**Kratak pregled:** Sadržaj dobrog SRS dokumenta karakteriše formalan jezik i dosljednost u poštovanju standarda po kojem se dokument piše. Kvaliteti dobro napisanog SRS dokumenta ogledaju se u tome što znatno pojednostavljuje implementaciju sistema, jer daje detaljan opis svih zahjeva i slikovito prikazuje procese (pomoću dijagrama). Ovaj dokument daje opis sistema za pomoć na putu. Cilj je da se specifikuju zahjevi softvera koji se razvija.

**Ključne riječi:** dispečer, radnik, administrator, supervisor, specifikacija softvera, specifikacija zahtjeva

**Uvod**

Opisani su preporučeni pristupi za specifikaciju softverskih zahtjeva. Dokument je zasnovan na modelu u kojem je rezultat procesa specifikacije softverskih zahtjeva nedvosmislen I potpuni document specifikacije. Trebalo bi da pomogne:

1. Kupcima softvera da precizno opišu šta žele da dobiju;
2. Dobavljačima softvera da razumiju šta tačno korisnik želi;
3. Pojedincima da postignu sljedeće ciljeve:
4. Izraditi standardnu specifikaciju softvera (SRS) za svoje organizacije;
5. Definisati format i sadržaj specifičnih specifičnih specifikacija softverskih zahtjeva;
6. Razviti dodatne lokalne podrške kao što su kontrolna lista kvaliteta SRS ili priručnik za one koji pišu SRS dokument.

Klijentima, dobavljačima i drugim pojedincima, dobar SRS treba da pruži nekoliko specifičnih koristi, kao što su:

* Uspostaviti osnovu za sporazum između kupaca i dobavljača o tome šta softver treba da uradi. Kompletan opis funkcija koje treba da izvrši softver naveden u SRS-u će pomoći potencijalnim korisnicima da utvrde da li određeni softver zadovoljava njihove potrebe ili kako softver mora biti modifikovan kako bi se zadovoljile njihove potrebe.
* Smanjiti razvojni napor. Priprema SRS-a primorava razne zainteresovane grupe u organizaciji kupca da rigorozno razmotre sve zahtjeve prije početka dizajna i smanji naknadni redizajn i ponovno testiranje. Pažljiv pregled zahtjeva u SRS-u može otkriti propuste, nesporazume i nedosljednosti ranije u razvojnom ciklusu kada su ovi problem lakši za ispravljanje.
* Obezbijediti osnovu za procjenu troškova i rasporeda. Opis proizvoda koji se razvija kako je dat u SRS-u je realna osnova za procjenu troškova projekta i može se koristiti za dobijanje saglasnosti za ponude ili procjene cijena.
* Obezbijediti osnove za validaciju i verifikaciju. Organizacije mogu razviti svoje planove validacije i verifikacije mnogo produktivnije od dobrog SRS-a. U sklopu ugovora o razvoju, SRS obezbjeđuje osnovnu tačku prema kojoj se može mjeriti usklađenost.
* Omogućiti transfer. SRS olakšava prenos softvera na nove korisnike ili nove mašine. Korisnicima je lakše prenijeti softver u druge dijelove svoje organizacije, a dobavljačima je lakše prenijeti na nove kupce.
* Služi kao osnova za poboljšanje. Pošto SRS diskutuje o proizvodu, služi kao osnova za kasnije poboljšanje gotovog proizvoda. SRS će možda morati da se izmijeni, ali on pruža osnovu za kontinuiranu procjenu proizvodnje.

1. **Pregled**

Pregled je podijeljen na pet klauzula. Klauzula 1 objašnjava obim ovog dokumenta. Klauzula 2 navodi reference na druge standarde. Klauzula 3 daje definicije specifičnih izraza koji se koriste. Klauzula 4 daje osnovne informacije za pisanje dobrog SRS-a. Klauzula 5 razmatra svaki od bitnih dijelova SRS-a. Dokument takođe imam dva aneksa, jedan koji obezbjeđuje alternativne šablone formata i onaj koji daje smjernice za usklađenost sa IEEE/EIA 12207.1-1997.

