uglavnom specificiraju tako da izraze ono što sistem treba da radi. To ih čini jasnim i nedvosmislenim. Noviji pristupi inženjeringu zahteva preporučuju izražavanje zahteva iz perspektive korisnika, posebno što što se od korisnika očekuje da budu uključeni u proces razvoj. Da bi mogli da pomognu, oni najpre moraju da razumeju zahteve, a najjednostavniji način za to je postaviti zahteve iz njihove perspektive.

U nastavku vežbe dati su primeri pisanja zahteva, primenom pomenutih perspektiva, kao i primenom dveju različitih notacija.

Perspektiva sistema

Sistem treba da omogući izmenu podataka o zaposlenju registrovanog zaposlenog osoblja klinike.

Ako pacijent sa unetim JMBG brojem postoji u sistemu, sistem treba da prikaže sadržaj njegovog elektronski karton.

Perspektiva korisnika

Administrativni radnik treba da ima mogućnost izmene podataka o zaposlenju registrovanog zaposlenog osoblja klinike.
Medicinskom radniku treba da bude prikazan sadržaj elektronskog kartona pacijenta, ukoliko pacijent sa unetim JMBG brojem postoji u sistemu.

PISANJE ZAHTEVA

Primeri pisanja zahteva primenom konvencije rednih brojeva i hijerarhijskih tekstualnih oznaka

Primena rednih brojeva

REQ-80: Za sve izveštaje koji se generišu iz sistema, treba da postoji mogućnost generisanja u formi Word, PDF ili Excel fajla.

REQ-85: Sistem treba da omogući menadžeru generisanje izveštaja o rashodima napravljenim u određenom periodu.

REQ-86: Izveštaj o rashodima u određenom periodu treba da ima kao obavezne ulazne parametre početni datum za period i krajnji datum za period. Opcioni ulazni parametar je odeljenje klinike za koje se generiše izveštaj.
REQ-87: Ako se izveštaj o rashodima u određenom periodu generiše za sva odeljenja klinike, spisak rashoda treba da bude sortiran prema odeljenjima. Ako se izveštaj o rashodima u određenom periodu generiše za odabrano odeljenje, on treba da sadrži samo spisak rashoda za odabrano odeljenje. Za svaki rashod na spisku treba da bude naznačeno da li je rashod izmiren ili nije.
Primena hijerarhijskih tekstualnih oznaka

Izveštaji: Generisanje izveštaja iz sistema

Izveštaji.Format: Za sve izveštaje koji se generišu iz sistema, treba da postoji mogućnost generisanja u formi Word, PDF ili Excel fajla.

Izveštaji.RashodiZaPeriod: Sistem treba da omogući menadžeru generisanje izveštaja o rashodima napravljenim u određenom periodu.

Izveštaji.RashodiZaPeriod.ObavezniParametri: Izveštaj o rashodima za period treba da ima kao obavezne ulazne parametre početni datum za period i krajnji datum za period.

Izveštaji.RashodiZaPeriod.OpcioniParametri: Izveštaj o rashodima za period treba da ima kao opcioni ulazni parametar odeljenje klinike za koje se generiše izveštaj.
Izveštaji.RashodiZaPeriod.Soritanje: Ako se izveštaj generiše za sva odeljenja klinike, spisak rashoda treba da bude sortiran prema odeljenjima. Ako se izveštaj generiše za odabrano odeljenje, treba da sadrži samo spisak rashoda za odabrano odeljenje. Za svaki rashod na spisku treba da bude naznačeno da li je izmiren ili nije.

PITANJA ZA DISKUSIJU

Tekst pitanja za diskusiju

PITANJE 1.

Pogledajte sadržaj templejta SRS dokumenta. Njegova namena nije u tome da dobijete na broju stranica i popunite baš svako poglavlje, već da osigurava da ćete sakupiti potrebne informacije za uspešan projekat. Taj šablon je koristan kao podsetnik i primer je dobre prakse dokumentovanja zahteva. Koja poglavlja nikada ne ispuštate iz vida kada utvrđujete zahteve? S druge strane, koja su to poglavlja koja zanemarujete i smatrate da bi trebalo da im posvetite više pažnje da biste potpunije dokumentovali zahteve? (10 min)

PITANJE 2.

Koju ćete konvenciju za označavanje pojedinačnih zahteva datih u tekstu predavanja odlučiti da primenjujete? Zbog čega? (5 min)

ZADACI ZA VEŽBU

Tekst zadataka za vežbu

ZADATAK 1.

Dopuniti zahteve za sistem privatne klinike, poštujući pravila data u primerima. Napišite zahteve koji se odnose na slučajeve korišćenja u kojima je akter Lekar. (15 min)

ZADATAK 2.

Primenom odabrane konvencije za označavanje zahteva, popišite zahteve za ISUM-ov model E-student za svaki pojedinačan slučaj korišćenja koji ste identifikovali. Povedite računa o tome da nemate sukobljene, nekonzistentne ili nedovoljno jasne zahteve. (20 min)

ZADATAK 3.

Napisane zahteva za E-studenta, preformulisati tako da budu izraženi potpuno korišćenjem samo jedne od ponuđenih perspektiva (korisnik ili sistem). (20 min)

Poglavlje 7

Domaći zadatak

DOMAĆI ZADATAK 8

Tekst domaćeg zadatka

Prođite kroz opisan slučaj upotrebe, koje ste napisali u šestoj nedelji nastave za sistem dodeljen za DZ01.

