Prototipski pristup razvoju informacijskog sustava

Štimac, Siniša

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET EKONOMIJE I TURIZMA
"DR. MIJO MIRKOVIĆ"

SINIŠA ŠTIMAC

PROTOTIPSKI PRISTUP RAZVOJU INFORMACIJSKOG SUSTAVA

Završni rad

Pula, 2015.

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET EKONOMIJE I TURIZMA
"DR. MIJO MIRKOVIĆ"

SINIŠA ŠTIMAC

PROTOTIPSKI PRISTUP RAZVOJU INFORMACIJSKOG SUSTAVA

Završni rad

JMBAG: 2421000553
Studijski smjer: Informatika

Predmet: Projektiranje informacijskih sustava
Mentor: prof.dr.sc Vanja Bevanda

Pula, travanj 2015.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Siniša Štimac, kandidat za prvostupnika Informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student:

U Puli, . . 2015. _____

SADRŽAJ

Uvod

U ovom radu će biti obrađen prototipski pristup razvoju informacijskih sustava odnosno procesi, pristupi, strategije, metode, alati koje koristimo u svrhu izrade informacijskih sustava. Cilj rada je opisati prototipski pristup, što on omogućava i aktivnosti procesa u razvoju različitih informacijskih sustava i aplikacija.

Obradit će se glavni pristupi razvoja pomoću prototipa koji imaju različite ciljeve: istraživanje, eksperimentiranje, evoluciju. Vrste prototipa biti će prikazane kroz strategije i metode koje ih koriste kako bi se olakšao razvoj, smanjili troškovi i skratio razvojni ciklus informacijskog sustava.

Informacijski sustavi mogu biti različiti i samim time zahtijevaju različite pristupe analizi i dizajnu. Istraživanjem karakteristika informacijskog sustava može se odrediti hoće li sustav biti efikasnije izrađen predvidivim, tradicionalnim metodama ili prilagodljivim, suvremenim metodama koje uključuju prototipski pristup. Kao predvidiva, tradicionalna metoda razvoja informacijskih sustav obradit će se vodopadna metoda razvoja a onda će se obraditi pristupe i metode prototipskog razvoja nastale kao odgovor na nedostatke klasične, tradicionalne vodopadne metode. Opisat će se procese u takvim pristupima, metodama razvoja koji koriste prototipe kako bi se izradio efikasan, funkcionalan, cijenom prihvatljiv informacijski sustav i prikazati da se pomoću prototipskog pristupa uspješno može razviti suvremeni informacijski sustav. Na kraju će se opisati istaknuti alati koji pomažu pri procesu izrade prototipa informacijskog sustava ili aplikacija i obraditi CASE alat Oracle Designer.

1. Informacijski sustavi i razvoj

Informacijski sustav se može definirati kao kolekcija hardvera, softvera, podataka, ljudi, procedura koje rade zajedno u svrhu pružanja kvalitetnih informacija. Predstavlja skup svih resursa, odnosno: podataka, metoda, organizacije, tehničkih sredstava za pružanje informacija potrebnih za donošenje poslovnih odluka u cilju boljeg funkcioniranja organizacijskog sustava. Suvremeni informacijski sustavi oslanjaju se na računalnu podršku i sazidani su od programa, aplikacija koje prihvaćaju informacije, upisuju ih u bazu podataka, čitaju podatke, interpretiraju, kreiraju izvješća.

Kako bi se izradio kvalitetan informacijski sustav potrebno je izvršiti sustavnu analizu zahtjeva i zatim dizajnirati odgovarajući sustav na osnovu zahtjeva koji su definirani. Pristup sustavnoj analizi i dizajnu ovisi o kompleksnosti projekta odnosno o namjeni sustava i očekivanjima klijenta, korisnika.

Gary B. Shelly, Harry J. Rosenblatt opisuju razvoj i analizu sustava kao: Analiza sustava i dizajna je proces koji se odvija korak po korak kako bi se razvio informacijski sustav visoke kvalitete. Informacijski sustav kombinira informacijsku tehnologiju, ljude i podatke za podršku poslovnih zahtjeva (Shelly and Rosenblatt, 2012: 7).

U svrhu sustavne analize i dizajna informacijskog sustava mogu se koristiti razne metodologije i procesi. Za razvoj efikasnog, troškovno učinkovitog i kvalitetnog informacijskog sustava koristi se **Životni Ciklus Razvoja Sustava (Software Development Life Cycle, SDLC)** proces razvoja. SDLC je u osnovi serija aktivnosti koje pružaju model za razvoj i upravljanje životnim ciklusom sustava.

Životni ciklus informacijskog sustava je proces koji se sastoji od faze planiranja razvoja, analize, dizajna, konstrukcije, testiranja, implementacije i naknadnog održavanja. Svaki element je sačinjen od serija aktivnosti koje ovise o različitim razvojnim tehnikama koje su prilagođene kompleksnosti projekta ali svi projekti imaju navedene faze.

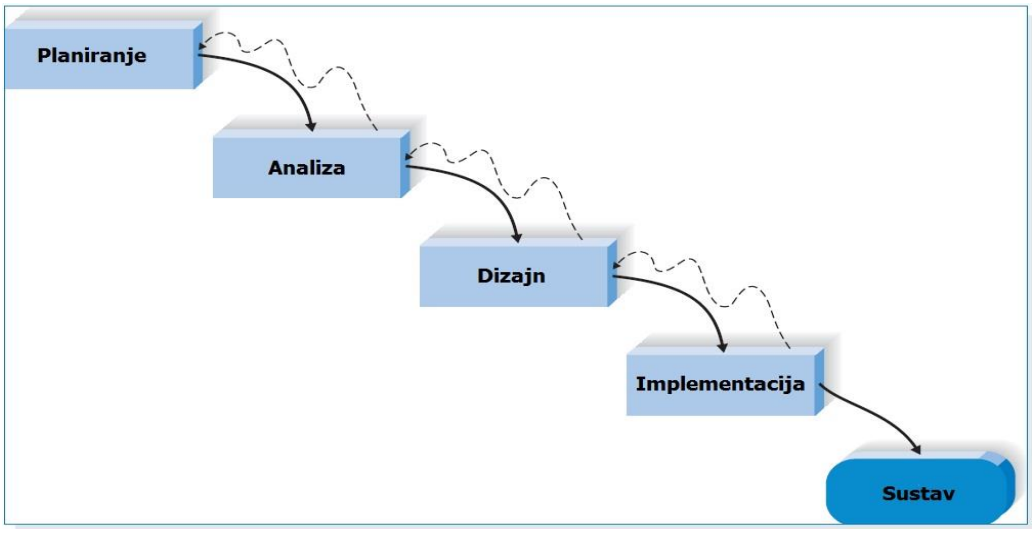
1.1. Pristupi SDLC razvoju (tradicionalni, suvremeni)

Pristup razvoju informacijskog sustava može biti različit zavisno o vrsti, ulozi sustava i zahtjevima klijenta. Različiti sustavi tako će koristiti različiti SDLC pristup kako bi se postigao najefikasniji razvoj. Pristupe razvoju je moguće sagledati s različitih aspekata te se mogu podijeliti na predvidive i prilagodljive odnosno na tradicionalne i suvremene pristupe.

Slika 1. „ Predvidiv razvoj naspram prilagodljiv razvoj „
Izvor: John W. Satzinger, Robert B. Jackson, Stephen D. Burd, Systems Analysis and Design in a Changing World, 5th Edition, str 39.

Predvidiv SDLC pristup pretpostavlja da se razvojni projekt može planirati, organizirati unaprijed i da se novi informacijski sustav može razvijati sukladno predviđenom planu. Predvidivi SDLC je koristan za izgradnju sustava koje se dobro razumije i dobro je definirani. (Satzinger i suradnici, 2010: 39)

Kao model koji ispunjava sve uvjete predvidivog razvoja može se izdvojiti klasičnu vodopadnu metodu razvoja. Ovaj stariji, tradicionalni model pristupa pretpostavlja niz različitih etapa koje su samostalne a na kraju svake etape započinje sljedeća. U vodopadnom razvoju korisnici prvo jasno definiraju zahtjeve za sustav i onda prepuštaju razvojnom timu da napravi analizu i dizajn sustava. Izrađeni kod se testira kako bi se potvrdila ispravnost sustava i nakon toga se implementira.



Slika 2. Vodopadni model

Izvor: Alan Dennis, Barbara Haley Wixom, Roberta M. Roth, Systems Analysis and Design, 5th edition, str 51.

U teoriji ovo je uredan prikaz modela ali rigidan i pojednostavljen. U stvarnom svijetu niti jedna od etapa nije samostalna i korisnici su često dobivali sustav koji nije bio prikladan prikaz željenog sustava. Često su prisutni nesporazumi tijekom dizajna, rade se kompromisi u dizajnu i donose su se krivi zaključci kod testiranja. Jasno je da takav jednostrani pristup nije idealan a nedostaci se često vide tek kad razvoj sustava dođe u točku visoke razvijenosti. Korisnik također može naknadno promijeniti, nadopuniti zahtjeve vezane uz sustav.

Prilagodljivi SDLC razvoj koristi pristup u kojem se projektne aktivnosti podešavaju, prilagođavaju kako projekt napreduje kako bi se izbjegli nedostaci i rizici u razvoju informacijskog sustava. Primjer ovakvog prilagodljivog i suvremenog pristupa su agilne metode koje identificiraju četiri glavne vrijednosti:

Individue i interakcija iznad procesa i alata. Fokusirati se na tim i interakcije u njemu a ne na formalne procese i sofisticirane alate.

Funkcionalan softver iznad detaljne dokumentacije.

Ne potrošiti sve vrijeme na izradu dijagrama i pisanih specifikacija. Koristiti modeliranje za rješenje problema i zatim pisati kod.

Suradnja s klijentima iznad pregovora o ugovoru. Bliski rad s klijentima i korisnicima tako da postignuto rješenje bude pravo rješenje.

Reakcija na promjenu iznad praćenja plana. Fleksibilnost u prihvaćanju izmjena u organizaciji i korisničkim potrebama (Satzinger i suradnici, 2010: 82).