* 1. **Obim**

Obim dokumenta ukratko opisuje sistem koji se razvija. Sistem je namijenjen za brzu i efikasnu pomoć na putu u slučaju manjih ili većih kvarova vozila na putu. Aplikacija sa strane dispečera, sa komunikacijom sa radnicima na terenu (dispečer šalje podatke o mjestu potrebe, radnik na terenu vraća izvještaj). Sistem se izvršava u realnom vremenu. Sistem prati dostupnost vozila (radnika). Korisnici imaju slojevit pristup (dispečer, terenski radnik, sistemski administrator, supervizor). Dispečer se bavi organizacijom poziva i intervencija. Sistemski administrator upravlja nalozima. Supervizor ima pristup izvještajima i radnicima.

1. **Preporuke**

Ova preporučena praksa će se koristiti zajedno sa sljedećim publikacijama:

ASTM E1340-96, Standard Guide for Rapid Prototyping of Computerized Systems.

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Plans. IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning.

IEEE Std 828-1998, IEEE Standard for Software Configuration Management Plans.

IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software.

IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reli-able Software.

IEEE Std 1002-1987 (Reaff 1992), IEEE Standard Taxonomy for Software Engineering Standards.

IEEE Std 1012-1998, IEEE Standard for Software Verification and Validation.

IEEE Std 1012a-1998, IEEE Standard for Software Verification and Validation: Content Map to IEEE/EIA 12207.1-1997.

IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions. IEEE Std 1028-1997, IEEE Standard for Software Reviews.

IEEE Std 1042-1987 (Reaff 1993), IEEE Guide to Software Configuration Management.

IEEE P1058/D2.1, Draft Standard for Software Project Management Plans, dated 5 August 1998.

IEEE Std 1058a-1998, IEEE Standard for Software Project Management Plans: Content Map to IEEE/EIA 12207.1-1997.

IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes.

IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements Specifications.

1. **Definicije**

Generalno, definicije pojmova koje se koriste u ovoj preporučenoj praksi odgovaraju zahtjevima IEEE Std 610.12-1990. Sljedeće definicije su ključni termini koji se koriste u ovoj preporučenoj praksi:

* 1. **dispečer:** Dispečer je osoba koja se bavi organizacijom poziva i intervencija.
  2. **radnik:** Radnik kreira izvještaje.
  3. **administrator:** Administrator upravlja nalozima.
  4. **supervizor:** Supervizor ima pristup izvještajima i radnicima.

1. **Razmatranja za pisanje dobrog SRS-a**

Ova klauzula pruža osnovne informacije koje treba uzeti u obzir prilikom pisanja SRS-a. Ovo uključuje sljedeće:

1. Prirodu SRS-a;
2. Okruženje SRS-a;
3. Karakteristike dobrog SRS-a;
4. Zajednička priprema SRS-a;
5. Evolucija SRS-a;
6. Izrada prototipa;
7. Dizajn ugradnje u SRS;
8. Uključivanje projektnih zahtjeva u SRS.
   1. **Priroda SRS-a**

SRS je specifikacija za određeni softverski proizvod, program ili skup programa koji obavljaju određene funkcije u određenom okruženju. SRS može pisati jedan ili više predstavnika isporučitelja, jedan ili više predstavnika kupaca, ili oboje. Podklauzula 4.4 preporučuje oboje.

Osnovna pitanja koja će pisac SRS-a adresirati su sljedeća:

