



SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Osnove projektovanja softvera

Lekcija 01

PRIRUČNIK ZA STUDENTE

SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Lekcija 01

OSNOVE PROJEKTOVANJA SOFTVERA

- ✓ Osnove projektovanja softvera
- ✓ Poglavlje 1: Proces projektovanja
- ✓ Poglavlje 2: Uloge projektnih aktivnosti
- ✓ Poglavlje 3: Projektovanje kao proces rešavanja problema
- ✓ Poglavlje 4: Projektovanje u procesu razvoja softvera
- ✓ Poglavlje 5: Proces projektovanja softvera
- ✓ Poglavlje 6: Pokazna vežba - projektovanje softvera
- ✓ Poglavlje 7: Individualna vežba - Zadaci
- ✓ Poglavlje 8: Domaći zadatak br.1
- ✓ 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

Proces projektovanja softvera se može definisati kao proces rešavanja problema.

Primer važnosti projektovanja može se uvideti u primeru softvera sistema za podršku i upravljanje avionom. Veliki broj ljudi smatra neophodnim da se navedeni softver detaljno planira, projektuje i rigorozno testira, čak i ako nikada nisu putovali avionom. Takođe, isti princip projektovanja softvera treba da važi i za manje sisteme kao što je recimo program za pisanje teksta i tekstualnih dokumenata. Nijedan korisnik programa ne želi da mu se deo napisanog teksta u programu slučajno obriše ili jednostavno ne bude sačuvan iako se radi o programu koji nije kritičan iz pogleda sigurnosti kao softver avionskog sistema. Potrebno je napomenuti da verovatno nije isplativo primeniti iste tehnike projektovanja za program za obradu teksta i za softver za podršku i upravljanje avionom (koji je kritičan po pogledu bezbednosti njegovih korisnika) ali potreba za dobro projektovanim softverskim proizvodom i dalje postoji. Rezultat projektovanja treba da bude funkcionalnost softverskog proizvoda dovoljna da zadovolji potrebe njegovih korisnika.

Osnovni problem projektovanja je što su projektanti obavezni da koriste trenutne informacije kako bi predvideli buduće stanje koje neće doći osim ako njihova predviđanja nisu tačna. Konačno stanje projektovanja se mora prepostaviti od strane projektanata a projektanti moraju raditi unazad od prepostavljenog efekta sve do početka lanca događaja koji do efekta dovodi. (Design Methods: Seeds of Human Futures (Jones, 1970)).

Ova lekcija vam obezbeđuje sledeće ishode učenja:

- razumevanje procesa projektovanja softvera
- razumevanje i primenu procesa projektovanja softvera i projektnih aktivnosti
- upoznavanje sa procesom projektovanja softvera, modelom rešenja i kreiranjem incijalnog modela projektovanja softvera
- definisanje i razumevanje dugoročnog plana izmena softvera i procesa restrukturiranja softvera

UVODNI VIDEO LEKCIJE

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

▼ Poglavlje 1

Proces projektovanja

REŠAVANJE PROBLEMA U PROCESU PROJEKTOVANJA

Rešavanje problema u procesu projektovanja softvera može biti izvršeno primenom različitih pristupa.

Kao što se može primetiti iz navedenog puta, nauka se bavi proučavanjem stvari kakve jesu uz dve glavne aktivnosti posmatranje i eksperimentisanje. Rešavanje nekog identifikovanog problema koristeći put nauke obuhvata pristup rešavanju problema kroz sledeće faze:

- posmatranje karakteristika nekog fenomena
- merenje karakteristika fenomena
- razvijanja teorije koja objašnjava karakteristike fenomena
- merenje karakteristika fenomena
- potraživanje i objašnjavanje razvijene teorije fenomena

Put inženjeringu, za razliku od puta nauke, bavi se stvaranjem novih stvari kroz dve ključne aktivnosti u tom procesu:

- izgradnja
- evaluacija

Aktivnosti su usmerene na postizanje određenog definisanog efekta (cilja) i dešavaju se tako što se prepostavlja krajnji rezultat a zatim se prelazi na definisanje načina za dolazak do prepostavljenog krajnjeg rezultata.

FAZE UNUTAR PROCESA PROJEKTOVANJA

Faze u procesu projektovanja obuhvataju: zahtevanje rešenja, projektovanje modela rešenja, evaluaciju modela rešenja i elaboraciju modela rešenja.

Potrebno je naglasiti da naučni put i put inženjeringu nisu izolovani jedan od drugog. Međusobna povezanost dva puta omogućila je da proizvodi inženjeringu unaprede naučna znanja a da posmatranja i merenja naučnog puta unaprede proces razumevanja osnovnih koncepta projektovanja u cilju dobijanja novih načina za predviđanje efekata projektovanja. Softver se smatra artefaktom (proizvodom) kao i proizvodi drugih inženjerskih disciplina (mostovi, zgrade, statue, automobili). Proces projektovanja zavisi od oblasti u kojoj se softver koristi, može biti složeniji ili prostiji. Identifikovane faze u procesu projektovanja su:

- zahtevanje rešenja
- kreiranje modela rešenja
- izvršavanje evaluacije modela shodno početnim zahtevima
- izvršavanje elaboriranja modela u svrhu kreiranja detaljne specifikacije rešenja

Često se dešava da postoji nekoliko iteracija između navedenih faza unutar procesa projektovanja kao što je ponovno vraćanje na reviziju izbora projektovanja. Navedeni proces elaboriranja prvo bitno definisanog modela može otkriti nedostatke ili čak i potpunu neprilagođenost početnim zahtevima čime se dovodi u pitanje kompletan proces projektovanja.

POKAZNI PRIMER: ZAHTEVANJE REŠENJA

U okviru pokaznog primera prikazani su funkcionalni i nefunkcionalni zahtevi za softver koji treba da služi korisnicima kao asistent za različite režime ishrane i treninga.

	Title ID	Full Description	Code	Priority	Workload	Risk	Status
1	1.	Unos ljudnih podataka <ul style="list-style-type: none">• Sistem će omogućiti korisniku unos ljudnih podataka, kao što su ime, prezime, pol, datum rođenja, adresa, putne telefonske linije i visine	REQ_0001	Undefined	Undefined	Defined	Draft
2	2.	Unos težine i visine <ul style="list-style-type: none">• Nakon unetih ljudnih podataka sistem od korisnika traži da unesu svoju težinu i visinu kako bi se izračunao indeks telesne mase korisnika	REQ_0002	Undefined	Defined	Defined	Draft
3	3.	Prikaz indeksa telesne mase i preporučenih granica <ul style="list-style-type: none">• Nakon unesenog unutrašnjeg sistema korisniku ispisuje trenutni indeks telesne mase i granice za koju bi trebao da se hraniti• U slatko manji dio vremena korisnik se daje da bira način ishrane i kao i trening koji bi trebao da praktički koristiti u kontekstu ishrane	REQ_0004	Defined	Defined	Defined	Draft
4	4.	Prikaz preporučenih ishrana <ul style="list-style-type: none">• Sistem korisniku prikazuje preporučene ishrane na osnovu njegovog indeksa telesne mase	REQ_0005	Defined	Defined	Defined	Draft
5	5.	Odobrili ishrana <ul style="list-style-type: none">• Na osnovu preporučenih ishrana sistem od korisnika zahteva da odabere neki od ishrana	REQ_0006	Defined	Defined	Defined	Draft
6	6.	Unos dnevnih namirnica <ul style="list-style-type: none">• Na osnovu izbrane ishrane korisnik generisao za korisnika, pratići tu ishranu sistem unosi sve namirnice unute na dnevnom nivou bolje	REQ_0007	Defined	0	Defined	Draft
7	6.1	Izmjena namirnice <ul style="list-style-type: none">• Sistem korisniku omogućuje da izmjeni unete namirnice, kako kolичinu tako i samu namirnicu	REQ_0008	Defined	Defined	Defined	Draft
8	6.2	Brisanje namirnice <ul style="list-style-type: none">• Sistem korisniku omogućuje da izbriše unete namirnice	REQ_0009	Defined	Defined	Defined	Draft
9	7.	Održivi trening <ul style="list-style-type: none">• Na osnovu preporučenih ishrana sistem od korisnika zahteva da odabere neki od preporučenih treninga	REQ_0010	Defined	Defined	Defined	Draft
10	8.	Puštanje video materijala <ul style="list-style-type: none">• Sistem omogućuje korisniku da pušta video materijale o određenim vježbama, kako ih učiti, koliko se sružiti i koliko potražiti	REQ_0011	Defined	Defined	Defined	Draft
11	9.	Ocenjivanje treninga i ishrane <ul style="list-style-type: none">• Korisnik je u mogućnosti da oceni izbranu ishranu kao i trening na osnovu njegove interakcije sa sistemom	REQ_0015	Defined	Defined	Defined	Draft
12	10.	Unos komentara <ul style="list-style-type: none">• Korisnik je u mogućnosti da unosi komentare na određene vježbe tako i na ishrane	REQ_0012	Defined	Defined	Defined	Draft
13	11.	Bežeće zadatke vježbi <ul style="list-style-type: none">• Sistem zahteva od korisnika da vrši bežeće izvršenje izabranih vježbi na dnevnom nivou	REQ_0013	Defined	Defined	Defined	Draft

Slika 1.1 Primer specifikacije funkcionalnih zahteva za softver [Izvor: Nebojša Gavrlović]

Kao što je navedeno u fazama procesa projektovanja, prva faza podrazumeva zahtevanje rešenja identifikovanog problema kroz specifikaciju zahteva za softver. **Funkcionalni zahtevi** definisani od strane naručioca softvera služe kao ulazni parametar u proces projektovanja. Na slici 1 prikazan je primer specifikacije funkcionalnih zahteva za softver koji predstavlja ličnog asistenta za različite režime ishrane i treninga.

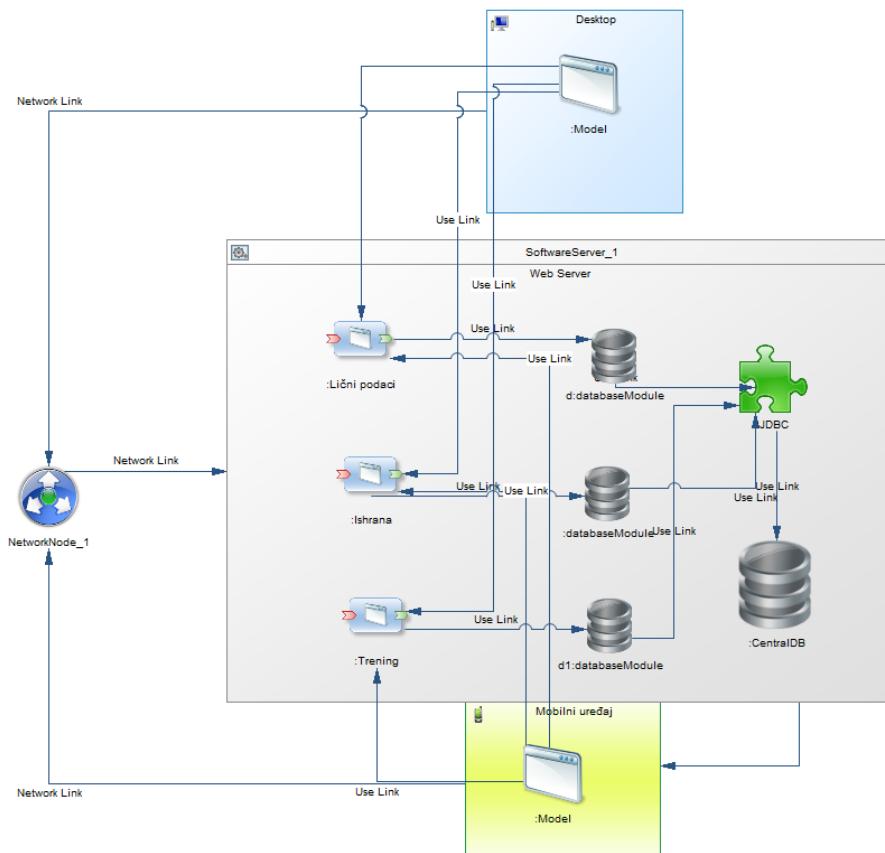
Na slici 2 prikazani su **nefunkcionalni zahtevi** za softver koji se takođe koriste u početnom procesu projektovanja softvera. Takođe, iz nefunkcionalnih zahteva moguće je definisati potencijalna ograničenja rešenja koje će biti rezultat procesa projektovanja.