Agilne metode daju naglasak na timski rad i razdvajaju razvojni proces na cikluse ili iteracije, u tu svrhu koriste se:

Spiralni pristup razvoju koji nastoji upravljati rizikom korištenjem prototipa u razvojnom procesu informacijskog sustava. Proces razvoja informacijskog sustava započinje konceptualnim dokumentom i napreduje u ciklusima do programiranja individualnih programa u spiralnom modelu. Svaki ciklus sadrži isti redoslijed s koracima: utvrđivanje zahtjeva, ograničenja i alternative koje rješavaju rizik korištenjem prototipa, simulacija.

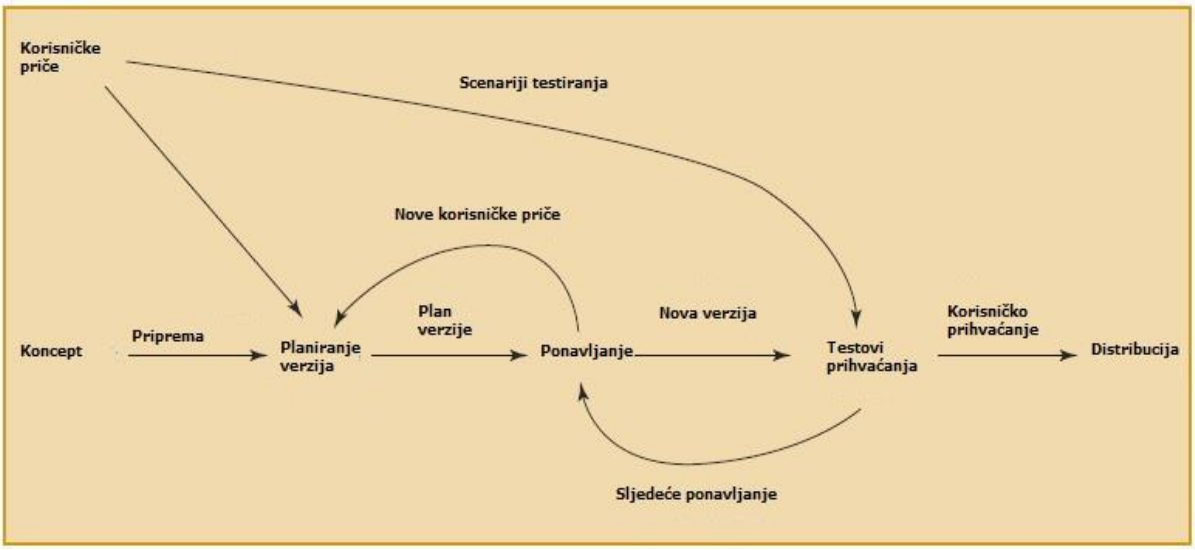
Iterativni pristup ja način razvoja gdje ponavljanjem (iteracijama) sustav u aktivnostima razvoja analize, dizajna, konstrukcije usavršavamo dok ne dobijemo željeni rezultat.

Kao modeli koji proizlaze iz Agilne metode razvoja ističu se XP (ekstremno programiranje) i Scrum.

Ekstremno programiranje (extreme programming, xp), ovaj suvremeniji model razvoja naglašava snažni timski rad i svaki sudionik koji pridonosi razvoju modela smatra se članom tima. Svaki zahtjev zamišljen od strane klijenta i razvojnog tima biva ocijenjena po osnovi cijene i potrebnog vremena za razvoj. Programeri obično rade na kodu u paru zajedno sa ostatkom tima kako bi svaki član bio upoznat s aktualnim kodom. Tim aktivno radi s klijentima, korisnicima na dnevnoj bazi i kontinuirano se izrađuje mala izdanja, koja klijent onda odabire, odnosno značajke koje će biti uključene u sljedeću verziju bazirano na osnovi cijene naspram koristi. Ovaj razvojni model se ističe svojom fleksibilnošću jer može funkcionirati i s visokim stupnjem dvosmislenih specifikacija od strane korisnika. Može se reći da brzo i efikasno reagira na promjene zahtjeva. Na ovakvoj vrsti razvoja može raditi do 25 programera, veći broj programera zahtjeva razdvajanje timova što onda usporava komunikaciju i usklađenost tako da najveću korist od ovakvog pristupa razvoja imaju mali i srednji sustavi.

Ekstremno programiranje se dijeli na četiri glavna procesa razvoja sustava: planiranje, dizajn, programiranje i testiranje.

Planiranje se odnosi na dva glavna pitanja u procesu razvoja: što će biti napravljeno do određenog dana te određivanje što treba sljedeće napraviti. Kod planiranja se koriste korisničke priče (user stories), koje su slične korisničkim scenarijima, a služe za potrebe planiranja izlazaka pojedinih verzija programa. Za svaku korisničku priču potrebno je izraditi test kojim će se potvrditi ispunjenje korisničke priče. Svrha ovih priča je da se procijeni trajanje implementacije prije same implementacije tako da programeri dobiju detaljnije specifikacije.



Slika 3. Pojednostavljeni prikaz Ekstremnog programiranja.
Izvor: Gary B. Shelly, Harry J. Rosenblatt, Systems Analysis and Design, Ninth Edition, 2012, str 512.

Na zajedničkom sastanku kreira se raspored verzija sustava. Taj raspored služi da bi se kreirali planovi za svaku pojedinačnu iteraciju. Preporuka je često izdavanje verzija koje ne trebaju biti značajno poboljšane, samo je važno otkriti funkcionalnosti od koji korisnik ima koristi te ih isporučiti čim prije. Sve programske aktivnosti nije potrebno planirati na samom početku projekta već se aktivnosti planiraju na početku pojedine iteracije. To je takozvano (just in time) planiranje iteracija koje omogućava brzo praćenje promjena korisničkih zahtjeva. U slučaju da unutar iteracije nije moguće izvršiti zadatke, rokovi se ne smiju pomicati te stoga treba izbaciti neke od zadataka. Potrebno je mjeriti i brzinu razvoja projekta. Pri tome se procjenjuje trajanje ispunjenja korisničkih priča u pojedinim iteracijama na temelju prijašnjih iteracija.

Prilikom dizajniranja sustava glavni cilj je jednostavnost jer je uz jednostavan dizajn projekt uvijek moguće brže izvršiti. U tu svrhu nikad se ne bi trebale dodavati nove funkcionalnosti prije roka koji je predviđen za to. U slučaju kompliciranih tehničkih ili dizajnerskih problema potrebno je kreirati zamjenska rješenja. Cilj takvog pristupa je smanjiti rizik mogućih tehničkih problema ili povećati pouzdanost procjena korisničkih priča.

Na kraju dizajna je potrebno primjenjivati redizajn, odnosno odbacivati nepotrebne funkcionalnosti, te ukloniti redundancije.

Tijekom programiranja koda važan je preduvjet dostupnost krajnjeg korisnika, budući da sve faze XP razvoja zahtijevaju komuniciranje s korisnicima. Preporuka je da se unaprijed kreiraju osnovni testovi za provjeru funkcionalnosti, prije same izrade programskog koda.

Time se ne produžuje izrada samog osnovnog programskog koda, a kasnije nije potrebno trošiti dodatno vrijeme na izradu testova. Vrlo je važno da se prilikom integriranja novog programskog koda u sustav koristi postepena integracija, odnosno da se zabranjuju slučajevi u kojima bi različiti programeri istodobno uključivali svoj programski kod i testirali ga. Svi programeri mogu slobodno mijenjati gotovi programski kod koji nisu sami napisali, kako bi dodali novu funkcionalnosti ili ispravili pogreške.

Testiranje je jedan od temelja ekstremnog programiranja a koristimo ga u svrhu provjere ispravnosti implementiranih funkcija. Za testiranje je potrebno uzeti ili razviti vlastito programsko okruženje pomoću kojega će se moći kreirati automatizirani testovi. Testove je potrebno kreirati prije razvoja samog programskog koda sustava. Razvijeni programski inkrement ne može biti integriran u cjelokupni sustav ako ga ne prate popratni testovi.

U slučajevima kada se u programskom kodu otkrije neki tip pogreške potrebno je kreirati testove za pogrešku kako bi se kasnije lakše detektirala u drugim modulima. Iz definiranih korisničkih priča kreiraju se testovi za prihvaćanje sustava a klijent definira scenarije koje je potrebno testirati na temelju definiranih korisničkih priča. Za jednu priču moguće je definirati jedan ili više testova.

Scrum proces razvoja predstavili su Ken Schwaber i Jeff Sutherland 1995 godine na Oopsla konferenciji u Austinu, Texas (US) i izdali znanstveni rad "SCRUM Software Development Process". Naziv scrum je posuđen iz analogije postavljene u istraživačkom radu „The New New Product Development Game" napisan od Takeuchi and Nonaka 1986 godine gdje su timovi visokih performansi uspoređeni sa scrum formacijama korištenim u ragbi timovima.