1. *Funkcionalnost.* Šta softver treba da radi? Softver pruža usluge pomoći na putu. Detaljniji opis u tački 1.1.
2. *Spoljašnji interfejs.* Kako softver interaguje sa ljudima, hardverom sistema, drugim hardverom i drugim softverom? Softver interaguje sa ljudima putem mobilnog telefona. Hardver sistema su računari i mobilni uređaji. Nema sekundarnog softvera.
3. *Performanse.* Kolika je brzina, raspoloživost, vrijeme odziva, vrijeme oporavka različitih softverskih funkcija,...? Raspoloživost i brzina su sekvencijalni jer postoji samo po jedan član (dispečer, radnik, administrator, supervizor) koji obavljaju poslove jedan za drugim. Zbog ovoga, vrijeme odziva može biti relativno sporo. Za pojedine izuzetne slučajeve su implementirane posebne funkcionalnosti koje ih razrješavaju.
4. *Atributi.* Koja su razmatranja za prenosivost, ispravnost, održivost, bezbijednost i sl? Aplikacija je za korisnika prvenstveno namijenjena kao mobilna aplikacija, dok se sa serverske strane koriste računari. Kao što je rečeno pod tačkom d), neki izuzeci će biti dodatno riješeni. Sistem je održiv i bezbijedan (**NAKNADNO**).
5. *Ograničenja dizajna nametnuta implementaciji.* Postoje li zahtjevani standardi na snazi, programski jezik za implementaciju, pravila za integritet baze podataka, ograničenja resursa, radno okruženje,...? Za implementaciju se koristi programski jezik Java, pravila za integritet **NAKNADNO**, ograničenje resursa se ogleda u tome što postoji samo po jedan član (dispečer, radnik, administrator, supervizor). Radno okruženje **NAKNADNO**

Pisci SRS-a bi trebalo da izbjegavaju postavljanje zahtjeva za projektovanje ili projekat u SRS. Za preporučeni sadržaj SRS-a pogledajte klauzulu 5.

* 1. **Okruženje SRS-a**

Važno je razmotriti dio koji SRS igra u ukupnom projektnom planu, koji je definisan u IEEE Std 610.12-1990. Softver može u suštini sadržavati svu funkcionalnost projekta ili može biti dio većeg sistema. U posljednjem slučaju tipično će biti SRS koji će navesti interfejse između sistema i njegovog softverskog dijela i postaviti spoljašnje performanse i funkcionalnost na softverski dio. Naravno, SRS se onda treba složiti sa ovim sistemskim zahtjevima i proširiti ih.

IEEE Std 1074-1997 opisuje korake u životnom ciklusu softvera i primjenljive ulaze za svaki korak. Ostali standardi, kao što su navedeni u članu 2, odnose se na druge dijelove životnog ciklusa softvera i tako mogu dopunjavati softverske zahtjeve.

Pošto SRS ima specifičnu ulogu u procesu razvoja softvera, pisci SRS-a bi trebalo da budu pažljivi da ne prelaze granice te uloge. To znači da SRS:

1. Trebalo bi da pravilno definiše sve softverske zahtjeve. Zahtjev softvera može postojati zbog prirode zadatka koji se mora riješiti ili zbog posebne karakteristike projekta.
2. Ne treba opisati bilo koji dizajn ili detalje o implementaciji. Ovo bi trebali opisati u fazi projektovanja projekta.
3. Ne smije nametati dodatna ograničenja softveru. One su propisno navedene u drugim dokumentima, kao što je plan za osiguranje kvaliteta softvera.

Stoga, ispravno napisan SRS ograničava opseg validnog dizajna, ali ne specifikuje nikakav konkretan dizajn.

* 1. **Karakteristike dobrog SRS-a**

SRS treba da bude:

1. Ispravan;
2. Nedvosmislen;
3. Kompletan;
4. Dosljedan;
5. Rangiran po značaju i/ili stabilnosti;
6. Verifikovan;
7. Moguće modifikovan;
8. Sljedljiv

**4.3.1 Ispravan**

SRS je tačan ako i samo ako je svaki zahtjev koji je naveden u njemu onaj koji softver mora ispuniti. Ne postoji alat ili postupak koji obezbjeđuje ispravnost. SRS treba uporediti sa bilo kojom primjenjivom superiornom speciifikacijom, kao što je specifikacija sistema zahtjeva, sa ostalom projektnom dokumentacijom i sa drugim važećim standardima, kako bi se osiguralo da se slaže.Alternativno, korisnik može utvrditi da li SRS ispravno reaguje na stvarne potrebe. Sljedljivost čini ovaj postupak lakšim i manje sklonim na greške (vidi 4.3.8).