1	1.	Performanse	REQ_0001	Undefined	Undefined	Draft
2	2.	Bezbednost	REQ_0002	Undefined	Undefined	Draft
3	3.	Sigurnost	REQ_0003	Undefined	Undefined	Draft
4	4.	Pouzdanost	REQ_0004	Undefined	Undefined	Draft
→	5.	Korisnički interfejs	REQ_0006	Undefined	Undefined	Draft

Slika 1.2 Primer specifikacije nefunkcionalnih zahteva za softver [Izvor: Nebojša Gavrlović]

POKAZNI PRIMER: MODEL REŠENJA

Primer modela rešenja za softver koji treba da služi korisnicima kao asistent za različite režime ishrane i treninga.



Slika 1.3 Primer modela rešenja [Izvor: Nebojša Gavrlović]

Na slici 3 dat je primer napravljenog modela rešenja na osnovu prethodno definisanih funkcionalnih i nefunkcionalnih zahteva. Model rešenja ne mora biti konačan ali treba da sadrži sve zahteve koji su dobijeni od strane naručioca softvera. U ovom slučaju predstavljen je dijagramom arhitekture, projektant softvera može na drugi način skicirati model rešenja ili koristiti drugi UML dijagram ukoliko je to potrebno za detaljniji prikaz modela rešenja.

Opis modela rešenja: Na slici koja predstavlja arhitekturu sistema nalazi se i mobilni uređaj i desktop računar (preko kojih korisnik pristupa aplikaciji). Aplikacija koristi MVC šablon arhitekture. Odvojena komponenta „Model“ pristupa servisima aplikacije koji se nalaze na serveru i oni takođe predstavljaju komponente za sebe a to su „Lični podaci“, „Ishrana“, „Trening“. Na osnovu tih servisa oni pojedinačno pristupaju modulima baze podataka koji nakon toga pristupaju glavnoj bazi podataka putem JDBC drafvera odnosno JDBC komponente. Servis „Lični podaci“ predstavljen je kao veza „Write/Read“ jer taj servis služi za čuvanje podataka o korisniku kao i za dodatno ažuriranje podataka. Servis „Ishrana“ predstavljen je kao veza „Read“ jer samo preuzima podatke iz baze podataka i prikazuje ih korisniku, isto kao i servis „Trening“.

POKAZNI PRIMER: EVALUACIJA MODELA SHODNO POČETNIM ZAHTEVIMA I ELABORIRANJE MODELA

Evaluacija modela vrši se analizom projektovanog modela rešenja i specifikacije korisničkih zahteva. Elaboriranje modela predstavlja detaljan opis svake komponente modela.

Evaluaciju modela rešenja moguće je izvršiti :

- Proverom da li su svi korisnički zahtevi ispunjeni na osnovu liste koja je definisane u okviru zahtevanog rešenja.

Nefunkcionalni zahtevi ne mogu biti provereni kroz model rešenja osim ukoliko se radi o ograničenjima budućeg sistema

Ukoliko model rešenja ispunjava sve funkcionalne zahteve prelazi se na proces elaboriranja rešenja. Jedan od mogućih načina elaboriranja modela (tekstualnim opisom) predstavljen je u okviru primera model rešenja.

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

▼ Poglavlje 2

Uloge projektnih aktivnosti

IDENTIFIKOVANJE PLANA PROJEKTOVANJA

Plan projektovanja omogućava prikaz potencijalnog rešenja i načina rešavanja identifikovanog problema.

Glavni zadatak projektanta je da odredi najbolje rešenje identifikovanog problema i da opiše način kako da se to rešenje primeni. Opis načina rešavanja problema formira plan koji može biti korišćen od strane implementatora sistema. Plan kao način izrade određenog *artefakta* se koristi od davnih vremena jer je malo verovatno da su građevine kao što su piramide izgrađene bez prethodno napravljenog plana i identifikovane arhitekture.

1. **Znanje stečeno naučnim istraživanjem** - kao primer uzeta je izgradnja mostova u devetnaestom veku gde su tadašnji inženjeri stvorili arhitektonski drugačije i izdržljivije mostove sve na osnovu prethodno proučenih i unapređenih materijala za izgradnju.
2. **Koncept ponovne upotrebe** - industrijska revolucija kao posledicu donela je ideju o standardizaciji komponenti. Postojanje *standardizovanih komponenti* omogućava ponovno korišćenje, različitu primenu i brži razvoj ideje uz niže troškove.

Primer studije slučaja gde se je izvršeno planiranja novog stambenog prostora za rezultat imao:

- plan koji informiše korisnike koji izvršavaju premeštanje nameštaja
- set planova koji informiše korisnike o dimenzijama komada nameštaja i kako se vrši spajanje određenih komada nameštaja

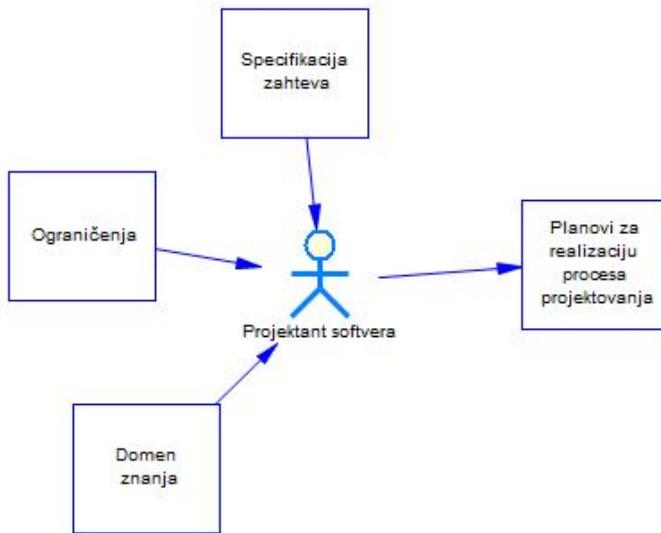
Prvi plan omogućava raspoređivanje nameštaja u okviru prostorija i zahteva određenu preciznost dok drugi set planova zahteva i veću preciznost dok se svi delovi jednog komada nameštaja spoje u krajnji izgled. *Plan koji proizilazi iz procesa projektovanja ne zavisi samo od prirode problema nego i od eventualne implementacije.*

Komunikacija sa osobom odgovornom za definisanje mogućeg rešenja predstavlja važan deo uloge projektanta kao i komunikacija sa izvorom specifikacije zahteva za softver u kojoj se često nalazi naručilac (kupac) softvera.

KANALI KOMUNIKACIJE PROJEKTANTA SOFTVERA

Projektant softvera kroz kanale komunikacije dobija sve potrebne informacije relevantne za proces projektovanja softvera.

U procesu projektovanja softvera potrebno je da projektant softvera ima određeni stepen znanja o domenu (domain knowledge) kao početni vid informacija koji je potreban za obavljanje bilo kog konkretnog zadatka u procesu projektovanja. Određene informacije, kao što je već navedeno, projektant softvera može dobiti iz specifikacije ali to često nije dovoljno. Na slici 1 je prikazan projektant softvera koji mora da odgovori na različite kanale komunikacije.



Slika 2.1 Kanali komunikacije projektanta softvera (Izvor: Budgen [2])

Takođe, može se desiti da unutar velikih projekata koji obuhvata veliki broj programera, planovi projektovanja obuhvataju širi spektar faktora nego u slučaju plana projektovanja na kome učestvuje jedan član. U slučaju velikih projekata i projektant softvera može biti programer.

PLANNOVI PROJEKTOVANJA SOFTVERA

Glavni zadatak projektanta softvera je da vrši kontrolu i modifikaciju plana projektovanja softvera shodno potrebnim izmenama u procesu projektovanja.

U slučaju velikih softverskih sistema planovi projektovanja sadrže:

- statičku strukturu sistema (sve potprograme predstavljene u detaljnoj hijerarhiji)
- sve objekte koji se koriste u sistemu
- algoritme koji se koriste
- pakete sistema (grupisane komponente ukoliko ih ima)
- interakcije između komponenti i prikaz svih uzročnih veza

Takođe, plan projektovanja može sadržati i način implementacije planiranih procesa, definisanje plana razvoja (koji potprogram, komponenta ili modul sistema ima prioritet u razvojnem procesu) kao i strategiju za eventualnu integraciju u kompletan sistem (ovaj plan je prilično sličan planu za sastavljanje komada nameštaja koji je opisan u studiji slučaja).

Potrebe projektnog proizvoda

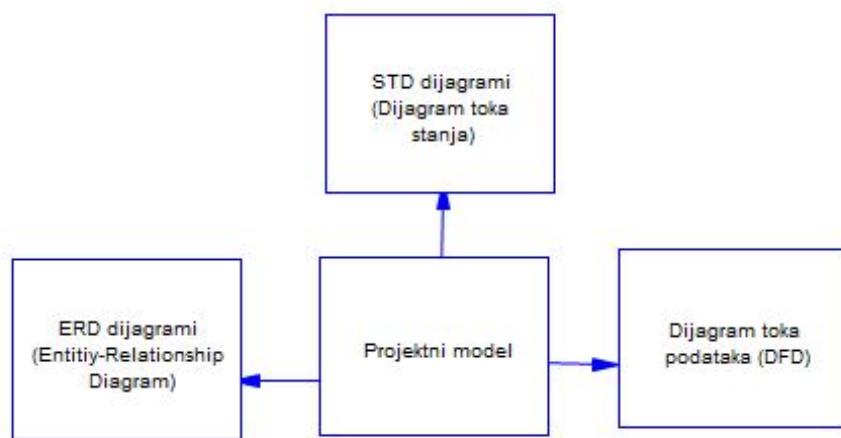
Projektanti softvera moraju da kreiraju i modifikuju planove koji određuju kako će konačni softverski proizvod biti sastavljen. Da bi kreirali detaljan plan projektovanja projektanti moraju upotrebiti različite "poglede" na sistem. Različiti pogledi na sistem omogućavaju da projektant softvera vidi sistem iz različitih uglova, pronađe najbolje rešenje za određene identifikovane probleme (primer iz studije slučaja sa visinom komada nameštaja i prozora i trodimenzionalnim pogledom na stambeni prostor). Pored različitih pogleda potrebno je imati i određenu toleranciju jer je potrebno softverske komponente uklopiti u jedan sistem.

Prilikom projektovanja softverskih sistema potrebno je izvršiti modelovanje softvera, opis ponašanja softvera, strukturu i funkcije koje izvršavaju određene zadatke.

PROJEKTNI MODEL

Izrada projektnog modela omogućava detaljan pregled identifikovanog problema i mogućih rešenja u toku projektovanja softvera.

Projektni model treba da obuhvata opise svakog dela sistema. Projektni model nastaje iz niza različitih oblika predstavljanja projektovanja softvera (upotrebom različitih pogleda na projektovani softver). **Složeniji sistem zahteva više različitih oblika predstavljanja (dijagrama koji opisuju stanja unutar softvera).** Na slici 2 prikazani su različiti pogledi na softver u procesu projektovanja.