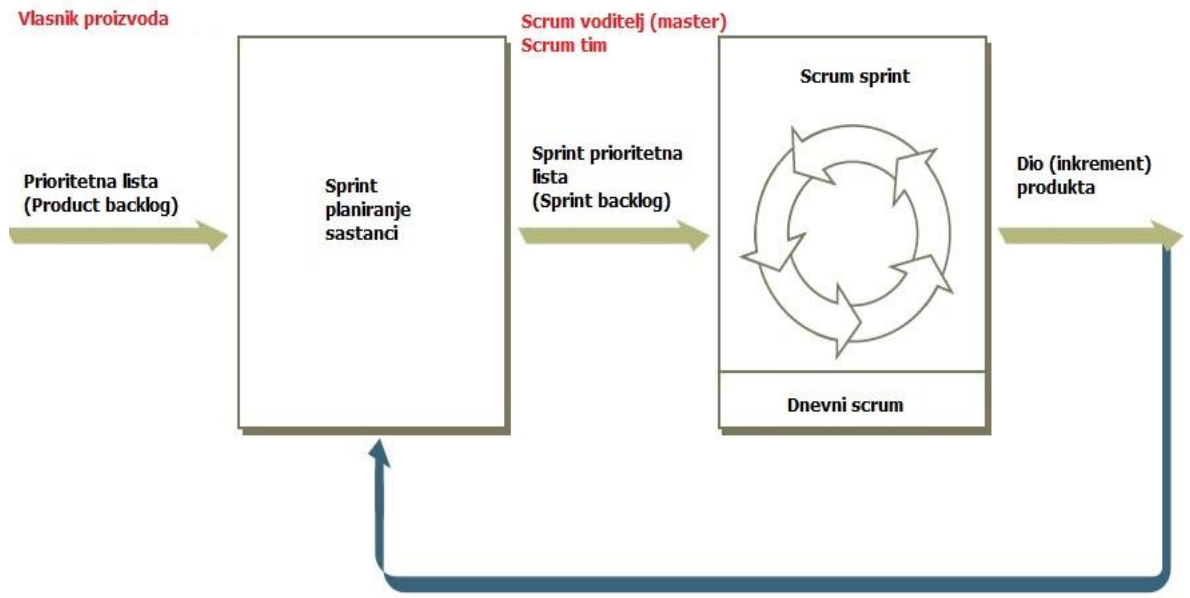
Filozofija scrum razvoja je osjetljivost na vrlo promjenjivu i dinamičku okolinu u kojoj je moguće da korisnici ne moraju znati što je točno potrebno i mogu često mijenjati prioritete. U ovakvom tipu okruženja promjene su toliko brojne da projekt može zastati i nikad ne biti završen. Scrum se ističe u ovakvim situacijama jer se prvenstveno fokusira na razvojni tim i njihov posao, daje naglasak na individue više nego na proces i opisuje kako razvojni timovi mogu raditi na razvoju sustava u serijama kratkih mini projekata. Ključ ovakve filozofije je kompletna kontrola tima stručnjaka nad organizacijom i napretkom razvoja. Sustav se razvija inkrementalno i nameće se kontrola fokusiranjem na dijelove koji se mogu razviti. Glavni mehanizam projekta je lista predmeta, prioriteta (product backlog) koje bi sustav trebao sadržavati. Ovakva lista sadrži funkcije, značajke, tehnologiju iz koje se odabiru prioriteti. Nakon odabira prioriteta vrši se razvoj samo njih nekoliko u isto vrijeme zavisno o potrebama projekta.(Satzinger i suradnici, 2010: 680)

U scrum timu postoje 3 uloge:

Vlasnik proizvoda je odgovoran za maksimizaciju vrijednosti proizvoda i rada razvojnog tima. Način na koji se to postiže vrlo je različit između raznih organizacija, timova i ljudi. Vlasnik proizvoda je jedini odgovoran za upravljanje **prioritetnom listom** (product backlog) a njegova uloga uključuje jasno objašnjavanje razvojnom timu: vizije, ciljeve i stavke na prioritetnoj listi. Razvojni tim radi prema naputcima vlasnika stoga i cijela organizacija mora poštivati njegove odluke.

Razvojni tim zadužen je za izradu dijelova (inkrmenta) koji trebaju zadovoljiti zahtjeve vlasnika proizvoda. Timovi u scrum razvoju su samo organizirajući.

Scrum voditelj (master) nije tradicionalni vođa tima pošto je razvojni tim samoorganizirajući. On ima ulog da nametne scrum tehnike razvoja i pomaže timu da ostvari svoj posao uklanjajući moguće prepreke za rad.



Slika 4. SCRUM razvojni proces

Izvor: John W. Satzinger, Robert B. Jackson, Stephen D. Burd, Systems Analysis and Design in a Changing World, 5th Edition, str 681.

Događaji u scrum razvoju dijele se na:

Sprint je jezgra scrum-a, vremenski ograničeni period od jednog mjeseca ili manje tijekom kojega se proizvede upotrebljiv i potencijalno isporučiv inkrement proizvoda. Sprintovi su jednakog trajanja tijekom cijelog razvoja proizvoda. Novi sprint započinje neposredno nakon što završi prethodni a sastoji se od sastanka za planiranje sprinta, dnevnog scrum-a, posla razvoja, revizije sprinta i retrospektive sprinta. Svaki sprint se može smatrati projektom čiji

horizont ne prelazi mjesec dana. Svaki sprint ima definirano što će se obaviti i na koji način. To će definirati izradu, opseg zadataka i konačni proizvod.

Dnevni scrum je 15-minutni, vremenski ograničen događaj, koji služi da razvojni tim uskladi aktivnosti i donese plan za sljedeća 24 sata. To se čini kontrolom rada od prethodnog dnevnog scrum sastanka i procjenom posla koji bi mogao biti odrađen prije sljedećeg sastanka. Dnevni scrum se održava uvijek na istom mjestu svaki dan da bi se smanjila kompleksnost. Scrum master forsira pravilo da samo razvojni tim sudjeluje na dnevnom scrum sastanku i da sastanak traje 15 minuta.

RAD (rapid application development) tehnika za brzi razvoj aplikacija je još jedna metoda koju se svrstava pod prilagodljivi razvoj**.** Tehnika za brzi razvoj aplikacija metodologija je nastala kao odgovor na slabosti vodopadnog modela. RAD uključuje specijalne tehnike, računalne alate kako bi ubrzali faze analize, dizajna, implementacije. Cilj je brzo izradili neki dio sustava, dati ga korisniku na evaluaciju i dobili povratne informacije (Dennis i suradnici, 2012: 54).

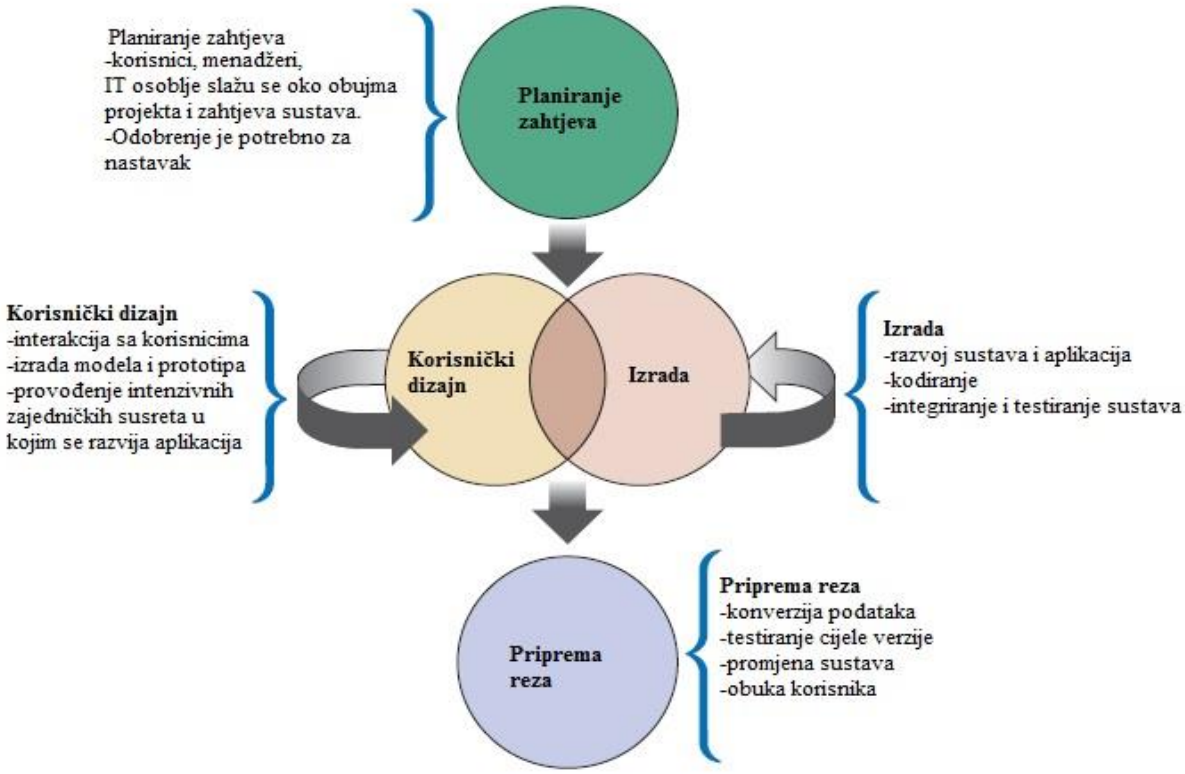
RAD ima četiri ključna aspekta, to su: metodologija, ljudi, menadžment i alati. U slučaju da jedan od ovih aspekata nije dostatan razvoj informacijskog sustava neće biti odrađen brzo. Metodologija RAD-a promiče veći, brži, bolji razvoj i ovisi o bliskoj suradnji svih četiri glavna aspekta uključena u razvoj. Kako bi se to postiglo koriste se najbolje dostupne tehnike evolucijskog razvoja prototipa, radione umjesto intervjua, automatiziranje razvojnih tehnika.

Ovakav razvoj zahtjeva aktivnu uključenost korisnika kako bi se izradio informacijski sustav koji korisnici zaista trebaju. Uspjeh tehnike za brzi razvoj aplikacija zavisan je o uključenosti ljudi koji posjeduju prave vještine i znanja.

Alati omogućavaju razvojnom timu da brzo i kvalitetno izrade informacijski sustav odgovarajućih karakteristika. Iako je vrlo važno koristiti prikladne alate za razvoj oni sami ne garantiraju uspješan razvoj.

Menadžment mora biti fokusiran na brzi razvoj i uloga mu je motivirati sve sudionike u razvoju.

Životni ciklus se odvija kroz četiri faze prikazane na slici i uključuje sve aktivnosti i zadaće potrebne da se obuhvati zahtjeve, dizajn, razvoj, implementaciju sustava.



Slika 5. Četri faze RAD tehnike razvoja inf. sustava.

Izvor: Gary B. Shelly, Harry J. Rosenblatt, Systems Analysis and Design, Ninth Edition, 2012, str 146

Planiranje zahtjeva (requirements planning), faza definira potrebne funkcije i podatke koje će sustav podržavati te određuje okvire sustava.