**4.3.1 Nedvosmislen**

SRS je nedvosmislen ako i samo ako svaki postavljeni zahtjev ima samo jedno tumačenje. Kao minimum, ovo zahtjeva da svaka karakteristika finalnog proizvoda bude opisana korištenjem jedinstvenog izraza. U slučajevima kada izraz koji se koristi u određenom kontekstu može imati višestruka značenja, izraz treba da bude uključen u pojmovnik gdje je njegovo značenje precizirano.

SRS je važan dio zahtjeva procesa životnog ciklusa softvera, i koristi se za projektovanje, implementaciju, monitoring projekata, verifikaciju i validaciju, kao i za obuku kako je opisano u IEEE Std 1074-1997. SRS bi trebalo da bude nedvosmislen kako za one koji ga stvaraju, tako i za one koji ga koriste. Međutim, ove grupe često nemaju istu pozadinu i stoga nemaju tendenciju da opisuju softverske zahtjeve na isti način. Reprezentacije koje poboljšavaju specifikaciju zahtjeva za programera mogu biti kontraproduktivne jer smanjuju razumijevanje za korisnika i obrnuto.

Podklauzule 4.3.2.1 do 4.3.2.3 preporučuju kako izjeći nejasnoće.

**4.3.2.1 Prirodne jezičke zamke**

Zahtjevi se često pišu na prirodnom jeziku (npr. na engleskom). Prirodni jezik je inherentno dvosmislen. Prirodni jezik SRS-a treba pregledati od strane nezavisne stranke da bi identifikovao dvosmislenu upotrebu jezika, tako da se može ispraviti.

**4.3.2.2 Jezici specifikacije zahtjeva**

Jedan od načina da se izbjegne dvosmislenost inherentnog prirodnog jezika jeste da se SRS napiše u određenim jeziku specifikacije zahtjeva. Njegovi jezički procesi automatski otkrivaju mnoge leksičke, sintaktičke i semantičke greške. Jedna mana u upotrebi takvih jezika je trajanje potrebnog vremena za njihovo učenje. Takođe, mnogi netehnički korisnici ih smatraju nerazumljivim. Štaviše, ovi jezici imaju tendenciju da budu bolji u izražavanju nekih tipova zahtjeva.

**4.3.2.3 Alati za reprezentaciju**

Generalno, metode i jezici zahtjeva i alati koji ih podržavaju, spadaju u tri opšte kategorije: objekat, proces i ponašanje. Objektno orijentisani pristupi organizuju zahtjeve u smislu objekata iz stvarnog svijeta, njihove atribute i usluge koje obavljaju ovi objekti. Pristupi zasnovani na procesima organizuju zahtjeve u hijerarhije funkcija koje komuniciraju putem tokova podataka. Oponašajući pristupi opisuju spoljno ponašanje sistema u smislu nekog apstraktnog pojma, matematičke funkcije ili mašine stanja. Stepen do kojeg takvi alati i metode mogu biti korisni za pripremu SRS-a zavisi od veličine i složenosti programa. Ovdje se ne pokušava opisati ili podržati neki određeni alat. Kada se koristi bilo koji od ovih pristupa, najbolje je zadržati opis prirodnog jezika. Na taj način, kupci koji nisu upoznati sa notacijama, i dalje mogu razumjeti SRS.

**4.3.3 Kompletan**

SRS je kompletan ako i samo ako uključuje sljedeće elemente:

1. Svi značajni zahtjevi, bilo da se odnose na funkcionalnost, performanse, ograničenja dizajna, atribute ili eksterne interfejse. Posebno treba tretirati sve spoljne uslove koje nametne specifikacija sistema.
2. Definisanje odgovora softvera na sve realizovane klase ulaznih podataka u svim situacijama realizovanih klasa. Imajte na umu da je važno navesti odgovore na tačne i netačne ulazne vrijednosti.
3. Labele i reference na sve podatke, tabele i dijagrame u SRS-u i definiciju svih izraza i mjernih jedinica.

**4.3.3.1 Upotreba TBD**

Svaki SRS koji koristi frazu TBD (to be determined) nije kompletan SRS. Međutim, TBD je povremeno neophodan i treba ga propratiti sa:

1. Opisom uslova koji izazivaju TBD, tako da se situacija može riješiti;
2. Opisom onoga što se mora učiniti kako bi se eliminisao TBD, koji je odgovoran za njegovo eliminisanje, i kada se mora ukloniti.