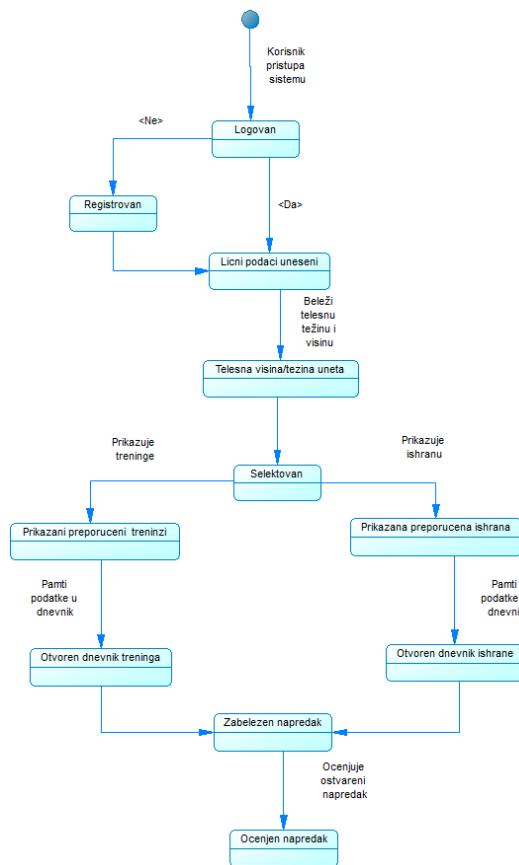


Slika 2.2 Primer različitih pogleda na softver u procesu projektovanja [Izvor: Budgen [2]]

POKAZNI PRIMER: PROJEKTNI MODEL

Projektni model može biti prikazan u formi dijagrama stanja.

Projektni model prikazan kroz dijagram stanja:



Slika 2.3 Projektni model softvera koji se koristi kao asistent za razlike režime ishrane i treninga [Izvor:
Nebojša Gavrlović]

▼ Poglavlje 3

Projektovanje kao proces rešavanja problema

PROCES REŠAVANJA PROBLEMA

Svrha projektovanja softvera je da jednostavno proizvede rešenje uočenog problema.

Problem se definiše kroz određenu specifikaciju zahteva a zadatak projektanta softvera je da omogući opis rešenja problema i način na koji će zahtevi biti ispunjeni. Proces projektovanja zahteva od projektanta softvera da vrši procenu različitih opcija za rešavanje problema i da donosi odluke shodno kriterijumima odlučivanja. Kriterijumi odlučivanja mogu biti složeni (uključujući razmenu faktora kao recimo brzine ili lakoće prilagođavanja) ili problemски orijentisani. Podrazumevani i krajnji zahtev odnosi se na to da sistem bude pogodan za svrhu za koju će biti korišćen. Koliko god softver bio dobro struktuiran, elegantan ili efikasan od korisnika će biti ocenjen na osnovu kriterijuma ispunjavanja svrhe za koju je napravljen.

Projektovanje softvera se može izvršiti kroz različite tipove alata, metoda i obrazaca. Takođe, izgradnjom modela (primenom različitih UML dijagrama) omogućen je prikaz i procenu ponašanja sistema. Kombinovanjem izgradnje modela i apstrakcije omogućena je izrada upravljačkih modela složenih sistema sa identifikovanjem njihovih kritičnih osobina. Apstrakcija omogućava uklanjanje detalja iz opisa problema uz zadržavanje suštinskih osobina svoje strukture. Dvodimenzionalni ili trodimenzionalni plan stambenog prostora koji je opisan u studiji slučaja predstavlja različite apstrakcije. U navedenim planovima nedostaju potpun opis celokupne strukture ali su zadržane osnovne informacije koje su bitne za dalje projektovanje.

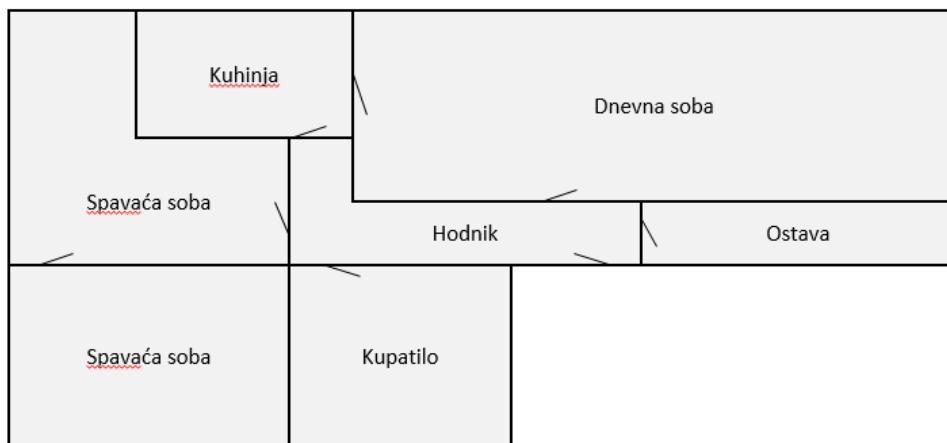
Projektant softvera treba da razume i razmišlja o sistemu na apstraktan način kroz događaje, entitete, objekte ili neke druge predmete koji su bitni za sistem. Proces projektovanja retko može biti konvergentan u smislu da bude u stanju da projektanta usmeri ka jednom poželjnном rešenju. Efikasna upotreba apstrakcije predstavlja ključnu veštinu koju svaki projektant softvera treba da nauči i primeni.

3.1 Studija slučaja - Proces projektovanja selidbe

PROJEKTOVANJE PROCESA SELIDBE I PLANIRANJE PROSTORA

Na primeru projektovanja procesa selidbe biće prikazan značaj primene faza projektovanja.

Selidba iz jednog stambenog prostora u drugi često zahteva veliko angažovanje u procesu planiranja budućeg rasporeda prostorija i nameštaja. Na ovom primeru se mogu prikazati primeri faza procesa projektovanja. Preporuka vlasniku u slučaju selidbe je da izvrši merenje dimenzija svih prostorija u novom stambenom prostoru kako bi dobio sve potrebne mere i počeo sa izradom plana prostora. Plan prostora obuhvata sve prostorije, vrata, prozore sa svim relevantnim dimenzijama. Sledeći korak podrazumeva merenje različitih predmeta nameštaja i ucrtavanje u plan budućeg stambenog prostora. Navedeni koraci formiraju početni model koji izgleda kao na slici 1:



Slika 3.1.1 Početni model novog stambenog prostora [Izvor: Budgen [2]]

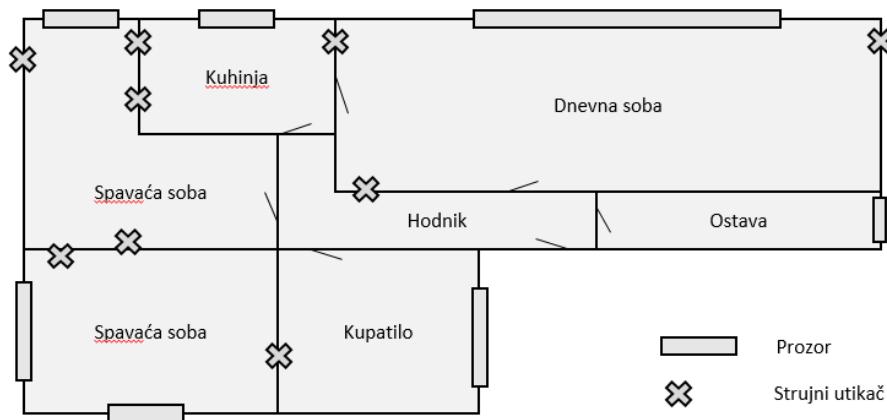
PRILAGOĐAVANJE PROCESA PROJEKTOVANJA SELIDBE

Primer procesa selidbe prikazuje i potrebu da se plan projektovanja prilagodi novim zahtevima koji su dobijeni na osnovu analize stambenog prostora.

Nakon početnog modela novog stambenog prostora prelazi se na identifikovanje najboljih pozicija za nameštaj. Primenom različitih strategija (osvetljenost prostorije, površina, pozicija) moguće je utvrditi koja soba ima najveći prioritet a shodno tome i nameštaj za prostoriju. Potrebno je obratiti pažnju i na ograničenja. Ograničenja mogu biti u vidu nameštaja (veliki broj ljudi ne bi postavio nameštaj za kuhinju u dnevnu sobu ali bi većina ljudi postavila fotelju koja predstavlja višak u dnevnoj sobi u svoju spavaću sobu). Takođe, potrebno je omogućiti neometan pristup strujnoj utičnici što vodi do narednog koraka u procesu projektovanja.

Pored prvobitnog plana novog stambenog prostora potrebno je obratiti pažnju i na trodimenzionalnu projekciju koja bi omogućila pregled prozora i vrata kako ne bi došlo do situacije da visoko pozicioniran nameštaj u prostoriji blokira prozor. Potrebno je izvršiti ustupke u planu između poželjnog i praktičnog. Slika 2 prikazuje način proširenja prvobitnog modela u svrhu dodatnog pojašnjenja. Odluke donešene u ovom procesu predstavljaju plan za korišćenje timu koji će izvršiti unošenje i postavljanje nameštaja u novi stambeni prostor. Takođe, biće izbegnuto čekanje i problem u razmeštanju nameštaja prilikom istovara. Ova jednostavna studija slučaja prikazuje nekoliko načina u kojima se put inženjerskog procesa projektovanja razlikuje od naučnog puta. Naučni put u ovoj studiji slučaja bi izvršio identifikovanje glavnih varijabli na osnovu koga bi proisteklo jedino rešenje.

Takvo rešenje, bez prilagođavanja mogućim "novim" situacijama u procesu projektovanja (kao što je višak nameštaja ili prilagođavanje rasporeda strujnoj utičnici), ne bi dovelo do optimalnog rešenja. Za opisanu studiju slučaja može biti mnogo mogućih rešenja i izbora, samim tim potrebno je izvršiti prilagođavanje trenutnim zahtevima što podrazumeva ponavljanje analize modela. Projektovanje softvera često je dosta kompleksnije nego projektovanja plana novog stambenog prostora. U studiji slučaja dimenzije prostorija su fiksne kao i dimenzije nameštaja pa je vrlo jednostavno odrediti gde i na koji način je moguće postaviti određeni komad nameštaja. U softveru to često nije slučaj, objekte softvera često nije lako izmeriti i prepoznati.



Slika 3.1.2 Dopunjeni model novog stambenog prostora [Izvor: Budgen [2]]

▼ Poglavlje 4

Projektovanje u procesu razvoja softvera

RAZLIČITI PROCESI RAZVOJA SOFTVERA

Velike softverske sisteme je moguće razviti na više različitih načina kako bi bile ispunjene određene potrebe ili zadovoljena potražnja određenog tržišta.

Softverski procesni model bavi se sledećim pitanjima:

1. Šta je potrebno uraditi sledeće?
2. Koliko dugo treba da nastavimo to da radimo?

Takođe, često se dešava da se projektant softvera u toku procesa projektovanja softvera više bavi identifikovanjem mesta gde je potrebno postaviti navedena pitanja nego analizirajući dobijene odgovore. Razvojni procesi se najčešće razvijaju kroz faze identifikovane u originalnoj strukturi modela vodopada koji se može primeniti u procesu razvoja softvera. Model vodopada, kao što je već poznato, sadrži niz zadataka koje je potrebno rešiti na određeni način. Primena modela vodopada i specifičnih zadataka biće posmatrana iz ugla projektovanja softvera. Potrebno je svakom zadatku dodeliti određenu važnost unutar procesa projektovanja. Zadatak koji je inicijalno klasifikovan u okviru faze "detaljan dizan" povezan je sa zadacima koji mu prethode u fazama modela vodopada.

Linearni proces omogućava jak upravljački okvir za proces planiranja razvoja softvera a kasnije i za kontrolu i praćenje. U okviru procesa planiranja i projektovanja moguće je identifikovati određene ključne tačke (prekretnice) koje mogu služiti za praćenje napretka razvojnog procesa. Pored modela vodopada često se, kao linearan proces razvoja, koristi i model transformacije. Cilj modela transformacije je da omogući konverziju formalne specifikacije sistema u programski kod. Prilikom upotrebe modela transformacije potrebno je detaljno analizirati specifikaciju sistema pre izvršenja transformacije.

EKONOMSKI FAKTORI

Cena ispravke greške u procesu projektovanja otkrivene u fazi testiranja softvera je deset puta veća od ispravke greške uočene u fazi implementacije.