Na početku rada navedite koju ćete konvenciju označavanja zahteva primenjivati u radu i zbog čega. Opredelite se za izražavanje zahteva iz perspektive sistema ili perspektive korisnika. Takođe na početku obrazložite zbog čega ste napravili takav izbor.
Identifikujte potrebne funkcionalne zahteve iz prethodno opisanih glavnih tokova, iz preduslova, postuslova, poslovnih pravila i poslovnih zahteva. Specificirajte identifikovane funkcionalne zahteve. Vodite računa da pravilno označite sve funkcionalne zahteve, tako da ih je moguće lako pratiti. Proverite da li svaki od Vaših zahteva ima karakteristike odličnih zahteva.
Napomene:

Zadatak se rešava opisno i šalje kao .docx fajl.

Rešenje zadatka pošaljite na mejl adresu predmetnog asistenta. Rok za izradu je definisan Plan i programom predmeta.

Poglavlje 8

Projektni zadatak

ZADATAK ZA RAD NA PROJEKTU

Tekst zadatka za rad na projektu

Prođite kroz sve slučajeve upotrebe, koje ste za potrebe projekta napisali nakon šeste nedelji nastave. Pokušajte da dobijete potrebne funkcionalne zahteve na svakom koraku, iz preduslova, postuslova, poslovnih pravila i drugih zahteva. Specificirajte identifikovane funkcionalne zahteve (**Poglavlje 3** – **Funkcije sistema**). Preporučeno je da pratite formu slučajeva korišćenja ili eventualno neku sličnu koja je predložena u uzorku SRS dokumenta. Vodite računa da pravilno označite sve funkcionalne zahteve, tako da ih je moguće lako pratiti. Proverite da li svaki od Vaših zahteva ima karakteristike odličnih zahteva.
Popunite i druge odeljke SRS dokumenta koje ste u mogućnosti nakon do sada pređenih materijala, na primer **Poglavlje 1 – Uvod** i **Poglavlje 2 – Opšti opis**. Dokument poslovnih zahteva i predlog projekta koji ste slali asistentu vam umnogome može pomoći da uradite ovaj deo. Proverite pažljivo svoj rad.

Poglavlje 9

Zaključak

ZAKLJUČAK

1. Progresivno poboljšanje detalja je ključni princip efektivnog razvoja zahteva.
2. Specifikacija softverskih zahteva sadrži funkcije i sposobnosti koje softverski sistem mora da obezbedi, njegove karakteristike, i ograničenja koja mora da poštuje.
3. Svaki zahtev treba da ima jedinstveni identifikator. On se koristi kod svakog pozivanja na odgovarajući zahtev.
4. Hijerarhijski tekstualni tagovi su su strukturisani, ukazuju na značenje i ne menjaju se pri dodavanju, brisanju ili pomeranju drugih zahteva
5. Postoje različiti uzorci SRS dokumenta, jer postoje i različite veličine projekata razvoj softvera. Na vama je da izaberete onaj koji najviše odgovara vašoj organizaciji i projektu.
6. Pri agilnom razvoju softvera, umesto funkcionalnih zahteva, najčešće se razvoj upravlja prema postavljenim testovima prihvatanja nastalim iz korisničkih priča.
7. Kompletnost, tačnost, ostvarljivost, potreba, usklađenost sa prioritetima, nedvosmislenost, proverljivost - to su karakteristike dobro definisanih zahteva.
8. Kompletnost, konzistentnost, promenljivost i sledljivost su karakteristike skupa zahteva izdanja.
9. Pri pisanju zahteva, možete ga pisati tako da kažete šta sistem treba da uradi (perspektiva sistema) ili šta korisnik treba da uradi (perspektiva korisnika).
10. Stil pisanja - prvo navedite iskaz o potrebi ili o funkcionalnosti, a onda detalje podrške *razloge, izvor, prioritet, i druge atribute zahteva. Važno je pisati jednostavno i razumljivo.
11. Nivo detalja: Zahteve treba pisati na onom nivou detalja koliko je minimalno potrebno za njihovu primenu
12. Izaberite najefektniji način predstavljanja zahteva. Izbegavajte dvosmislenosti.
13. Postoje nekoliko tehnika da pronađete nedostajuće zahteve: Simetrija, složena logika i nedostajući izuzeci.

REFERENCE

Nastavi materijal pripremljen za studente se pravi s namerom da im omogući brži i skraćeni uvid u program lekcije, a na bazi jedne ili više referentnih udžbenika i drugih izvora . Nastavni materijal nije zamena za ove udžbenike, koje treba koristiti ako student želi da se detaljnije upozna sa nastavnom materijom. Očekuje se od studenta da poseduje bar jedan od navedenih udžbenika u Planu i programu predmeta.

Ova lekcija je urađena na bazi teksta datom **u poglavljima 10 i 11** knjige: **Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013**. Za detaljnije proučavanje

i primere, studentima se preporučuje da pročitaju ovo poglavlje. Manji uticaj na sadržaj lekcije imaju ostale reference navedene u Planu i programu predmeta,