Dizajniranje s korisnicima (user design) faza koristi radionice kako bi s korisnicima odredili dizajn i model prototipa.

Izrada prototipa (construction), završava se izrada fizičkog dijela sustava.

Implementacija, završno testiranje od strane korisnika, pretvorba podataka i implementacija u informacijski sustav.

Fazu 2. i 3. dobila je naziv „radionica“ (workshop) zbog intenzivnog timskog rada na prototipu. Razvojni tim i korisnici aktivno rade na funkcionalnom prototipu i iznose se mišljenja i moguće modifikacije na modelu kako bi se unaprijedio dizajn.

2. Prototipski pristup razvoju informacijskih sustava

Prototipi se koriste u mnogim područjima proizvodnje. Proizvodni inženjeri izrađuju prototipe da istraže i kontroliraju slabosti u dizajnu proizvoda kako bi predvidjeli probleme u procesu izrade i korištenja prije nego se proizvod plasira ili počne masovno proizvoditi. Prototip može biti drugačije definiran zavisno o polju, industriji u kojoj će se koristiti. Za primjer može se uzeti prototip aviona u umanjenoj verziji originala.

Slika 6: Primjer korištenje prototipa u avio industriji
Izvor: Shelly Cashman , System Analysis and Design 9th Edition, str: 315

Industrija softvera prihvatila je ovu tehniku da se izradi prototip kao model, simulaciju, parcijalni dio sustava, i koristi u za različite svrhe testiranja. Može se reći da je cilj izrade prototipa u razvoju informacijskih sustava otklanjanja ili bar smanjivanje nesigurnosti vezane uz: procjena isplativosti projekta, funkcije i korisničke zahtjeve, redoslijed funkcija, ispravan prikaz sustava, izgled i osjet korisničkog sučelja, prikladnost dizajna za korištenje.

Razvoj informacijskih sustava pomoću prototipa koristi se već dugi period. Prototip informacijskog sustava može se prikazati kako konkretnu reprezentaciju jednog dijela ili cijelog interaktivnog sustava. Prototip je opipljiv proizvod a ne apstraktni prikaz koji zahtjeva objašnjenje. Svi uključeni u razvoj prototipa i njegovo korištenje mogu na osnovu prototipa predvidjeti krajnji izgled i funkcionalnost sustava, softvera.

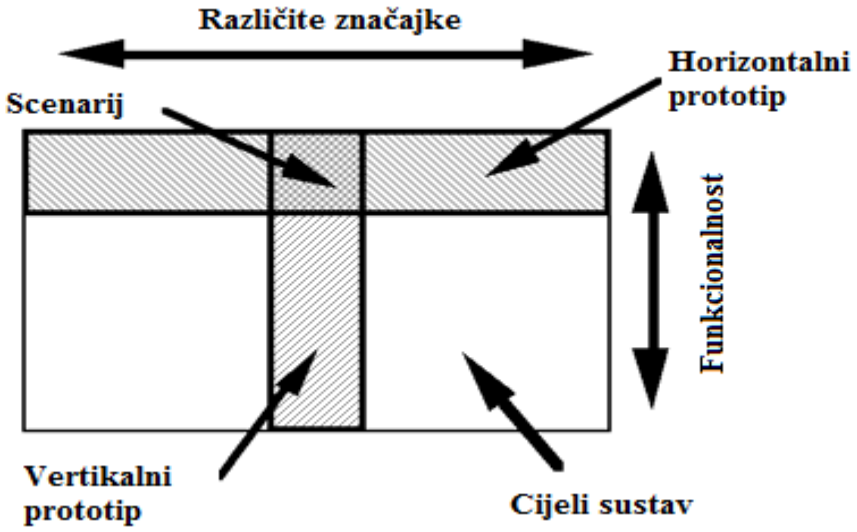
Jednu od definicija prototipa postavili su Naumann and Jenkins koji smatraju da bi prikladna definicija za prototip bila: “sustav koji obuhvaća osnovne značajke kasnijeg sustava također koji je “namjerno nekompletan, biti će modificiran, zamijenjen ili nadograđen (Nauman i Jenkins, 1982: 29).

Ovo nije moguće napraviti s interaktivnim prototipom sustava, dizajner može limitirati količinu informacija ali sučelje mora biti prezentirano u pravoj veličini. Odnosno sučelje prototipa može koristiti samo mali dio baze podataka ali mora prikazivati realan prikaz podataka i tehnike interakcije s njima. Neki prototipi interaktivnih informacijskih sustava započinju kao jedinstveni model koji se onda mogu distribuirati u većem broju i testirati. Najuspješniji prototip zatim evoluira odnosno integrira u finalni proizvod koji se i nakon masovne proizvodnje može još unaprijediti te nastaju nove verzije sustava, softvera. Kad se gleda prototip kao proizvod može se definirati neke osnovne značajke uspješnosti: pomažu programerima da generiraju i oblikuju ideje, olakšavaju otkrivanje potreba korisnika, potiču komunikaciju između dizajnera, programera. korisnika, klijenta omogućava ranu procjenu proizvoda pošto ga je moguće testirati od strane korisnika.

2.1. Strategije prototipskog razvoja

Dizajneri sistema moraju donesti odluku koju će ulogu prototip igrati u odnosu na finalni sustav i kojim redom će se kreirati određeni aspekti prototipa. Strategiju prototipskog razvoja se mogu definirati kao skup odluka koje diktiraju koja će se akcije provesti u svrhu ostvarivanja razvoja prototipa.

Horizontalni razvoj prototipa najčešći je kod velikih razvojnih timova, gdje dizajneri s različitim razvojnim znanjima rade na različitim slojevima softverske arhitekture. Primjer horizontalnog prototipa je korisničko sučelje koje je korisno za sagledavanje ukupne slike sistema iz korisničke perspektive. Odnosno pruža širi pogled na cijeli sustav ili pod-sustav i fokusira se više na interakciju korisnika sa sustavom umjesto na određene funkcije sustava kao što je pristup bazi podataka. U korisničkom sučelju se može iz korisničke perspektive vidjeti dosljednost (sličnim funkcijama se pristupa sličnim naredbama), da li su sve funkcije uključen, redundanciju odnosno pristup istim funkcijama kroz različite korisničke komande. Horizontalni prototip može započeti brzim prototipom i napredovati u funkcionalni kod. Softverski prototip se može izraditi s različitim razvojnim alatima bez potrebe za temeljnim funkcijama. To omogućava testiranje kako će korisnik biti u interakciji s korisničkim sučeljem bez brige kako će ostatak sistema funkcionirati. Ali mora se voditi računa da se u određenoj mjeri simulira interakcija s ostatkom sistema inače se ne bi moglo pravilno ocijeniti funkcionalnost prototipa. Horizontalni prototip obično je evolucijski odnosno postepeno se unaprjeđuje u finalni proizvod.



Slika 7. Horizontalni i vertikalni prototip

Izvor: Jakob Nielsen Usability Engineering, 1993, http://www.nngroup.com/articles/guerrilla-hci/.

Vertikalni prototip omogućava dizajneru mogućnost pristupanja funkcijama cijelog sistema od sloja sučelja do sistemskog sloja. Ove funkcije pružaju mogućnost da se podaci unose i spremaju u bazu podataka, mogućnost prikaza podataka i korištenje sučelja. Vertikalni prototip se najčešće radi kako bi se procijenila izvedivost određene značajke u horizontalnom razvoju. Vertikalni prototip u pravilu je prototip visoke preciznosti jer je njegov cilj procjena ideje na razini cijelog sistema. Obično se izrađuju u ranom stadiju prije nego je donesena odluka o izgledu sveukupne arhitekture projekta, fokusiraju se samo na jedno pitanje vezano uz dizajn i nakon toga se odbacuju.

2.2. Prototipski pristup

Prototipski pristup razvoju sustava može se gledati kao „proces“ koji je dobro definiran unutar životnog ciklusa razvoja sustava ili kao pristup koji utječe na cjelokupan razvoj. Ovakav pristup potiče učinkoviti razvoj aplikacija razbijanjem kompleksnog i često loše definiranog problema na nekoliko razumljivih, manjih i jednostavnijih dijelova. Prototipski pristup razvoju pomaže nam pri izgradnji i naknadnom usavršavanju proizvoda kako bi bio zadovoljavajuće kvalitete za krajnjeg korisnika. Model informacijskog sustav se brzo izrađuje kako bi se testirale ili ilustrirale značajke dizajna i ideje, kako bi se dobile povratne informacije korisnika. Novi modeli se izrađuju rafiniranjem starije verzije, s ciljem da se izradi što bolji model koji bi bio prihvatljiv kao finalni proizvod. Različite verzije prototipa se koriste u dizajniranju softvera, razvoj sustava ili drugim granama koje imaju korist od

prototipskog razvoja. Razvoj sustava koji uključuje prototipski pristup temelji se na istraživanju, eksperimentiranju i evoluciji.

Istraživački pristup se uglavnom koristi da se istraže i razjasne korisničke potrebe. Pomaže razvojnom timu da steknu uvid u radne zadatke i probleme s kojima se korisnik susreće, te pomaže razbistriti nejasnu korisničku percepciju i potrebu u zahtjevima za inicijalni sustav. Često je potrebno istražiti zahtjeve i određene detalje prije ili tijekom razvoja informacijskog sustava. Tu se ističu dva pristupa koja uključuju ovakav razvoj, to su: brzi razvoj (rapid development) i spiralni model. Brzi razvoj omogućava nam da saznamo kompletne potrebe korisnika. Model koji se radi tijekom ove faze je izrađen brzo i prljavo ("quick and dirty"). Slabo razumljivi zahtjevi se implementiraju prvi. Dok se prototip razvija čimbenici kvalitete kao što su učinkovitost, prenosivost, cjelovitost, održavanje i dokumentacija se ne gledaju.