Jedan od identifikovanih problema kroz proces projektovanja je apstraktnost softvera zbog čega je teško identifikovati sve potrebe softvera i rešenja problema koji je predstavljen kroz specifikaciju zahteva. Nastavak obrazloženog problema vodi ka loše definisanom prostoru mogućeg rešenja a samim tim i do procesa projektovanja koji sadrži greške. Greške koje se mogu javiti su da:

- projektovanje nije samoodrživo i kao takvo ne može se uspešno primeniti
- projektovanje i specifikacija nisu održivi
- projektovanje i zahtevi nisu održivi

Neusaglašenost u okviru procesa projektovanja može biti otkrivena u različitim fazama:

- kada se vrši preraspodela unutar procesa projektovanja (prerađivanje određenih zadataka) i upoređivanje informacije koje su dobijene iz različitih gledišta na softver
- u toku procesa razvoja softvera (kada dolazi do transformacije neusaglašenosti u programske strukture)
- u toku sprovođenja implementacije softvera

Procenat izmena u toku procesa projektovanja zavisi isključivo od oblika i tipa identifikovane greške. Verovatnoća da se neusaglašenost identificuje tokom testiranja softvera je veća nego u toku između faza specifikacije i projektovanja. Upotreborom i testiranjem prototipova omogućeno je ranije identifikovanje neusaglašenosti u fazama procesa razvoja softvera.

Verifikacija projektovanja softvera vrši se na osnovu softverske specifikacije. Provera neusaglašenosti između stvarnih zahteva korisnika (potreba) i projektovanja softvera naziva se validacija. Razlika između procesa verifikacije i validacije je definisana od strane Boehma i glasi:

- verifikacija: da li pravilno proizvodimo proizvod?
- validacija: da li proizvodimo pravi proizvod?

NEUSAGLAŠENOSTI U PROCESU PROJEKTOVANJA SOFTVERA

Neusaglašenosti u procesu projektovanja moraju biti otkrivene u što ranijoj fazi kako bi projektant doneo odluku u načinu ispravke uočene greške.

Uočavanje grešaka u kasnijim fazama procesa projektovanja zahteva dodatnu analizu ispravljanja grešaka i kako ispravljanje neusaglašenosti utiče na druge delove softvera. Takođe, potrebno je planirati i kasnije održavanje softvera jer se vremenom potrebe korisnika sistema menjaju kao i okolina u kojoj se softver koristi (operativni sistemi računara, performanse hardvera). Tri tipa održavanja softvera su:

1. održavanje koje se bavi proširivanjem i unapređenjem operativnog softvera uz dodavanje novih funkcionalnosti (**Perfective maintenance**)
2. održavanje kojim se vrše potrebne promene nametnute van specifikacije zahteva (primer može biti izmene u zakonodavstvu određene države ili promena operativnog sistema) (**Adaptive maintenance**)

3. održavanje koje za cilj ima ispravku uočenih grešaka u toku operativnog rada sistema (**Corrective maintenance**)

Prvi tip održavanja se najviše koristi u praksi jer treba omogućiti da se softver modifikuje i prilagodi očekivanim promenama u svom životnom ciklusu. Korišćenje ovog načina održavanja omogućava stvaranje ograničenja koja projektantu softvera sužavaju prostor za rešenje a samim tim povećavaju početni trošak a smanjuju dugoročne troškove.

DUGOROČNI PLAN IZMENA

Proces projektovanja se završava kada je ispunjen određeni plan razvoja softvera.

Kao što je već navedeno, često se u toku korišćenja softvera javi potreba za određenim modifikacijama i unapređenjima softvera. Planiranje dugoročnih izmena u procesu projektovanja zavisi od nekoliko faktora. Jedan od faktora je budžet. Ukoliko održavanje nije unapred planirano često se dešava da programeri nemaju podsticaj za izvršavanje izmena u toku procesa održavanja. Drugi faktor je nemogućnost projektanata softvera da stvore plan razvoja i održavanja softvera u budućnosti. Takođe, neadekvatna projektna dokumentacija može onemogućiti modifikovanje softvera shodno početnim planovima. Na osnovu Lehmanovog istraživanja (Lehman i Ramil, 2002) softverski proizvodi koji automatizuju ljudske ili društvene aktivnosti moraju se menjati i dodatno unapređivati kako bi zadovoljili potrebe korisnika. Primer softverskih proizvoda koji automatizuju ljudske ili društvene aktivnosti su: operativni sistemi moraju podržavati nove uređaje, programi koji se koriste u radnoj organizaciji moraju podržavati nove zadatke ili načine poslovanja definisane unutar organizacije, sistemi koji se bave plaćanjem moraju poštovati trenutne zakonske regulative. Fleksibilnost softvera, na osnovu istraživanja Parnasa (Parnas, 1979), ne može biti naknadna i mora biti planirana unutar specifikacije zahteva u toku procesa projektovanja. Takođe, projektovanje softvera se može smatrati opštim samo ako se može koristiti bez promene u različitim situacijama a može se smatrati fleksibilnim ukoliko se jednostavno menja u toku korišćenja u različitim situacijama.

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

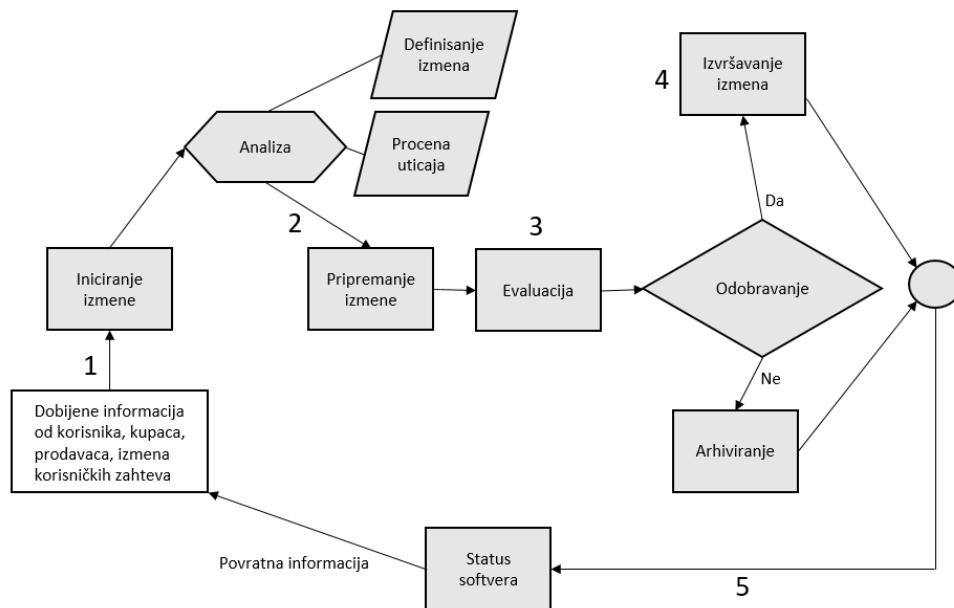
SOFTWARE DEVELOPMENT PROCESS (VIDEO)

Trajanje: 1:20 minuta.

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

POKAZNI PRIMER: DUGOROČNI PLAN IZMENA NA OSNOVU NEUSAGLAŠENOSTI U PROCESU PROJEKTOVANJA

Dugoročni plan izmena softvera sadrži nekoliko faza i aktivnosti kroz je potrebno proći u procesu projektovanja.



Slika 4.1.1 Primer dugoročnog plana izmene unutar procesa projektovanja softvera [Izvor: Budgen [2]]

4.1 Studija slučaja - Bibliotečki sistem

PRIMENA MODELA VODOPADA U PROCESU PROJEKTOVANJA BIBLIOTEČKOG SISTEMA

Kroz primer softvera za podršku bibliotečkog sistema biće predstavljena upotreba modela vodopada u procesu projektovanja softvera.

U početnoj fazi vrši se **prikupljanje zahteva** od bibliotečkog osoblja i budućih korisnika sistema. Definisana je potreba za projektovanjem objekta u okviru koga će biti informacije o knjigama koje imaju dugačke liste rezervacija. Na osnovu toga projektant softvera za bibliotečki sistem može razviti mogućnost kreiranja izveštaja iz navedenog objekta u svrhu dodavanja dodatnih kopija knjige od strane bibliotečkog osoblja.

Faza analize treba da kao rezultat prikaže model funkcionisanja biblioteke, listu funkcija koje sistem treba da ima kao i predviđanje ponašanja sistema u određenim uslovima korišćenja. Za potrebe bibliotečkog sistema, unutar faze analize, identifikovana je funkcija rezervisanja

knjige kada je broj kopija određene knjige koji je moguće iznajmiti prekoračen. Analiza treba da odredi da li sistem treba da "zamrže" mogućnost rezervisanja iznajmljivanja knjige koja trenutno dostupna za preuzimanje (shodno organizacionom modelu biblioteke) ili treba da omogući da korisnik izvrši rezervaciju a da sistem kada knjiga bude dostupna obavesti korisnika koji je prethodno rezervisao kopiju knjige. Organizacioni model biblioteke predstavlja dokument koji između ostalog sadrži i pravila o iznajmljivanju knjiga i takođe ulazi u fazu analize unutar procesa projektovanja. Projektant softvera mora da ima uvid u sva pravila biblioteke kako bi izvršio planiranje i implementaciju u budući sistem.

Faza specifikacije opisuje način organizacije interakcije korisnika sa bibliotečkim sistemom. Projektovanje interfejsa bibliotečkog sistema treba da obezbedi efikasnost i lakoću korišćenja kao i pregled velikog broja informacija koje će biti dostupne unutar sistema. Efikasnost sistema direktno utiče na brzinu rešavanja problema bibliotečkog osoblja kao i na interakciju sa korisnicima sistema. Projektovanje interfejsa aplikacije zavisi i od sopstvenog mentalnog modela projektanta i načina na koji on definiše strukturu informacija unutar sistema.

U slučaju sekvencijalnog procesa projektovanja softvera često je potrebno vratiti se na prethodnu fazu i izvršiti reviziju dobijenih rezultata faze. Razlog za povratak na prethodnu fazu je uočavanje nedoslednosti ili propusta u nekoj od narednih faza projektovanja. U procesu projektovanja softvera podrazumeva se i razvoj različitih prototipova budućeg sistema kako bi se uočio i eventualno unapredio način organizovanja funkcionalnosti sistema. Takođe, prototipovi mogu služiti i za ispitivanje tržišta ukoliko softver nema tačno definisane profile budućih korisnika u toku procesa projektovanja. Korisnost aplikacije može biti demonstrirana u ranoj fazi projektovanja a povratne informacije dobijene od korisnika koji koriste prototip mogu biti korišćene nastavku procesa projektovanja kako bi se na pravi način iskoristio pun potencijal budućeg softvera.

▼ Poglavlje 5

Proces projektovanja softvera

SOFTVER I ISPORUKA SOFTVERA

Softver predstavlja skup naredbi koji računaru daje instrukcije za izvršavanje određenog zadatka.

Početak razvijanja dovoljno velikih računarskih aplikacija zahtevao je detaljno projektovanje toka razvojnog procesa. Do sredine sedamdesetih godina prošlog veka softver je podrazumevan kao "stvaranje binarnog koda koji je namenjen izvršavanju na jednoj mašini". Struktura navedenog softvera bila je fiksirana i bilo kakva izmena zahtevala je pisanje većeg dela programskog koda iz početka. Nakon sedamdesetih godina prošlog veka razvila se ideja da se računarski programi mogu distribuirati na više računara uz mogućnost promene veze između njih u toku rada. Ukoliko je tako razvijen sistem trebalo implementirati kao distribuirani, projektanti softvera su morali izvršiti analizu načina funkcionisanja, podataka koji se dele između računara, testiranje komunikacionih mehanizama koji mogu biti korišćeni i testiranje performansi.

Nastavak varijacija u različitim oblicima implementacija nastavljen je i u devedesetim godinama kroz razvoj interneta. Distribucija je dodatno proširena kao i različite mogućnosti povezivanja (primer "klijent-server"). Takođe, razvoj statičnih opisnih oznaka kao što su formati HTML i XML omogućio je primernu različitih šablona koje softver može da dobije kao ulazni podatak. Koncepti arhitektonskog stila takođe su došli do izražaja i bilo ih je moguće primeniti na razvoj softvera u delu implementacije.