**4.3.4 Konzistentan**

Konzistentnost se odnosi na unutrašnju konzistentnost. Ako se SRS ne slaže sa nekim dokumentom višeg nivoa, kao što je specifikacija sistema zahtjeva, onda to nije ispravno (pogledajte 4.3.1).

**4.3.4.1 Unutrašnja konzistentnost**

SRS je interno konzistentan ako i samo ako nijedan podskup pojedinačnih zahtjeva koji su poisaani u njemu nisu u sukobu.

Tri tipa vjerovatnih sukoba u SRS-u su sljedeći:

1. Navedene karakteristike objekata iz stvarnog svijeta mogu se sukobljavati. Npr.
2. Format izlaznog izvještaja može se opisati u jednom zahtjevu kao tabelarni, ali u drugom kao tekstualni.
3. Može biti logičan i vremenski konflikt između dvije navedene akcije. Npr.
4. Dva ili više zahtjeva mogu opisati isti objekat stvarnog svijeta, ali koriste različite termine za taj objekat. Na primjer, RADNIK je nadklasa za TERENSKI RADNIK. Upotreba standardne terminologije i definicija podstiče konzistentnost.

**4.3.5 Rangiranje po važnosti i/ili stabilnosti (DODATNO SA GRUPOM)**

SRS je rangiran za važnost i/ili stabilnost ako svaki zahtjev u njemu ima identifiikator koji ukazuje na važnost ili stabilnost tog određenog zahtjeva.

Obično svi zahtjevi koji se odnose na softverski proizvod nisu podjednako važni. Neki zahtjevi mogu biti od suštinske važnosti, posebno za kritične primjene, dok su drugi možda poželjni.

Svaki zahtjev u SRS-u treba identifikovati kako bi ove razlike bile jasne i eksplicitne. Identifikacija zahtjeva na ovaj način pomaže:

1. Da klijenti pažljivo razmotre svaki zahtjev, što često razjašnjava sve skrivene pretpostavke koje mogu imati.
2. Da programeri donose pravilne odluke o dizajnu i posvete odgovarajuće nivoe napora različitim dijelovima softverskog proizvoda.

**4.3.5.1 Stepen stabilnosti**

Jedan od načina identifikacije zahtjeva koristi dimenziju stabilnosti. Stabilnost se može izraziti u smislu broja očekivanih promjena bilo kog zahtjeva baziranog na iskustvu ili znanju o predstojećim događajima koji utiču na organizaciju, funkcije i osobe koje podržava softverski sistem.

**4.3.5.2 Stepen nužnosti (DODATNO SA GRUPOM)**

Drugi način za rangiranje zahtjeva je da se klase zahtjeva klasifikuju kao suštinske, uslovne i opcione.

1. *Suštinski.* **NAVESTI**
2. *Uslovni.* **NAVESTI**
3. *Neobavezni.* **NAVESTI**

**4.3.6 Moguće verifikovan (DODATNO SA GRUPOM)**

SRS se može provjeriti ako i samo ako je svaki zahtjev koji je naveden u njemu verifikovan. Zahtjev se može provjeriti ako i samo ako postoji određeni konačni postupak kojim osoba kojim osoba ili uređaj mogu provjeriti da li softverski proizvod ispunjava uslove. Generalno, svaki nejasan zahtjev nije verifikovan.

Neobjavljivi zahtjevi uključuju izjave kao što su “dobro radi”, “dobar interfejs” i “obično se dešavaju”. Ovi zahtjevi se ne mogu potvrditi jer je nemoguće definisati pojmove “dobro” ili obično. Izjava da ''program nikada ne ulazi u beskonačnu petlju'' je neprovjerljiva jer je testiranje ovog kvaliteta teoretski nemoguće.

Ako metoda ne može biti osmišljena da bi se utvrdilo da li softver ispunjava određeni zahtjev, onda taj zahtjev treba ukloniti ili revidirati.