Na osnovu tog prototipa korisnici zatim pružaju povratnu informaciju koja se zatim koristi za poboljšanje karakteristika sustava. Nakon što se postigao dogovor između korisnika i razvojnog tima o potrebama sustava ide se u dizajn i implementaciju sustava. prototip se u pravilu odbacuje iako se ponekad ponovno upotrijebiti. Brzi prototip (rapid prototype) poznat kao "odbacujući prototip" (throw-it-away prototype), može biti korišten u bilo kojoj fazi razvoju sustava ali najčešće se koristi u fazi elimininiranja i analize ili dizajna i izrade.

Spiralni model pokušava upravljati rizikom korištenja prototipa u razvojnom procesu informacijskog sustava. Kod spiralnog modela koriste se i termini kao što su pravi prototip (prototype proper) ili pilot prototip (pilot prototype).

Proces razvoja informacijskog sustava započinje konceptualnim dokumentom i napreduje u ciklusima do programiranja individualnih programa u spiralnom modelu. Svaki ciklus sadrži isti redoslijed s koracima: utvrđivanje zahtjeva, ograničenja i alternative koje rješavaju rizik korištenja prototipa, simulacija.

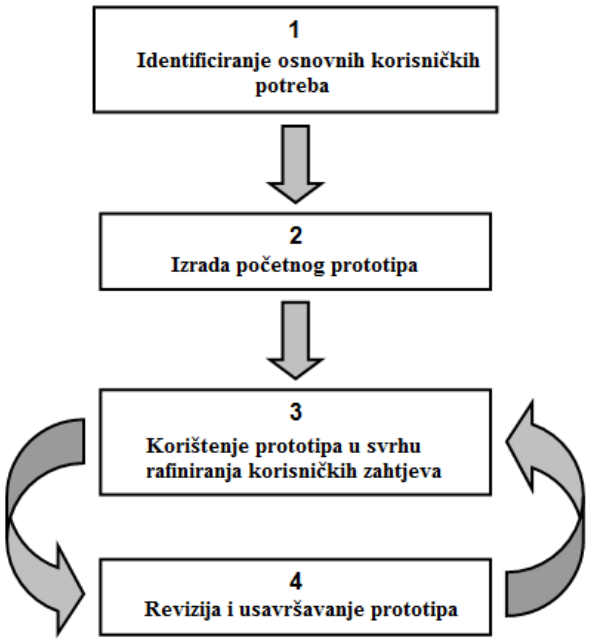
Eksperimentalni pristup pomaže pri određivanju izvedivosti budućeg sistema vezanih uz izvedbu i druge tehničke aspekte. Kod ovog pristupa , prototipi su izgrađeni tako da izvedivost predloženih rješenja se može ispitati pomoću eksperimenta. Prototip može biti djelomična funkcionalna simulacija koja pokazuje određene aspekte sustava, puna funkcionalna simulacija koja ima dostupne sve funkcije sustava ili maketa sučelja, kostur programa koja prikazuje strukturu sustava. Ovakav pristup razvoju potiče kreativnost i brze promjene u razvoju.

Evolucijski pristup omogućava fleksibilnost u procesu razvoja sustava. Ovdje se gleda proces razvoja sustava kao nešto što se kontinuirano razvija a ne kao izolirani samostalni projekt. Cilj ovakvog pristupa je da se razvije sustav kroz seriju ponavljanja (iteracija), koji se

svakim ponavljam treba približiti ispunjavanju korisničkih zahtjeva. Koristi se postepeni razvoj i dinamički reagira na promjene u korisničkim potrebama i naknadnim nepredvidivim promjenama zahtjeva. Ovakav pristup je jako koristan kad se u startu ne može definirati točne zahtjeve. Kao prednosti ovakvog pristupa može se navesti: ubrzana izrada sustava, korisnik se uključuje u razvoj, sustav će vjerojatnije zadovoljiti korisničke zahtjeve.

2.3. Proces izrade prototipa (prototyping process)

Proces izrade prototipa se uglavnom sastoji od serije aktivnosti, faza u kojima se raspravlja s korisnicima o unaprjeđenju karakteristika modela i onda se zatim implementiraju u postojeći model, prototip koji je započeo jednostavno raste u kompleksnosti. Svi sudionici u projektu su uključeni u sve razine razvoja od početka do kraja. Cilj takvog razvoja sustav je da se lakše shvate zahtjevi korisnika, minimiziraju nesporazumi i propusti.



Slika 8. Proces prototipiranja

Izvor: Davis, G. B. and M. H. Olson, Management Information Systems: Conceptual Foundations, Structure, and Development, McGraw-Hill, 1985.

U ovakvom razvoju jako je naglašena fleksibilnost pri implementaciji promjena do finalnog sustava. Na prvi pogled ovakva ideja izrade prototipa ne izgleda efikasna jer se "naizgled" mnogo vremena provodi u razgovoru, diskusiji o modelu i preradi ranije verzije. Tajna leži upravo u tim pripremama i postepenom razvoju. Pošto su modeli uglavnom odbacujući

naknadna verzija je izgrađena na dobrim dijelovima modela dok su loši odbačeni i rijetko se događa da je potrebno početi iznova. Koraci koji se koriste u razvoju prototipa omogućavaju da se: procjenjuje izvedivost, odlučuje o cilju projekta i njegovim glavnim cjelinama ili kategorijama, odaberu se početne cjeline za razvoj, izrađuje se preliminarni dizajn na papiru, kartonu ili specijaliziranom softveru, razgovara se s korisnicima o unaprjeđivanju dizajna. Ove korake se ponavlja dok osnovni dizajn ne postane zadovoljavajući za korisnike, klijente, on se zatim implementira i ponovo se ponavlja korake. Tako se prototip, model konstantno unaprjeđuje. Postoji nekoliko važnih aspekata koje treba gledati za vrijeme procesa: trebaju se uzeti u obzir mišljenja svih korisnika, razgovori se trebaju redovito održavati, definirati tko odobrava dizajn, odrediti duljinu aktivnosti, faza razvoja, konstantno tražiti poboljšanja, ne bojati se grešaka ili iznošenja čudnih rješenja, ne zazirati od promjena prethodno donošenih odluka ako se nova ideja čini boljom, ne se uvlačiti u detalje pri početnim fazama razvoja, paziti na vrijeme razvoja, razviti sustav koji je jednostavan za korištenje.

Odabir odgovarajuće duljine aktivnosti , faze razvoja je dosta važan kako bi se lakše rukovodilo projektom. Vremenski duga aktivnost koja uključuje previše detalja ili veći broj kompleksnih "teških" detalja može dovesti do sporog napretka u razgovorima s korisnicima i razvoju. Aktivnosti koje su pre kompleksne, detaljne mogu dovesti do gubitka vremena i resursa te povjerenja korisnika koji se pozivaju na konzultacije zbog malih promjena ili detalja.

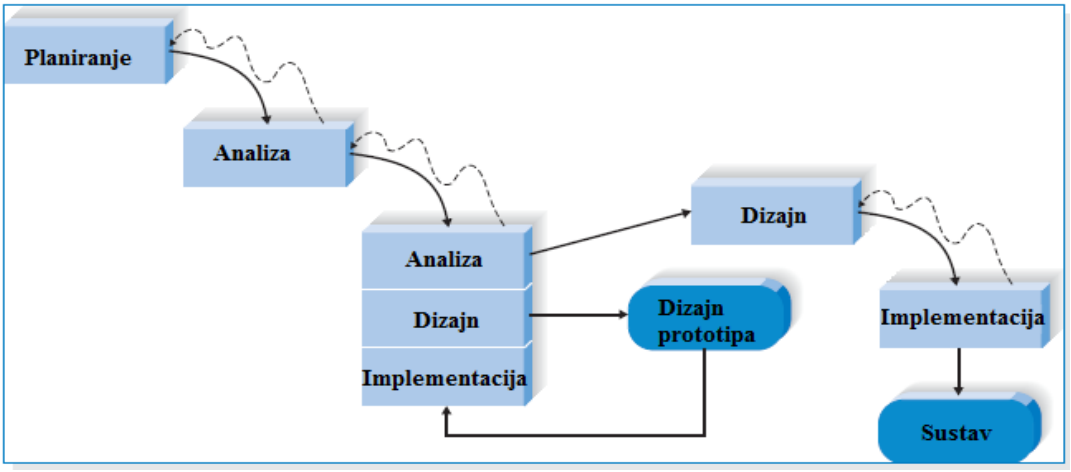
3. Metode prototipskog razvoja i vrste prototipa

Razvojem tehnologija i sve većim potrebama za informacijskim sustavima stvorila se i potreba za bržim, efikasnijim i jeftinijim metodama razvoja sustava. Po procjenama stručnjaka produktivnost programera i dizajnera sustava se povećala u veličini od nekoliko stotina puta u odnosu na vrijeme prije 50 godina. Takav skok u produktivnosti omogućile su nove metode prototipskog razvoja i alati koji su izrađeni na njihovoj osnovi. Zbog brze promjene tehnologija i zahtjeva tržišta informatički sustavi trebaju biti izrađeni brzo, biti konstruirani tako da ih se lakše nadograđuje, implementira u nova okruženja. Zbog toga novije metode razvoja prototipa aktivno uključuju sve sudionike kako bi se razvio efikasan, stabilan i cjenovno prihvatljiv informacijski sustav koji će biti prikladan poslovnim izazovima koje korisnik, organizacija ima pred sobom.

Modeli s prototipskim pristupom postepeno su zamijenili stariji, tradicionalni vodopadni (waterfall) model. Kako bi se pokušalo zaobići nedostatke vodopadnog modela predlažu se novi pristupi razvoju softvera, sustava kao što su "rapid throwaway prototyping", "evolutionary prototyping", "incremental development". Razvojni modeli bazirani na prototipskom razvoju se obraćaju nedostacima vodopadnog sustava prilagodljivim pristupom razvoju, procesom ponavljanja (iterative, spiral process) odnosno procesom u kojem se jednostavniji model kontinuirano poboljšava prema željenom cilju.