Jedan faktor koji se pojavio sa razvojem interneta je **stalna potražnja za bržom isporukom softvera**.

Navedeni faktor obezbeđuje i prilagođavanje i proširenje prvobitnih zahteva, načina na koji sistem radi kao i opisa kako taj sistem treba da funkcioniše. Jedna od karakteristika dobrog načina projektovanja je čuvanje strukture softvera iako je potrebno izvršiti proširenje ili modifikaciju. Takođe, potrebno je obratiti pažnju na vreme odziva i pouzdanost nakon svake izvršene izmene.

Jedan od načina za postizanje brže isporuke softvera je i upotreba višeg stepena apstrakcije u procesu projektovanja softvera. Tokom sedamdesetih godina brža isporuka softvera postignuta je prelaskom na pisanje mašinskog-specifičnog asemblerorskog programskega koda. U tom slučaju softver je bio oslobođen zavisnosti od jedne određene marke i vrste računara a takođe je ubrzao "proizvodnju" koda (pisanje i upotrebu prethodno napisanog programskega koda). Navedeni princip moguće je preslikati na proces projektovanja kroz razvoj komponenata i objekata softvera koji kasnije mogu biti ponovno iskorišćene u drugim delovima softvera bez potrebe za korišćenje novih.

MODELI RAZVOJA SOFTVERA

Projektovani modeli se koriste za predstavljanje namera projektanta softvera, istraživanje potencijalnih ograničenja rešenja, procene ponašanja i strukture.

Glavni faktori koji utiču na razvoj softvera su:

- **Složenost.** Osnovna osobina softvera u kome postoje različiti delovi koji zahtevaju više različitih stanja tokom izvršenja određene funkcije. Složenost softvera direktno zavisi od projektanta softvera.
- **Usaglašenost.** Od softvera se očekuje da bude fleksibilan i usklađen sa standardima drugih komponenti kao što su hardver ili softver koji se koristi kao podrška.
- **Promenljivost.** Kako softver stalno treba modifikovati i prilagođavati potrebama korisnika, potrebno je omogućiti jednostavnost izvršavanja izmena.
- **Nevidljivost.** Ukoliko je softver "nevidljiv" (nedostupan krajnjem korisniku) bilo kakvi oblici opisa koji se nalaze u softverskoj dokumentaciji neće biti dovoljni da projektant softvera utvrdi da li je softver u skladu sa specifikacijom zahteva. Primer može biti uzet iz arhitekture gde projektant može vizuelno uporediti plan izgradnje zgrade sa izgledom zgrade. Softver koji se smatra nevidljivim nije pravilno projektovan i kao takav ne može biti analiziran.

Poslednji faktor (nevidljivost) ukazuje na važnost procesa projektovanja softvera i probleme koji se mogu javiti. Takođe, vrlo bitno je pravilno projektovati model softvera koji se koristi za postavljanje krajnjeg rešenja. Različiti modeli mogu biti korišćeni za predviđanje ponašanja sistema u različitim scenarijima korišćenja ili u različitim kontekstima primene.

INICIJALNI MODEL PROJEKTOVANJA SOFTVERA

Inicijalni model projektovanja softvera prikazuje ulazne parametre (specifikacija zahteva, odluka projektanta i ograničenja) koji za rezultat daju opis programa koji se koristi u fazi implementaci

Na slici 1 prikazan je osnovni model projektovanja softvera. Ulaz u proces projektovanja predstavlja **dokument** specifikacije zahteva, eventualna ograničenja od strane organizacije ili ograničenja uočena na osnovu iskustva projektanta softvera. Takođe, odluke projektanta softvera u procesu projektovanja mogu biti definisana kao ulazni parametar. Proces projektovanja softvera može biti podeljen na dve faze:

1. Prva faza obuhvata razvoj apstraktnog modela rešenja od strane projektanta (arhitektonsko ili logičko projektovanje) u okviru koga su uključena samo eksterna svojstva elementa modela. U ovoj fazi projektovanja akcenat je na prirodi i obliku identifikovanog problema a manje na mogućem rešenju.
2. Druga faza podrazumeva mapiranje delova problema koji su definisani u prvoj fazi na specifične jedinice (detaljno ili fizičko projektovanje). Izlaz iz ove faze

projektovanja predstavlja specifikacije potrebne programeru za proces implementacije.



Slika 5.1 Model projektovanja softvera [Izvor: Budgen [2]]

PRIMENA RAZLIČITIH KONCEPATA I OGRANIČENJA U PROJEKTOVANJU SOFTVERA

Pored identifikovanja ponašanja sistema upotreba različitih modela omogućava projektantu softvera i da izvrši unapređenje i analizu početnih ideja.

Pomenuti arhitektonski model koji se koristi da opiše planiranu formu sistema uglavnom je vrlo apstraktan. Projektant softvera, na osnovu studije koji su izveli Adelson i Soloway (1985), treba da primeni sledeće koncepte :

- omogućiti korišćenje apstraktnih "mentalnih modela" projektanta u svrhu simulacije dinamičkog ponašanja sistema koji će biti rezultat procesa projektovanja
- omogućiti proširivanje detalja razvijenog modela zadržavajući sve detalje projektovanih delova softvera
- u slučaju rešavanja nepoznatog problema definisati što eksplicitnija ograničenja koja mogu uticati na proces projektovanja softvera (u svrhu rešavanja identifikovanog nepoznatog problema)
- omogućiti ponovno korišćenje prethodnih planova projektovanja softvera (prilikom ponovnog korišćenja određenog objekta ili komponente u određenom delu sistema)
- detaljno specificirati buduće izmene i planove za sistematsko proširenje softvera

U toku ranih faza projektovanja softvera formiraju se modeli na osnovu analize ili specifikacije problema. Projektant kroz razvoj određenog modela treba da predviđi buduće stanje i da ishod procesa projektovanja prilagodi navedenom stanju.

U praksi ima vrlo malo mogućnosti za slobodno projektovanje softvera jer svaki zadatak u projektovanju podrazumeva određena ograničenja.

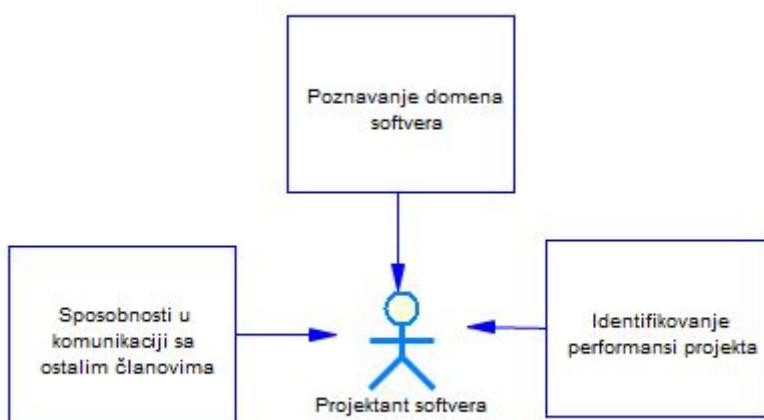
Ograničenja mogu biti nametnuta shodno određenom zadatku a neka ograničenja mogu biti prisutna shodno potrebi za usklađivanjem sa drugim softverskim proizvodima. U procesu projektovanja ograničenja se teško identifikuju. Veliki broj ograničenja identificuje se u specifikaciji zahteva za softver iako nije uvek obavezno i podrazumevano da se tamo nalaze. Uglavnom, ograničenja služe da ograniče ukupni prostor mogućeg rešenja projektantu softvera. Ograničenja su specifična za svaki novi identifikovani problem i potrebno ih je razmotriti u svakoj fazi projektovanja softvera. Praćenje ograničenja moguće je kroz redovne preglede i revizije faza procesa projektovanja.

PRENOŠENJE ZNANJA O PROJEKTOVANJU

Projektant softvera treba da poseduje određene karakteristike kako bi uspešno vodio proces projektovanja.

Projektant softvera (prikazan na slici 2) treba da poseduje tri značajne karakteristike:

1. Poznavanje domena aplikacije omogućuje jednostavno mapiranje strukture problema i strukture potencijalnog rešenja
2. Veštine u prenošenju tehničke vizije projektnog rešenja drugim učesnicima procesa projektovanja
3. Identifikovanje tehničkog napretka projekta



Slika 5.2 Karakteristike projektanta softvera [Izvor: Budgen [2]]

Proces projektovanja softvera obuhvata dve komponente:

- **Deo predstavljanja** (*representation part*) sadrži set opisnih oblika modela problema i opis strukturalnih osobina rešenja za moguće implementatore
- **Deo procesa** (*process part*) sadrži opis izvršavanja neophodnih transformacija modela u delu predstavljanja

Unutar dela predstavljanja vrši se identifikovanje osobina objekta u proces projektovanja (kao što su složene strukture podataka, broj parametara). Deo procesa bavi se:

- identifikovanjem projektnih aktivnosti koje je potrebno obaviti
- korišćenjem obrazaca reprezentacije
- procedurama za vršenje transformacija
- merenjem kvaliteta
- identifikovanjem određenih ograničenja koje je potrebno razmotriti
- operacijama verifikacije i validacije

METODA PREPOZNAVANJA, RESTRUKTURIRANJA I PROCEDURALNOG ZNANJA

Metode koje omogućavaju projektantu softvera smernice za moguće rešavanje problema.

Studija koju je sproveo Akin (1990. godine) definisala je tri klasične metode unutar projektovanja:

1. **Metoda prepoznavanja:** metoda u okviru koje projektant softvera prepoznaće rešenje problema koje je bilo u samom identifikovanom problemu
2. **Metoda restrukturiranja problema:** metoda koja koristi promenu stanovišta (ugla gledanja) na problem u cilju njegovog rešavanja
3. **Metoda proceduralnog znanja:** metoda koja podrazumeva analiziranje sličnih problema i načina njihovog rešavanja

Svaka transformacija u procesu projektovanja uključuje:

- ulazni model opisan grafičkom ili tekstualnom reprezentacijom problema
- izlazni model koji predstavlja rešenje (takođe može biti opisan grafički ili tekstualno)
- projektovanje ulaza u okviru koga projektant softvera definiše informacije o modelu

ODLUKE O DOKUMENTOVANJU PROCESA PROJEKTOVANJA

Potreba za dokumentovanjem odluka projektanta softvera bitna je za praćenje izvršavanja zadatka i za kasnije održavanje softvera.

Ukoliko proces projektovanja zahteva određenu reviziju (bilo od drugog projektanta ili rukovodioca projekta) potrebno je svaku odluku detaljno obrazložiti i dokumentovati. U toku procesa revizije odluka, svaka odluka mora biti proverena. Proces održavanja softvera

podrazumeva da svaka odluka (ukoliko se radi o modifikaciji ili proširivanju softvera) bude unesena u inicijalni originalni model softvera. Informacije o odlukama i načinu izvršenja određene izmene pomažu u sagledavanju kompletног projektnog rešenja softvera. Glavna motivacija za beleženje razloga donošenja određenih odluka u procesu projektovanja je kontrola kvaliteta, kako u fazi projektovanja tako i u fazi održavanja softvera. Evidentiranje odluka u toku projektovanja softvera mora biti detaljno iako se javljaju problemi u procesu projektovanja. Struktura procesa projektovanja je važna za novog člana tima a ne način na koji je proces projektovanja realizovan.

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

PROJEKTOVANJE UNUTAR PROJEKTNOG TIMA

Potrebno je omogućiti komunikaciju unutar projektnog tima u cilju poboljšanja procesa projektovanja.