**4.3.7 Moguće promjenljiv**

SRS se može modifikovati ako i samo ako su njegova struktura i stil takvi da svaka promjena zahtjeva može biti napravljena lako, potpuno i dosljedno, zadržavajući strukturu i stil. Prilagodljivost generalno zahtjeva od SRS-a da:

1. Ima koherentnu i jednostavnu organizaciju sa sadržajem, indeksom, i eksplicitnom unakrsnom referencom;
2. Ne bude suvišan (tj. isti zahtjev ne bi trebalo da se pojavljuje na više mjesta u SRS-u);
3. Izrazi svaki zahtjev odvojeno, umjesto da se miješa sa drugim zahtjevima.

Sama redudancija nije greška, ali može lako dovesti do grešaka. Redudantnost povremeno može pomoći da SRS postane čitljiviji, ali problem može nastati kada se redudantni dokument ažurira. Kad god je redudantnost neophodna, SRS treba uključiti eksplicitne unakrsne reference kako bi se ona mogla prilagoditi.

**4.3.8 Sljedljiv**

SRS se može pratiti ako je izbor svakog od njegovih zahtjeva jasan i ako olakšava upućivanje svakog zahtjeva u buduću dokumentaciju o razvoju ili poboljšanju. Preporučuju se sljedeće dvije vrste sljedljivosti:

1. sljedljivost unazad (tj. do prethodnih faza razvoja) . To zavisi od svakog zahtjeva koji eksplicitno upućuje na izvor u ranijim dokumentima.
2. sljedljivost unaprijed (tj. na sve dokumente koje je SRS uspostavio). To zavisi od svakog zahtjeva SRS-a koji ima jedinstveno ime ili referentni broj.

Napredna sljedljivost SRS-a je naročito važna kada softverski proizvod ulazi u fazu rada i održavanja. Kako su kod i dizajnerski dokumenti modifikovani, neophodno je biti u stanju utvrditi kompletan skup zahtjeva na koje mogu uticati te modifikacije.

* 1. **Zajednička priprema SRS-a**

Proces razvoja softvera trebalo bi da počne sa ugovorom o snabdjevaču i kupcima o tome šta kompletni softver mora da uradi. Ovaj sporazum u obliku SRS-a treba zajednički pripremiti. Ovo je važno jer obično ni kupac niti dobavljač nisu kvalifikovani da napišu dobar SRS sami.

1. Korisnici obično ne razumiju proces dizajniranja i razvoja softvera dovoljno dobro da napišu upotrebljiv SRS.
2. Dobavljači obično ne razumiju problem klijenta i oblast nastojanja dovoljno da preciziraju zahtjeve za zadovoljavajući sistem-

Zbog toga, kupac i dobavljač treba da rade zajedno da proizvedu dobro napisan i potpuno razumljiv SRS.

Posebna situacija postoji kada se sistem i njegov softver istovremeno definišu. Zatim funkcionalnost, interfejsi, performanse i drugi atributi i ograničenja softvera nisu unaprijed definisani, već su zajedničjki definisani, i predmet su pregovaranja i promjena. Ovo otežava ispunjavanje karakteristika navedenih u tački 4.3.Konkretno, SRS koji nije u skladu sa zahtjevima specifikacije matične kompanije nije ispravan.

Ova preporoučena praksa ne govori specifično o stilu, upotrebi jezika ili tehnikama dobrog pisanja. Međutim, veoma je važno da SRS bude dobro napisan. Opšte tehničke knjige pisanja mogu se koristiti za vođenje.

* 1. **Evolucija SRS-a (DODATNO S GRUPOM)**

SRS će možda morati da se razvija dok razvoj softvera napreduje. Možda je nemoguće navesti neke detalje u trenutku početka projekta. Moguće su dodatne izmjene.

Dva glavna pitanja u ovom procesu su sljedeća:

1. Zahtjeve treba precizirati temeljno i potpuno, čak i ako se evolucione revizije mogu predvidjeti kao neizbježne.
2. Treba započeti formalni proces promjene za identifikaciju, kontrolu, praćenje i prijavljivanje projektovanih promjena. Odobrene promjene u zahtjevima trebaju biti ugrađene u SRS tako da:
3. Obezbijede tačan i potpun revizorski trag promjena;
4. Dozvole pregled trenutnih i zamijenjenih dijelova SRS-a.
   1. **Izrada prototipa (DODATNO S GRUPOM)**

Izrada prototipa se često koristi tokom zahtjeva projekta. Postoje mnogi alati koji omogućavaju prototip, koji pokazuje neke karakteristike sistema, koji se kreiraju vrlo brzo i lako. Vidjeti i ASTM E1340-96.