3.1. Brzo ili odbacujuće prototipiranje “rapid, throwaway prototyping”

Brzo, odbacujuće prototipiranje uključuje izradu funkcionalnog modela u vrlo ranoj fazi razvoja nakon relativno kratke analize. Kod brzog prototipiranja najvažniji je faktor brzine izrade modela koji zatim postaje početna točka iz koje klijenti, korisnici mogu preispitati svoje zahtjeve i pročistiti ih. Brzo prototipiranje pomoću odbacujućeg modela proširuje proces analize s namjerom smanjivanja cijene sveukupnog životnog ciklusa, pojasniti zahtjeve, prikupiti dodatne informacije. Prototip je napravljen na osnovu obrisa ili sažetka željenih specifikacije sustava. Izrađuje se za istraživanje i modificira se dok postigne zadovoljavajuću funkcionalnost koju zahtjeva korisnik. Nakon što je napravljen prototip koji zadovoljava potrebe korisnika izvlače se specifikacije koje proizlaze iz prototipa i implementiraju se u završni model koji će predstavljati produkcijsku verziju. Prototip se u pravilu odbacuje ali može se i dalje koristiti kao sustav za kontrolu kvalitete.



Slika 9. Odbacujući (throwaway) prototipi

Izvor: Alan Dennis, Barbara Haley Wixom, Roberta M. Roth, Systems Analysis and Design, 5th edition, 2012, str 56.

Vršenje izmjena rano u razvoju sustava vrlo je isplativo a najočitija prednost ovakvog razvoja je brzina kojom se može izraditi model. Brzina je ključna u implementaciji odbacujućeg prototipa kako bi se postigao cilj smanjenja troškova, zbog same činjenice da je neisplativo potrošiti puno resursa na nešto što će se odbaciti. Još jedna pozitivna strana ovakvog razvoja je mogućnost izrade modela koji klijent, korisnik može testirati. Korisničko sučelje je ono što klijent, korisnik percipira kao sustav i mogućnost interakcije sa njim omogućava mu uvid u funkcioniranje sustava. Iako završni odbacujući prototip ima zadovoljavajuću funkcionalnost nije primjeren kao produkcijska verzija. Razlog tome su karakteristike sustava (performanse, pouzdanost, sigurnost) koje su možda bile ignorirane u procesu brzog odbacujućeg razvoja. Naknadno će se te karakteristike teško implementirati, a i sam pokušaj poništava pozitivne učinke ovakvog razvoja jer povećava troškove razvoja i isplativosti sustava.

Ovakvi prototipi se mogu podijeliti na osnovu svoje točnosti prikaza sustava, odnosno: izgleda, interakcije i podešavanja. Dijelimo ih na modele male “low fidelity” i velike “high fidelity” točnosti. Točnost prototipa opisuje koliko lako se prototip može razlikovati od finalnog proizvoda i mogućnost da se naglasi određeni aspekt dizajna.

3.1.1. Prototip male točnosti (Low fidelity)

Dobar primjer metode izrade modela male točnosti “low fidelity” je kartonski, papirnati prototip koji se kreira pomoću olovke, kartona, papira i oponaša izgled sustava. Izrada ovakvog prototipa je vrlo jednostavna i jeftina, također ne zahtjeva nikakvu posebnu stručnost.

Slika 10. Low fidelity papirnati prototip (Rani prototip Nintendo wii U)
Izvor: http://chiefdisruptionofficer.com/helpful-rapid-prototyping-methods-and-tools-to-bring-digital-ideas-to-life-fast/

Ovakav prototip omogućava dizajnerima i korisnicima da vide kako će sustav izgledati na hardverskoj bazi i pomaže da se otkloni većina problema u dizajnu sučelja sustava. Trošak izrade ovakvih prototipa je vrlo mali i zbog toga su vrlo dobri za testiranja alternativnih dizajna i brze iteracije. Vjerojatnije je da će korisniku biti lakše izraziti svoje želje i razmišljanja nego kod prototipa visoke točnosti zbog činjenice da nije potrebna nikakva posebna stručnost ni dodatni troškovi da korisnik sam modificira prototip i tako izrazi svoju viziju sustava.

3.1.2. Prototip visoke točnosti (high fidelity)

Prototip velike točnosti približava dizajnera i korisnika najbliže što se može pravom izgledu finalnog sustava. Korištenje programskih alata za izradu grafičkog korisničkog sučelja (GUI) primjer je izrade modela velike točnosti koji izgleda kao željeni sustav ali mu nedostaje funkcionalnost.

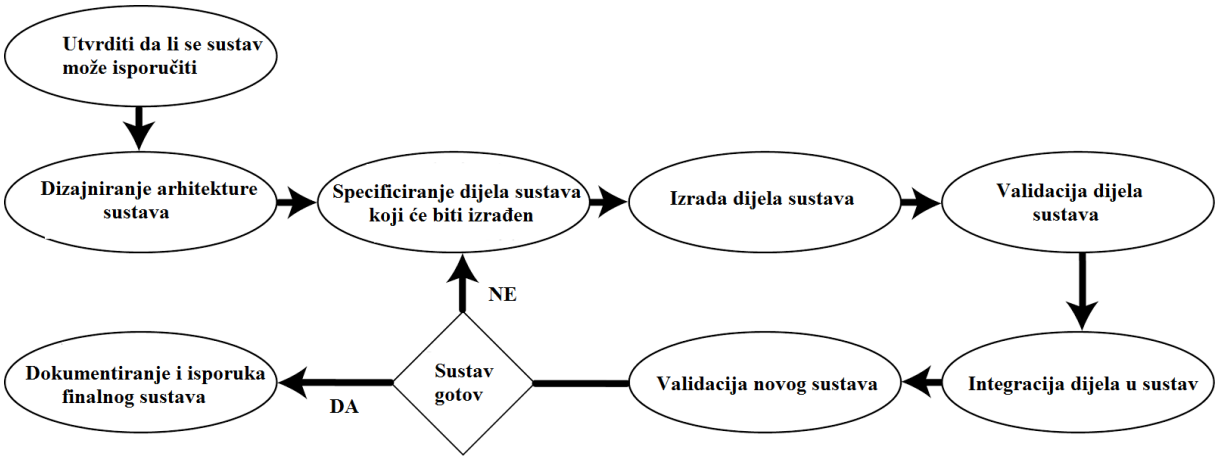
Slika 11. High fidelity web prototip, GUI.

Izvor: http://prototypesapp.com/

Ovakvi softverski alati su široko dostupni a mogu se naći i u web izvedbi. U velikoj većini dopuštaju interakciju sa mišem, tipkovnicom. Pretpostavlja se da prototip visoke točnosti pružaju višu efikasnost u sakupljanju podataka o aktivnostima korisnika i realnom prikazu sustava klijentima .

3.2. Postupni (inkrementalni) razvoj

Kod inkrementalnog razvoja imamo kombinaciju evolucijskog prototipiranja s kontrolom koju zahtjeva veliki projekt. Razvoj informacijskog sustava se vrši inkrementalno odnosno po dijelovima. Primjenom inkrementalnog razvoja nastoji se izbjeći konstantna promjena koja je prisutna u evolucijskom prototipu. Cjelokupna arhitektura, okvir sustava utvrđuje se rano u procesu razvoja i služi kao baza za daljnji razvoj. Kod inkrementalnog razvoja komponente sustava razvijaju se u dijelovima i unutar utvrđenog okvira sustava, tad se dostavljaju korisniku i potvrđuju. Same komponente sustava i okvir u kojem se razvijaju ne mijenjamo osim u slučaju da su otkrivene greške. Povratne informacije za isporučene dijelove koje dobijemo od korisnika mogu utjecati na dizajn komponenti koje slijede u kasnijem razvoju. Inkrementalni razvoj je lakši za upravljanje u usporedbi s evolucijskim pošto se slijede normalni standardi razvoja sustava.



Slika 12. Inkrementalni razvoj

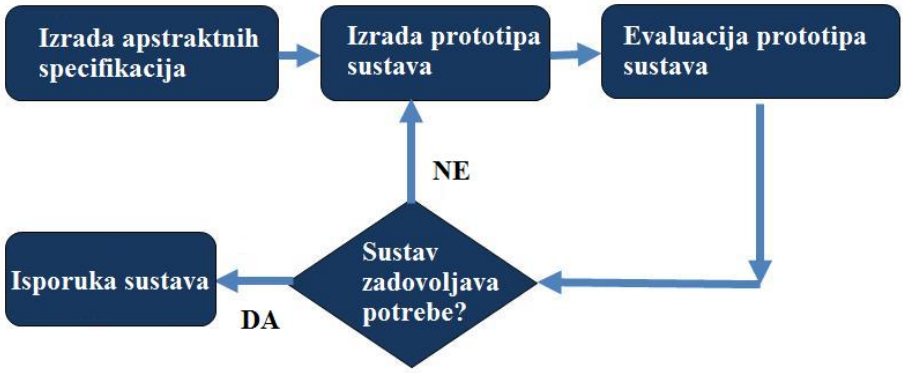
Izvor: http://sce.uhcl.edu/helm/REQ_ENG_WEB/My-Files/mod4/Software_Prototyping.pdf

Dokumentacija i plan razvoja mora se izraditi za svaki korak i pružaju se korisniku na uvid. Tako se dobiva mogućnost da se rano u razvoju dobije povratna informacija od korisnika i ispravi zahtjeve sustava ili greške. Također odvajanjem razvoja po dijelovima, razvojni tim se ne mora brinuti o interakciji totalno različitih dijelova sustava u početnom razvoju. Tek nakon što su svi inkrementi dovršeni adaptiraju se sučelja.

Problem na koji nailazimo kod inkrementalno razvoja je arhitektura sustava koja mora biti uspostavljena prije nego su zahtjevi sustava potpuni. Tako da su naknadne promjene zahtjeva ograničene uspostavljenom arhitekturom sustava.

3.3. Evolucijsko prototipiranje (evolutionary prototyping)

Evolucijski razvoj prototipa bazira se na ideji razvoja početne izvedbe sustav koji se zatim predstavlja korisniku (klijentu) koji ga zatim komentira i ocjenjuje, dobivene povratne informacije zatim se koriste za usavršavanje sustava, to se ponavlja dok se ne zadovolje potrebe korisnika i zatim postaje krajnji proizvod. Ovaj pristup dinamički reagira na promjene u korisničkim potrebama i naknadnim nepredvidivim promjenama zahtjeva. Evolucijska metoda razvoja jedan je realan način da se razvije sustav gdje je teško ili nemoguće utvrditi detaljne specifikacije sustava. Uspjeh evolucijskog pristupa je u korištenju tehnika koje omogućuju brza spiralna ponavljanja (iteracije) sustava. Predložene izmjene moraju se brzo inkorporirati i demonstrirati. To se postiže korištenjem programskih jezika visoke razine koji se koriste za razvoj sustava i aplikacija, posebnim razvojnim okruženjima, integriranim softverskim alatima.