Projektovanje softvera u timu definiše dva pitanja:

1. Kako podeliti projektni zadatak u timu?
2. Kako integrisati pojedinačne doprinose u procesu projektovanja?

Članovi projektnog tima mogu imati specijalizovane uloge kao što su administratori ili stručnjaci za dokumentaciju a često postoji rezervne uloge u timu u vidu zamenika. Često se dešava da veliki broj članova tima može zauzeti ulogu glavnog projektanta softvera pa je komunikacijom u timu potrebno odrediti pravu osobu za ovu ulogu a ne narušiti integritet projektnog tima. Veličina tima često predstavlja jedan od važnih faktora za proces projektovanja softvera unutar jednog tima. Istraživanja su pokazala da:

- od 10 do 12 članova tima predstavlja gornju granicu za produktivan rad
- veliki uticaj na ostatak tima može imati mali podskup članova tima koji poseduju napredno znanje iz ugla domena aplikacije
- uticaj organizacionih problema kompanije (održavanje veze između programera i naručioca softvera)

Često se u toku procesa projektovanja softvera dešava da su projektantima potrebni operativni scenariji korišćenja sistema kako bi razumeli ponašanje i okruženje aplikacije. Takođe, scenariji nisu uvek u potpunosti preneti sa naručioca softvera na programera koji vrši implementaciju softvera na osnovu projektnog plana. Razlog je generalizovanje scenarija od strane naručioca softvera bez konkretne dokumentacije zahteva za softver.

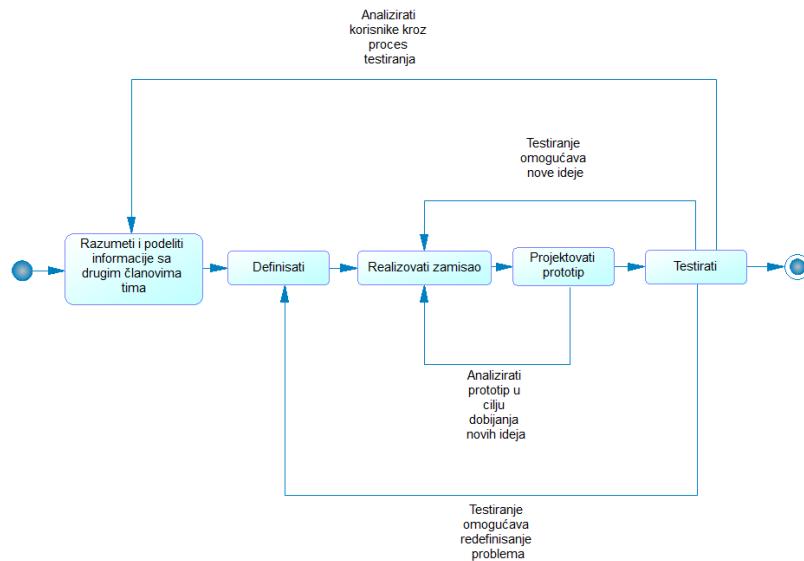
HOW TO BUILD A PROJECT TEAM (VIDEO)

Trajanje 04:12 minuta.

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

POKAZNI PRIMER: PROCES RESTRUKTURIRANJA SOFTVERA PO FAZAMA

Proces restrukturiranja softvera podrazumeva određene faze i aktivnosti koje je potrebno izvršiti.



Slika 5.3 Primer procesa restrukturiranja softvera [Izvor: Nebojša Gavrlović]

Faze koje su obuhvaćene su:

- razumeti i podeliti sa drugim članovima tima
- definisati
- realizovati zamisao
- projektovati prototip
- testirati

✓ Poglavlje 6

Pokazna vežba - projektovanje softvera

ZAHTEVANJE REŠENJA: SOFTVER ZA VOĐENJE KNJIGOVODSTVA

Za softver za vođenje knjigovodstva definisani su korisnički zahtevi koji predstavljaju ulazni parametar za fazu zahtevanja rešenja u procesu projektovanja.

Softver predstavlja sistem za vođenje knjigovodstva koje primenjuje sva načela prilikom knjiženja promena. Korisnik je u mogućnosti da vrši obradu izveštaja banke o poslovanju preduzeća, vođenje poslovnih knjiga (glavna knjiga, dnevnik) kao i vođenje bilansa kao što su bilans stanja i bilans uspeha. Postoji dosta aplikacija sličnih kao ova ali ni jedna nema toliko jednostavan interfejs koji ne zahteva dodatnu obuku korisnika.

Funkcije softvera:

- Obrada izveštaja banke
- Otvaranje poslovnih knjiga i bilansa stanja
- Evidencija poslovnih promena
- Generisanje bilansa uspeha
- Zatvaranje poslovnih knjiga
- Prihvatanja kursne liste sa sajta NBS

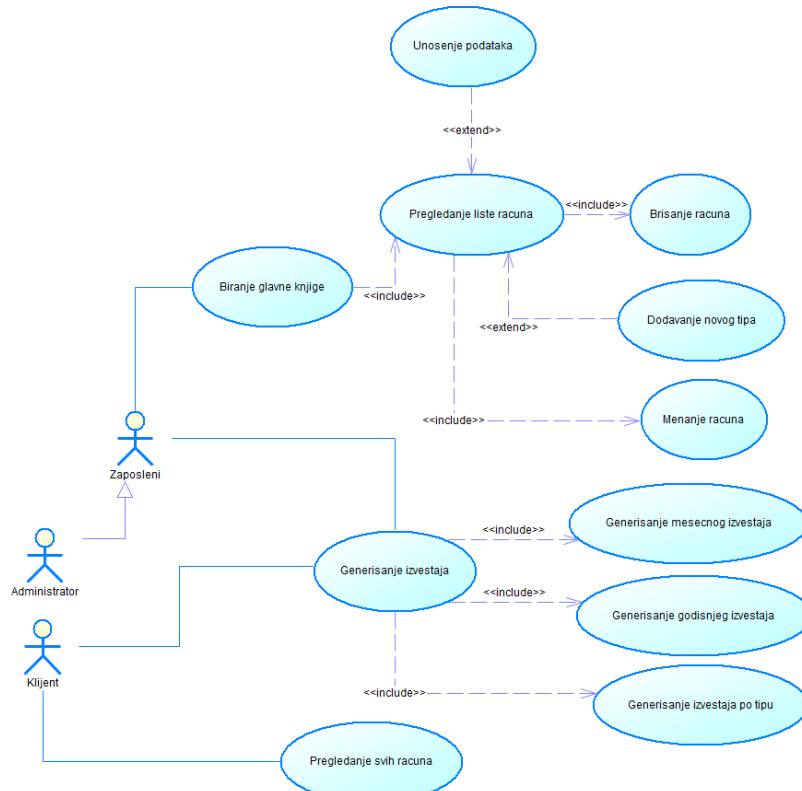
Jedini korisnik koga ovaj sistem ima je knjigovođa. Pre svega knjigovođa mora da ima validan nalog i da se uloguje u softver. Svaki računovođa ima za dozvoljeno da uradi sve gore navedene funkcije softvera. (10 min)

→	1.	Knjigovodja ima mogucnost izbora firme Pregled svih firmi za koje je zaduzen Izbor firme za koju zeli da radi pregled modifikaciju	REQ_0001	Undefined	Undefined	Draft
2	2.	Knjigovodja ima mogucnost pregleda informacija o firmi Pregled svih osnovnih podataka o firmi(naziv, adresa, pib, vlasnik)	REQ_0002	Undefined	Undefined	Draft
3	3.	Knjigovodja ima mogucnost pregleda glavne knjige Pregled svih racuna Filtriranje racuna po tipu Filtriranje racuna po datumu	REQ_0003	Undefined	Undefined	Draft
4	4.	Knjigovodja ima mogucnost unosa i izmene novog racuna	REQ_0004	Undefined	Undefined	Draft
5	5.	Knjigovodja ima mogucnost kreiranja novog tipa racuna	REQ_0005	Undefined	Undefined	Draft
6	6.	Knjigovodja ima mogucnost brisanja racuna	REQ_0007	Undefined	Undefined	Draft
7	7.	Knjigovodja i vlasnik imaju mogucnost prikazivanja izvestaja o firmi Izvestaj o ukupnom stanju po tipu Izvestaj na mesecnom nivou Izvestaj na godisnjem nivou	REQ_0006	Undefined	Undefined	Draft
8	8.	Vlasnik racuna moze pristupiti informacijama o svojoj firmi Pregled svih racuna	REQ_0008	Undefined	Undefined	Draft

Slika 6.1 Funkcionalni zahtevi softvera za vođenje knjigovodstva [Izvor: Nebojša Gavrlović]

MODEL REŠENJA SOFTVERA ZA VOĐENJE KNJIGOVODSTVA

Model rešenja može biti predstavljen različitim dijagramima koji opisuju ponašanje softvera.



Slika 6.2 Dijagram slučajeva korišćenja korisnika unutar sistema [Izvor: Nebojša Gavrlović]

Slučaj korišćenja predstavlja akciju aktera sa funkcionalnostima softvera. U modeli ovog slučaja korišćenja mogu se naći svi uslovi i rezultati za svaku komponentu ponaosob. Model rešenja softvera za vođenje knjigovodstva prikazan je na slici 2. Na osnovu početnog dijagrama moguće je predvideti i izvršiti eventualna unapređenja u kasnijim fazama projektovanja.(10 min)

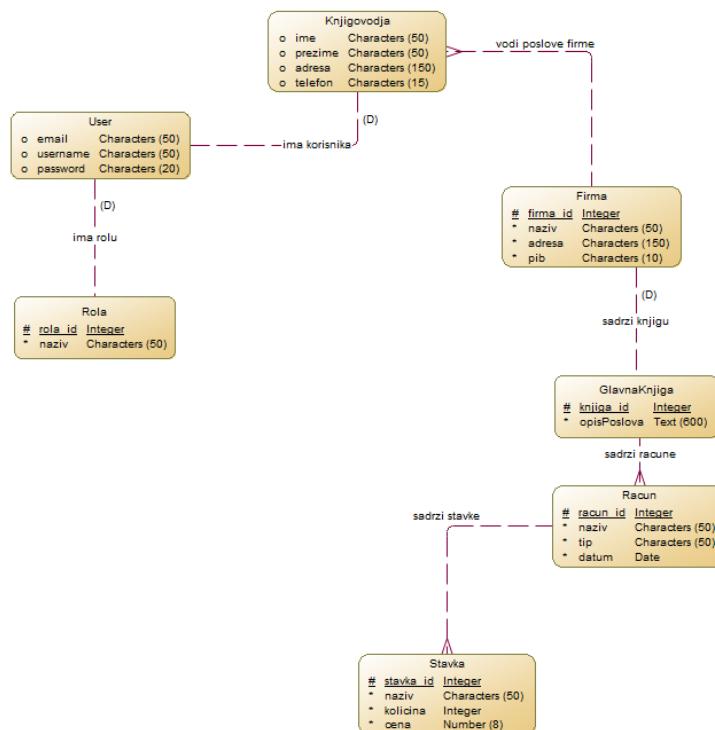
Preporuka je da se za model rešenja koristi dijagram slučajeva korišćenja.

PROJEKTNI MODEL SOFTVERA ZA VOĐENJE KNJIGOVODSTVA U FORMI KONCEPTUALNOG DIJAGRAMA

Projektni model softvera za vođenje knjigovodstva može biti predstavljen i konceptualnim dijagramom (ERD) sa opisom entiteta i identifikovanih veza u samom dijagramu.

U konceptualnom dijagramu mogu se naći sve potrebne informacije o entitetima baze podataka softvera kao i sve veze među njima (prikazan na slici 3). Projektni model omogućava nastavak procesa projektovanja i dalje analize softvera. Takođe, pored ERD dijagrama projektni model može biti predstavljen i dijagramom stanja i dijagramom toka podataka.(15 min)

Preporuka je da se za projektni model softvera koristi konceptualni dijagram ili dijagram stanja.