Prototipovi su korisni iz sljedećih razloga:

1. Kupac može vjerovatnije da pogleda prototip, nego da pročita SRS. Dakle, prototip obezbjeđuje brzu povratnu informaciju.
2. Prototip prikazuje neadekvatne aspekte ponašanja sistema. Stoga, to proizvodi ne samo odgovore, nego i nova pitanja.
3. SRS zasnovan na prototipu ima tendenciju da se tokom razvoja razvijamanje promjena, čime se skraćuje vrijeme razvoja.

Prototip bi se trebao koristiti kao način za izuzimanje softverskih zahtjeva.

* 1. **Dizajn ugradnje u SRS**

Zahtjev određuje eksterno vidljivu funkciju ili atribut sistema. Dizajn opisuje posebnu podkomponentu sistema i/ili njegove interfejse sa drugim podkomponentama. Pisac SRS-a treba jasno razlikovati između identifikacije potrebnih ograničenja dizajna i projektovanja određenog dizajna.

SRS treba da se fokusira na usluge koje treba obaviti. Ne bi trebalo da specifikuje dizajnerske stavke kao što su:

1. Particionisanje softvera u module;
2. Dodjeljivanje funkcija modulima;
3. Opisivanje informacija ili kontrole između modula;
4. Izbor struktura podataka.

**4.7.1 Neophodni zahtjevi za projektovanje (DODATNO S GRUPOM)**

U posebnim slučajevima, neki zahtjevi mogu ozbiljno ograničiti dizajn. Na primjer, sigurnosni zahtjevi mogu direktno da se odražavaju na dizajn kao što je potreba da se:

1. Zadrže određene funkcije u odvojenim modulima;
2. Dozvoli samo ograničena komunikacija između nekih oblasti programa;
3. Provjeri integritet podataka za kritične varijable.

Primjeri validnih ograničenja dizajna su fizički zahtjevi, zahtjevi za performansama, standardi razvoja softvera i standardi za osiguranje kvaliteta softvera.

* 1. **Uključivanje zahtjeva projekta u SRS**

SRS treba da adresira softverski proizvod, a ne proces proizvodnje softvera.

Zahtjevi projekta ne bi trebali biti uključeni u SRS. Ovdje su uključene sljedeće stavke:

1. Troškovi;
2. Raspored isporuke;
3. Procedure izvještavanja;
4. Metode razvoja softvera;
5. Osiguranje kvaliteta;
6. Kriterijumi validacije i verifikacije;
7. Procedura prihvatanja.

Projektni zahtjevi su navedeni u drugim dokumentima, obično u programu za razvoj softvera, planu za osiguranje kvaliteta softvera ili izjave o radu.

**5. Dijelovi SRS-a**

Ova klauzula razmatra svaki od bithih dijelova SRS-a. Dok SRS ne mora da prati ovaj plan ili koristi imena koja su ovdje data za svoje dijelove, dobar SRS treba uključiti sve informacije o kojima se ovdje govori.

Sadržaj  
1. Uvod  
1.1 Svrha  
1.2 Opseg  
1.3 Definicije, akronimi iskraćenice  
1.4 Reference  
1.5 Pregled  
2. Opšti opis  
2.1 Perspektiva proizvoda  
2.2 Funkcije proizvoda  
2.3 Korisničke osobine  
2.4 Ograničenja  
2.5 Pretpostavke i zavisnosti  
3. Posebni zahtjevi (vidi 5.3.1 do 5.3.8 za objašnjenja mogućih specifičnih zahtjeva. Vidi takođe Dodatak A za nekoliko različitih načina organiziranja ovog odjeljka SRS-a.)  
Dodaci  
Indeks

**Pregled SRS prototipa**