Slika 13. Evolucijski razvoj

Izvor: http://crackmba.com/evolutionary-prototype-model/

Bitna razlika između evolucijskog razvoja i pristupa baziranim na specifikacijama je u verifikaciji i validaciji sustava. Verifikacija ima smisla samo kad je sustav uspoređen sa zahtijevanim specifikacijama. Validacijski proces je važan jer demonstrira da je sustav primjeren za predviđenu namjenu a ne samo da tehnički zadovoljava tražene specifikacije.

Kako bi se smanjio rizik razvojni tim neće implementirati funkcije koje slabo razumije, iako mu možda nedostaju sve funkcije iz željenih specifikacija može se koristiti. Kroz korištenje sustava korisnik može doći do novih zaključaka i zahtijevati izmjenu specifikacija sustava.

Neki od nedostataka ovakvog načina prototipskih modela:

Postojeće strukture za upravljanje postavljenje su da se bave procesima modela koji generiraju redovite izvještaje za procjenu napretka.

Prototip obično brzo evoluira tako da nije isplativo proizvoditi veliku količinu sistemske dokumentacije i rasporeda što kasnije može uzrokovati nedostatak dokumentacije.

Kontinuirane promjene prototipa povećavaju šanse da dođe do korupcije prototipa sustava, što dovodi do teškog i skupog održavanja budućeg sustava. Ove poteškoće posebno su izražene ako održavanje neće obavljati originalni razvojni tim.

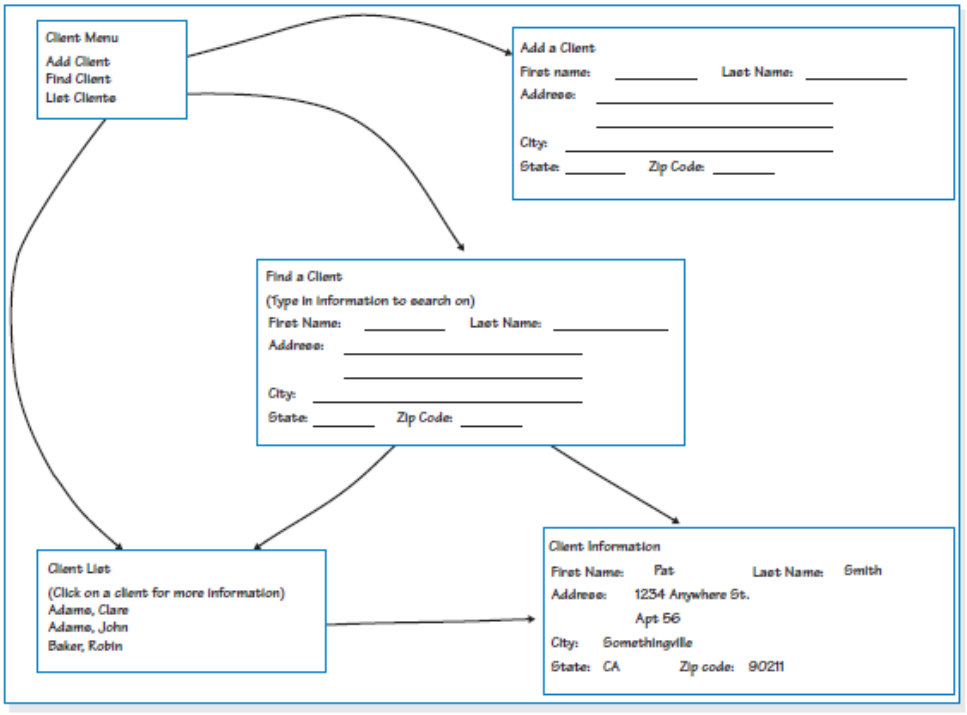
Organizacije koje namjeravaju koristiti evolucijsko model za razvoj sustava moraju uzeti u obzir da je moguće kako će životni vijek sustava biti kratak i da će s vremenom struktura sustava postati pre skupa i teška za nadograđivanje.

4. Alati

Prototipski pristup razvoju sustava trebao bi omogućiti brzi razvoj prototipa sustava. Osim modela razvoja sa uključenim prototipskim razvojem tu nam pomažu alati kao što su: jednostavni alati karton, papir i olovka, visoko razvijeni jezici, CASE alati koji sadrže mnoge manje alate. Ovi alati međusobno se ne isključuje i mogu se koristiti u kombinaciji.
Neki od softverskih alata koji se koriste u svrhu izrade prototipa:

- Softverski alati za crtanje kao FluidIA, Gliffy Online, Napkin Look & Feel
- Alati za animaciju i prezentaciju kao što je Microsoft Power Point, Autodesk
- CASE alati kao Oracle Designer
- RAD alati kao Optima++, Borland Delphi, and Visual Basic
- Jezici visoke razine 4GLs kao FoxPro, IBM Rational EGL, DataFlex (Very High Level Languages) kao Lisp, Prolog, Smaltalk.

Najbrži a mogli bi reći i najjednostavniji alat za izradu prototipa je ***papir i olovka***. U ovaj oblik izrade prototipa može se uključiti i razne prozirne folije i naljepnice koje se naknadno mogu slagati kako bi lakše kreirali, simulirali interaktivni model sustava. Ovim alatom dizajneri sustava mogu jednostavno i brzo izraditi model sustava koji će biti predstavljen klijentu, korisniku. Modeli mogu varirati od vrlo jednostavnih npr. skice do kompleksnih koji uključuju više prozirnih folija sa oznakama koje simuliraju interakciju korisnika sa sustavom. U početnom stadiju izrade modela cilj je izraditi paletu ideja i predstaviti ih bez odabira finalne solucije. Prototipi izrađeni papirom i olovkom su vrlo dobri za izradu horiznontalnog prototipa. Izrada kartonskog modela (cardboard model) omogućava izradu 3D prototipa. Ovakav način izrade prototipa vrlo je čest u industriji a koriste ga i dizajneri softvera kako bi dobili predodžbu kako će njihov sustav funkcionirati u 3D okruženju. Uglavnom izrađeni od kartona i predstavljaju fizičke prototipe sustava, ali često se koriste i softverski alati koji prikazuju model u 3d obliku na računalu.



Slika 14. Storyboard

Izvor: Alan Dennis, Barbara Haley Wixom, Roberta M. Roth, Systems Analysis and Design, 5th edition, 2012 str 329.

Takav je primjer storyboard-a, prikazan na slici 14, koji omogućava dizajnerima da se fokusiraju na pozicioniranje određenih funkcija i njihovu funkcionalnost na hardverskom nositelju, npr. pozicija tipki na ekranu. Može se izraditi više ovakvih modela i dati korisniku kako bi se utvrdili potencijalni problemi u radu i zatim generirati nove ideje za poboljšanje dizajna.

Jezici visoke razine su programski jezici koji omogućavaju razvoj programa u puno jednostavnijem programskom kontekstu i generalno su nezavisni od hardverske arhitekture. Dizajnirani su tako da omogućavaju lakše programiranje i implementaciju, oslanjaju se na snažne jedinice za upravljanje podacima. Na taj način pojednostavljuje se razvoj programa smanjivanjem problema koji se pojavljuju kod nedostatka skladišnog mjesta i upravljanja.

Jezike visoke razine može se podijeliti na:

Jezike 3. generacije (3GL) FORTRAN, BASIC, Pascal, C++, koristi uvjete i petlje tako da jedna linija koda može izvesti više linija objekta odnosno strojnog koda. Mogu biti nezavisni od platforme pa se kod može upotrijebiti i na drugom sustavu što uvelike olakšava razvoj, testiranje i implementaciju sustava.

Jezici 4. generacije (4GL) SQL, NOMAD, FOCUS dizajnirani su da olakšaju programiranje i smanje vrijeme potrebno za razvoj sustava i važna su komponenta u izgradnji prikladnog

prototipa sustava. Njihova uspješnost bazira se na velikoj povezanosti između aplikacija za obradu podataka. Aplikacije su zadužene za osvježavanje i izradu izvještaja na temelju informacijama koje su unesene u bazu podataka, koriste standardne formate za unos i izlaz. Programi 4. generacije mogu se koristiti u CASE alatima za razvoju sustava koji su male ili srednje veličine. Korištenjem programa 4. generacije pridonosi maloj cijeni razvoja što je vrlo izraženo kod malih sustava, ali zahtijevaju puno više memorijskog prostora i sporiji su od konvencionalnih razvojnih programa. Kod ovakvih programa nije prisutna standardizacija a posljedica toga može biti dodatni trošak u budućnosti jer nadogradnja nije moguća zbog zastarjelosti programa u kojem je sustav izrađen. Samim time vidljivo je da nije adekvatan za izgradnju velikih sustava i sustava koji bi se trebali održavati duži vremenski period.

Jezici 5. generacije LISP, PROLOG, MERCURY su programski jezici poznati pod nazivom „naturalni programi“. Baziraju se na rješavanju problema korištenjem ograničenja danih programu i prepuštaju računalu da riješi problem bez programera. Jezici 5. Generacije uglavnom koriste umjetnu inteligenciju za rješavanje zadanih problema. Ovakvi programi koriste vizualno grafičko sučelje koje je dobilo naziv „vizualno programsko okruženje“ koje omogućava izradu koda i pružaju mogućnost interakcije sa računalom i programiranja bez posebnih programskih znanja. Vrlo su pogodni za izradu prototipa, testiranje i olakšavaju komunikaciju između razvojnog tima i korisnika.