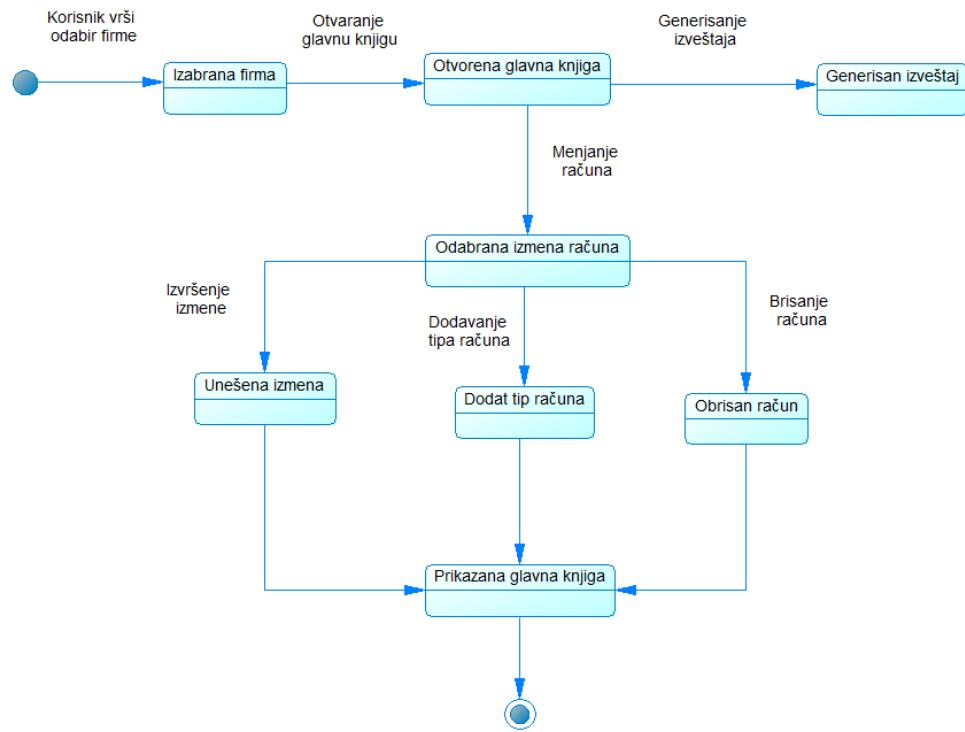


Slika 6.3 Konceptualni dijagram softvera za vođenje knjigovodstva [Izvor: Nebojša Gavrlović]

PROJEKTNI MODEL SOFTVERA ZA VOĐENJE KNJIGOVODSTVA U FORMI DIJAGRAMA STANJA

Projektni model softvera moguće je predstaviti i kroz dijagram stanja.

Na slici 4 prikazan je projektni model softvera za vođenje knjigovodstva u formi dijagrama stanja, iz drugog ugla u odnosu na predstavljanje softvera upotrebom konceptualnog dijagrama.(10 min)



Slika 6.4 Projektni model softvera za vođenje knjigovodstva u formi dijagrama stanja [Izvor: Nebojša Gavrlović]

ZAHTEVANJE REŠENJA - SISTEM ZA MENTORISANJE - MENTOR

Dat je primer sistema za mentorisanje osoba koje žele da nauče da programiraju ili unaprede svoje znanje u određenoj oblasti.

Ovaj sistem predstavlja web aplikaciju namenjenu za osobe koje žele da nauče da programiraju, unaprede svoje znanje u određenoj oblasti, mentorisu druge korisnike, pronađu tim za razvoj nekog sistema ili žele da se priključe nekom timu u razvoju. Za mentije koji imaju potrebu pronalaska odgovarajućeg mentora, kolaborativni sistem za učenje i pronalaženje mentora je sistem koji im omogućava lako kreiranje zahteva za koje se mentori prijavljuju, kao i lak odabir mentora iz liste prijavljenih. Za razliku od raznoraznih foruma i grupa, ovaj sistem će kroz podsistem za recenziranje i ocenjivanje mentora, mentiju dati realnu sliku kvaliteta mentora koji su se prijavili na mentorski rad, kao i mogućnost rešavanja zadataka zadatih od strane mentora. Za osobe koje traže saradnike u učenju, kolaborativni sistem za učenje i pronalaženje mentora je sistem koji im omogućava lako kreiranje sopstvenih projekta i odabir saradnika, kao i prijavljivanje na postojeće projekte. Za razliku od raznoraznih foruma i grupa, ovaj sistem omogućava korisnicima preglednije upravljanje projektima, saradnicima i svojim prijavama. Za mentore, tj. osobe koje žele mentorisati nekog korisnika, kolaborativni sistem za učenje i pronalaženje mentora je sistem koji im omogućava lako prijavljivanje na zahteve i pregledan prikaz svojih prijava. Za razliku od raznoraznih foruma i grupa, ovaj

sistem poseduje podsistem za ocenjivanje koji može pomoći mentoru da stvori jedan uspešan profil kvalitetnog mentora.(20 min)

Title ID	Full Description	Code
1.	Prijavljivanje na zahtev za mentorstvo <ul style="list-style-type: none">• Mentor ima mogućnost pregleda svih zahteva kreiranih od strane mentija.• Mentor ima mogućnost prijavljivanja na bilo koji zahtev koji je aktivan, nije istekao i nije zatvoren od strane mentija.• Mentor ima mogućnost pregleda svih svojih prethodnih prijava.	REQ_0001
1.1	Otkazivanje prijave <ul style="list-style-type: none">• Mentor ima mogućnost otkazivanja prijave na prethodno prijavljeni zahtev.	REQ_0004
2.	Zadavanje zadatka mentiji <ul style="list-style-type: none">• Mentor ima mogućnost zadavanja zadatka mentiji. Svaki zadatak mora sadržati detaljan opis problema i datum roka za predaju rešenja.	REQ_0002
3.	Ocenjivanje urađenih zadatka mentija <ul style="list-style-type: none">• Mentor ima mogućnost ocenjivanja urađenih zadatka mentija.	REQ_0003
4.	Pretraga zahteva <ul style="list-style-type: none">• Mentor ima mogućnost filtriranja zahteva prema jezicima i tehnologijama• Mentor ima mogućnost pretrage zahteva prema naslovu	REQ_0005

Slika 6.5 Funkcionalni zahtevi mentora [Izvor: Nebojša Gavrović]

ZAHTEVANJE REŠENJA - SISTEM ZA MENTORISANJE - MENTIJ I UČENIK

Dati su funkcionalni zahtevi (zahtevanje rešenja) za korisničke role mentij i učenik.

Title ID	Full Description	Code
1.	Kreiranje zahteva za mentorstvo <ul style="list-style-type: none">• Mentij ima mogućnost kreiranja zahteva za mentorstvo. Od podataka, mentij mora da unese naslov i opis. Osim ova dva podatka, mentij mora da odabere jezike i (ili) framework-ove za koje traži mentora.	REQ_0001
2.	Odabir mentora <ul style="list-style-type: none">• Mentij ima mogućnost pregleda svih prijavljenih mentora na svom zahtevu.• Mentij može odabrati samo jednog mentora na svom zahtevu.	REQ_0002
3.	Slanje urađenog zadatka mentoru <ul style="list-style-type: none">• Mentij ima mogućnost slanja urađenog zadatka mentoru.	REQ_0003
4.	Ocenjivanje mentora <ul style="list-style-type: none">• Mentij ima mogućnost ocenjivanja mentora. Ocenjivanje je moguće samo ukoliko mentij ima minimum 5 urađenih zadataka.	REQ_0004

Slika 6.6 Funkcionalni zahtevi mentija [Izvor: Nebojša Gavrović]

Title ID	Full Description	Code
1.	Kreiranje projekta <ul style="list-style-type: none">• Učenik ima mogućnost kreiranja projekta. Od podataka, učenik mora da unese naslov i opis i mora da odabere jezike i (ili) framework-ove projekta koji kreira.	REQ_0001
2.	Odabir saradnika <ul style="list-style-type: none">• Učenik ima mogućnost pregleda svih prijavljenih saradnika na svom projektu.• Učenik može odabrati više saradnika na svom projektu.	REQ_0002

Slika 6.7 Funkcionalni zahtevi učenika [Izvor: Nebojša Gavrović]

ZAHTEVANJE REŠENJA - SISTEM ZA MENTORISANJE - SARADNIK

Dati su funkcionalni zahtevi (zahevanje rešenja) za korisničku rolu saradnik

Title ID	Full Description	Code
1.	Prijavljivanje na projekat <ul style="list-style-type: none">Saradnik ima mogućnost pregleda svih projekata kreiranih na sistemu.Saradnik ima mogućnost prijavljivanja na projekat.	REQ_0001
1.1	Otkazivanje prijave <ul style="list-style-type: none">Saradnik ima mogućnost otkazivanja prijave na prethodno prijavljeni projekat.	REQ_0002
2.	Pretraga projekata <ul style="list-style-type: none">Saradnik ima mogućnost pretrage projekata po naslovuSaradnik ima mogućnost filtriranja projekata po jezicima.	REQ_0003

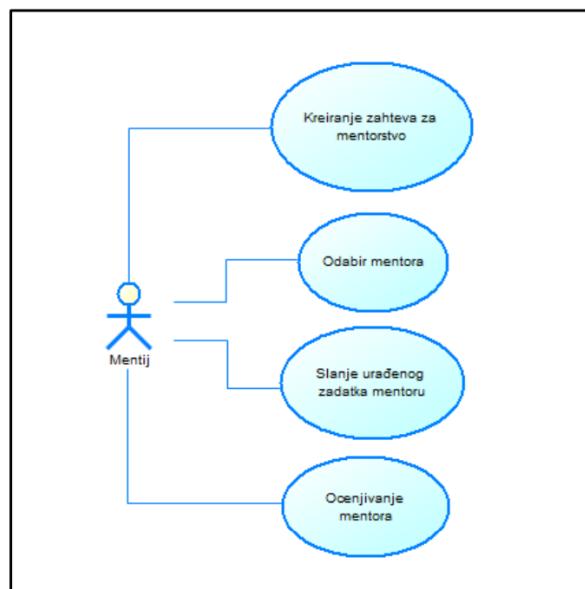
Slika 6.8 Funkcionalni zahtevi za korisničku rolu saradnik [Izvor: Nebojša Gavrlović]

MODEL REŠENJA SOFTVERA - SISTEM ZA MENTORISANJE

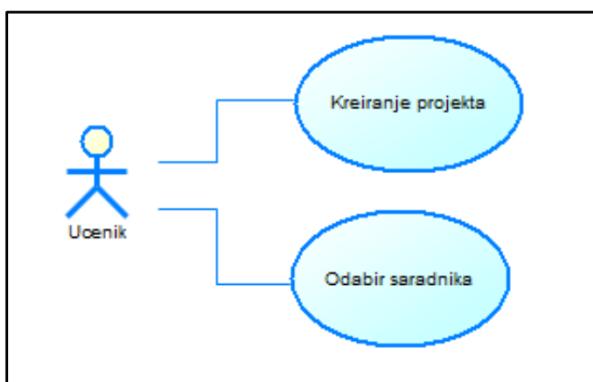
Model rešenja sistema za mentorisanje predstavljen je dijagramom slučajeva korišćenja za različite identifikovane korisničke role.



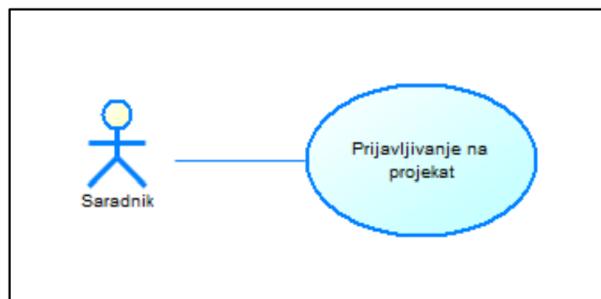
Slika 6.9 Slučajevi korišćenja – mentor [Izvor: Nebojša Gavrlović]



Slika 6.10 Slučajevi korišćenja – mentij [Izvor: Nebojša Gavrlović]



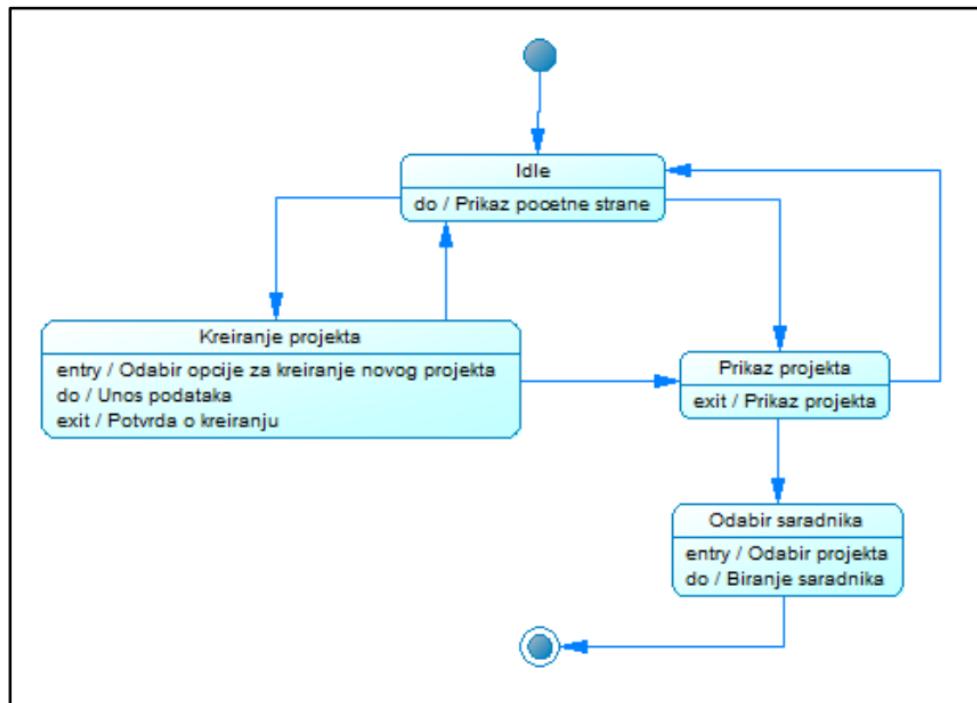
Slika 6.11 Slučajevi korišćenja – učenik [Izvor: Nebojša Gavrlović]



Slika 6.12 Slučajevi korišćenja – saradnik [Izvor: Nebojša Gavrlović]

PROJEKTNI MODEL - SISTEM ZA MENTORISANJE - DIJAGRAMA STANJA

Sledećim dijagramom stanja predstavljen je projektni model.



Slika 6.13 Projektni model sistema za mentorisanje [Izvor: Nebojša Gavrlović]

Nakon logovanja na sistem, učeniku je predstavljena početna stranica. Odatle, ima par opcija: da kreira novi projekat, pregleda svoj projekat ili da odabere saradnike za svoj projekat.(10 min)

▼ Poglavlje 7

Individualna vežba - Zadaci

ZADATAK ZA SAMOSTALNI RAD - PROCES PROJEKTOVANJA

Primeniti faze procesa projektovanja na softveru za automatsko očitavanje karata u autobusima gradskog prevoza.

Za navedeni softver koji vrši automatsko očitavanje karata u autobusima gradskog prevoza potrebno je:

1. Iz vašeg ugla gledanja napraviti specifikaciju mogućih korisničkih zahteva za softver (20min)
2. Specificirati fazu zahtevanja rešenja (20min)
3. Projektovati model rešenja i uraditi detaljan opis rešenja (30min)
4. Izvršiti evaluaciju modela rešenja (10min)
5. Izvršiti elaboraciju modela rešenja (10min)

ZADATAK ZA SAMOSTALNI RAD - ULOGE PROJEKTNIH AKTIVNOSTI

Potrebno je izvršiti modelovanje projektnog modela za softver za automatsko očitavanje karata u autobusima gradskog prevoza.

Na osnovu prethodno napravljene specifikacije korisničkih zahteva izvršiti modelovanje projektnog modela kroz sledeće dijagrame:

1. dijagrama stanja (15min)
2. ERD model (20min)
3. dijagram toka podataka (15min)

Detaljno opisati različite poglede na softver kroz modelovane dijagrame.

ZADATAK ZA SAMOSTALNI RAD - PROJEKTOVANJE U PROCESU RAZVOJA SOFTVERA

Potrebno je opisati moguće korake u procesu održavanja.

Zadatak 1: Navesti primere i moguće korake u procesu održavanja za softver za automatsko očitavanje karata u autobusima gradskog prevoza(20min).:

- Održavanje koje se bavi proširivanjem i unapređenjem operativnog softvera
- Održavanje kojim se vrše potrebne promene nametnute van specifikacije zahteva
- Održavanje koje za cilj ima ispravku uočenih grešaka u toku operativnog rada sistema

Zadatak 2: Modelovati dugoročni plan izmena za opisani softver i detaljno opisati sve aktivnosti identifikovane u okviru plana(20min).

ZADATAK ZA SAMOSTALNI RAD - PROCES PROJEKTOVANJA SOFTVERA

Primeniti proces restrukturiranja na softver za automatsko očitavanje karata u autobusima gradskog prevoza.

Na primeru softvera za automatsko očitavanje karata u autobusima gradskog prevoza potrebno je prikazati faze procesa restrukturiranja softvera u timu. Po nalogu klijenta u softver je potrebno dodati mogućnost očitavanja kreditne kartice korisnika i plaćanje vožnje istom. Izmena treba da obuhvati sledeće faze restrukturiranja(30min):

- razumeti i podeliti sa drugim članovima tima
- definisati
- realizovati zamisao
- projektovati prototip
- testirati

ZADACI ZA SAMOSTALAN RAD

Potrebno je modelovati model rešenja, projektni modela, dugoročni plan izmena i omogućiti proces restrukturiranja na osnovu date specifikacije zahteva za softver.

Softver je namenjen za zaposlene u medicinskoj ustanovi (medicinska sestra, doktor, direktor klinike).

Svaki korisnik se prijavljuje i na osnovu prijave ima različite opcije korišćenja sistema. Opcije su sledeće:

1. Prijavljanje na sistem klinike
2. Otvaranje kartona pacijenta
3. Pristup kartonu pacijenta
4. Slanje pacijentovih podataka iz kartona lekaru
5. Prepisivanje terapije
6. Izdavanje recepta za leka
7. Davanje uputa za dalje lečenje

8. Pretraživanje zaposlenih
9. Potpisivanje naloga, zapisnika i pravilnika klinike
10. Prikaz pravilnika klinike

Na osnovu definisane specifikacije zahteva za softver potrebno je uraditi sledeće zadatke:

1. Modelovati model rešenja (20 min)
2. Modelovati projektni model (25 min)
3. Napraviti dugoročni plan izmena softvera (25 min)
4. Prikazati mogući proces restrukturiranja softvera (10 min)

Napraviti jedan dokument u okviru koga će biti dokumentovani i detaljno opisan svako rešenje zadatka. (10 min)

ZADACI ZA DISKUSIJU NA VEŽBI

Otvorite diskusiju po svakom od dole postavljenih pitanja.

1. Koji pristupi se mogu koristiti za rešavanje problema u procesu projektovanja? (10min)
2. Da li je moguće preskočiti neku fazu projektovanja softvera ukoliko tim odluči da je tako nešto moguće?(10min)
3. Objasnite kako odluke u toku dokumentovanja procesa projektovanja mogu da utiču na proces razvoj softvera?(10min)
4. Na koji način dugoročni plan izmena može da utiče na ekonomski faktore procesa razvoja softvera?(10min)
5. Pojasnite na koji način inicijalni model projektovanja softvera može biti izmenjen u nekoj od faza projektovanja?(10min)

✓ Poglavlje 8

Domaći zadatak br.1

PRAVILA ZA IZRADU DOMAĆEG ZADATKA

Student zadatke treba samostalno da uradi i da dostavi asistentu

Domaći zadatak br.1: Osmisliti i opisati modifikaciju softvera koja spada pod "održavanje koje za cilj ima ispravku uočenih grešaka u toku operativnog rada sistema"
Pri radu, koristite Power Designer.

Domaći zadaci su dobijaju se po definisanim instrukcijama. Domaći zadatak se imenuje: SE311-DZ01-ImePrezime-brIndeksa gde su vrednosti Ime, Prezime i br.Indeksa vaši podaci.
Domaći zadatak je potrebno poslati na adresu asistenta:

nebojsa.gavrilovic@metropolitan.ac.rs (tradicionalni studenti iz Beograda i internet studenti)
jovana.jovic@metropolitan.ac.rs (tradicionalni studenti iz Niša)

sa naslovom (subject mail-a) SE311-DZ01. Potrebno je poslati modelovane dijagrame i dokument sa opisom dijagrama ili odgovorima na pitanja.

Napomena: Domaći zadaci treba da budu realizovani u zadatku navedenom razvojnog okruženju i da predstavljaju jedinstveno rešenje svakog studenta. Prepisivanje i preuzimanje rešenja sa interneta ili od drugih studenata strogo je zabranjeno.

▼ Poglavlje 9

Zaključak

ZAKLJUČAK

U okviru ove lekcije prikazana je priroda procesa projektovanja u opštim uslovima u različitim oblastima. Posebno važne ideje koje su identifikovane su:

- proces projektovanja se bavi opisivanjem načina koji dovodi do rešenja identifikovanog problema
- projektovanje omogućava informacije potrebne za modelovanje projektnih ideja kao i predstavljanje plana projektovanja osobama zaduženim za dalju implementaciju
- apstrakcija se koristi u procesu rešavanju problema kao i za razdvajanje logičkih od fizičkih aspekata procesa projektovanja

Razumevanje procesa projektovanja softvera zahteva procenu i važnost njegove uloge u procesu razvoja softvera a samim tim određuje i oblik pristupa i način projektovanja. Takođe, u okviru lekcije prikazana je i:

- primena različitih modela životnog ciklusa u procesu projektovanja
- uloga prototipova u procesu projektovanja
- način procenjivanja koliko proces projektovanja zadovoljava potrebe korisnika
- upotreba verifikacije i validacije
- uticaj troškova ispravljanja grešaka u određenim fazama projektovanja softvera

Proces projektovanja softvera podrazumeva i:

- složenost procesa izgradnje modela softvera i prikazivanje određenog identifikovanog ponašanja unutar softvera
- uticaj prirode softvera na opis softvera u procesu projektovanja
- potrebu za poznavanjem domena od strane projektanta softvera

LITERATURA

U ovoj lekciji korišćena je kao obavezna literatura, referenca 2, poglavljje 1, poglavље 2 i poglavljje 3 (2. izdanje).

Obavezna literatura:

1. Onlajn nastavni materijal na predmetu SE311 Projektovanje i arhitektura softvera, školska 2018/19, Univerzitet Metropolitan
2. D. Budgen, Software Design, 2nd ed., Addison-Wesley, 2003. poglavljje 1 (strane 3-24), poglavljje 2 (strane 25-44), poglavljje 3 (strane 45-62)

Dopunska literatura:

1. D. Budgen, Software Design, 2nd ed., Addison-Wesley, 2003.
2. P. Clements et ad. , Documenting Software Architectures: Views and Beyond, 2nd ed., Pearson Education, 2010.
3. T.C.Lethbridge, R. Lagariere - Object-Oriented Software Engineering - Practical Software Development using UML and Java - 2005
4. Ian Sommerville, Software Engineering, Tenth Edition, Pearson Education Inc., 2016. ili 9th Edition, 2011
5. P.B. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, vol. 12, no. 6, 1995, pp. 42-55.
6. E. Gamma et al., Design Patterns:Elements of Reusable Object-Oriented Software, 1st ed., Addison-Wesley Professional, 1994.
7. J. Nielsen, Usability Engineering, Morgan Kaufmann, 1993.
8. Service-oriented Architecture, Concept, Technology, and Design, autora T. Erl u izdanju Prentice Hall, 2005, ISBN 0-13-185858-0.
9. P. Stevens, Using UML – Software Engineering with Objects and Components, Second Edition, Assison-Wesley, Pearson Eduction, 2006
10. R. Pressman, Software Engineering – A Practitioner's Approach, Seventh Edition, McGraw Hill Higher Education, 2010

Veb lokacije :

- <http://www.software-engin.com>
- <http://www.uml.org/>