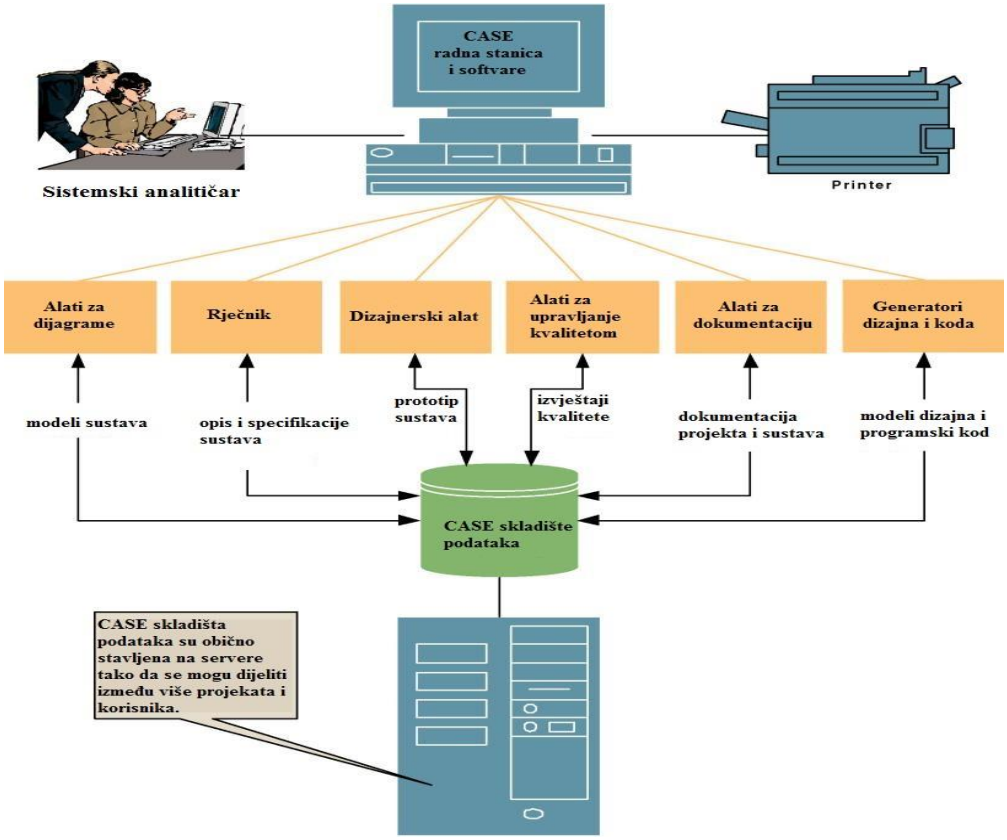
Unutar informacijskih tehnologija ***CASE alat*** (computer-aided system engineering) je softverski alat koji pomaže u izgradnji softverskih sustava i izradu prototipa sustava.

Neki **viši CASE (upper CASE)** alati imaju isključivo ulogu da pomognu kod procesa analize, izrade integriranih dijagrama sustava i skladišti informacije o komponentama sustava.

Niži CASE alati su namijenjeni za dizajnersku fazu, kreiraju dijagrame i onda generiraju kod za tablice baze podataka i funkcionalnost sustava.

Integrirani CASE alati sadrže funkcionalnost viših i nižih CASE alata.

Glavna uloga ovakvog alata je izgradnja programskog koda bez grešaka, lakog za održavanje, te brz i jeftiniji razvoj. Omogućuje dizajnerima, programerima, ispitivačima sustava, menadžerima da dijele mišljenja o projektu u svakoj njegovoj fazi i ujedno se može grafički prikazati detalje razvoja.



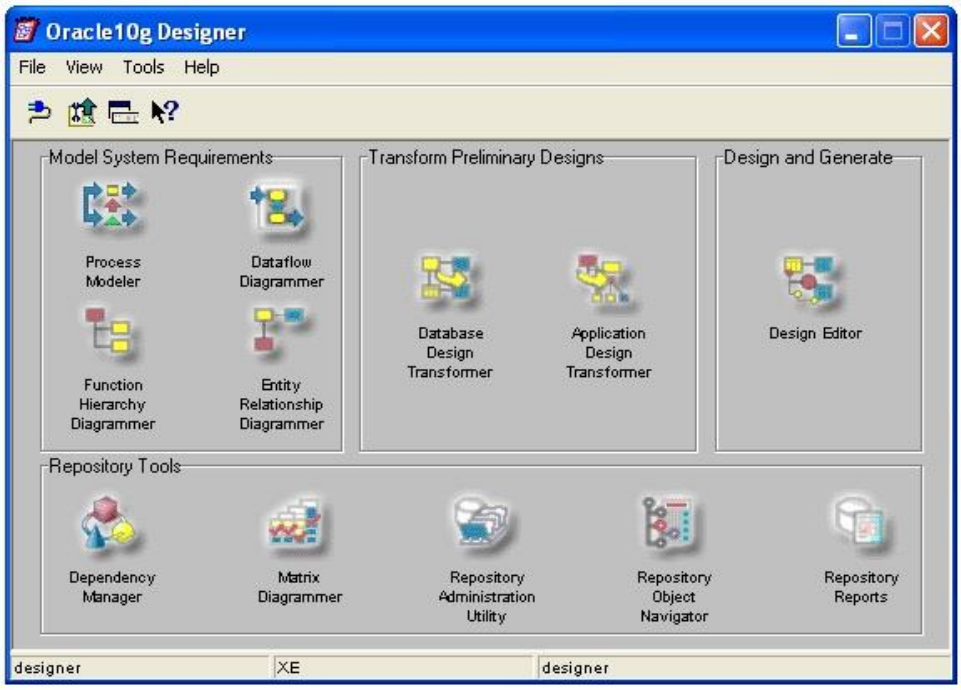
Slika 15. Arhitektura CASE alata

Izvor: Jeffrey L. Whitten, Lonnie D. Bentley, Systems Analysis and Design Methods, 7th Edition, McGraw-Hill, 2010, Str 110.

CASE alati se koriste kroz cijeli ciklus razvoja informacijskog sustava i pokrivaju razna područja: analiza zahtjeva, analiza sustava, dizajn, programiranje, osiguranje kvalitete, dokumentaciju. Centralna komponenta CASE alata je CASE skladište podataka (repository) također poznata kao informacijsko skladište ili rječnik podataka. U skladištu su pohranjene sve projektne informacije kao što su dijagrami, izvještaji, modeli. Ovakav alat koji nudi kombinaciju raznih alata i pokriva sve razine razvoja vrlo je pogodan za velike projekte. CASE alat u sebi sadrži razne alate za izradu prototipa sustava kao što su generatori modela, programskog koda i koriste se u izradi prototipa, implementaciji, testiranju, evaluaciji. Kao primjer takvog program može se navesti Oracle Designer.

4.1. Primjer rada sa Oracle Designer CASE alatom

Oracle Designer CASE alat se bazira na metodama sa kratkim razvojnim ciklusima. Koristi brzi razvoj aplikacija (RAD) i prototipski pristup da bi postigao najbolje rezultate u razvoju informacijskih sustava. Koristeći brzi razvoj aplikacija i integrirani rječnik podataka (repository), dizajn se može izraditi brzo od zahtjeva korisnika u fizičko okruženje prototipa. Projektiranje informacijskog sustava unutar Oracle Designer-a odvija se u cjelinama koje se nazivaju aplikacijski sistemi. Ovakvim pristupom izradi informacijskog sustava može se istovremeno raditi na više aplikacijskih sistema od jednom a rječnik, skladište podataka (repository) olakšava lakše traženje podataka i vršenje izmjena u sklopu projekta. Razvojne module i alate koje sadrži Oracle Designer može se vidjeti na slici 16.

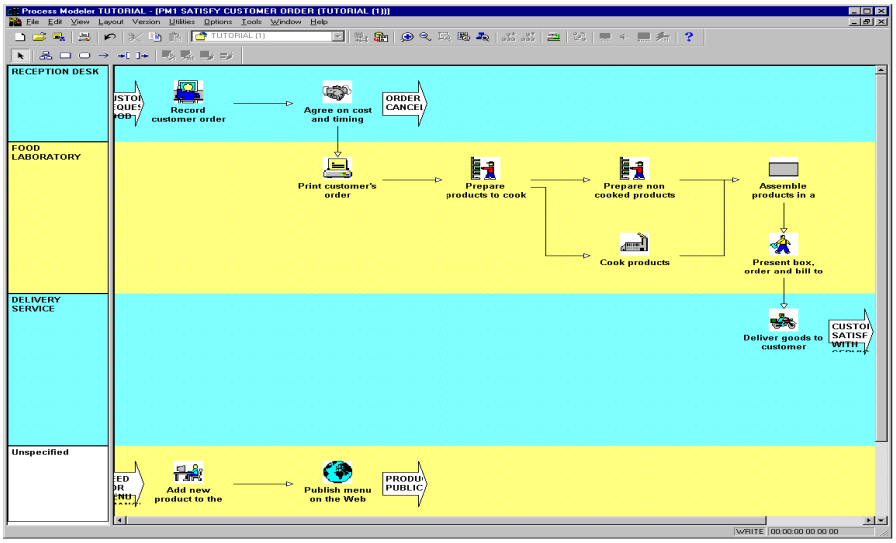


Slika 16. Oracle Designer razvojni moduli.

Izvor: https://pierfinazzi.wordpress.com/2011/06/06/crear-repositorio-para-oracle-designer/

Grafičkim prikazima olakšava se pristup **analizi informacijskog sustava** što se može vidjeti u primjeru **modeliranje procesa** (Process Modeller) na slici 17 koje spada pod modul modeliranje zahtjeva sustava a omogućava prikaz, oblikovanje organizacijske strukture informacijskog sustava. Također omogućava da se dokumentira poslovne procese i tok podataka sa minimumom terminologija za obradu podataka. Može se koristiti za animaciju procesa i za analizu kritičnog puta. Animacija procesa može prikazivati vrijeme potrebno za izvršenje procesa a dijagrami procesa mogu biti kreirani za individualne procese omogućavajući pristup dokumentiranju odozgo prema dolje. Kroz Dataflow Diagrammer

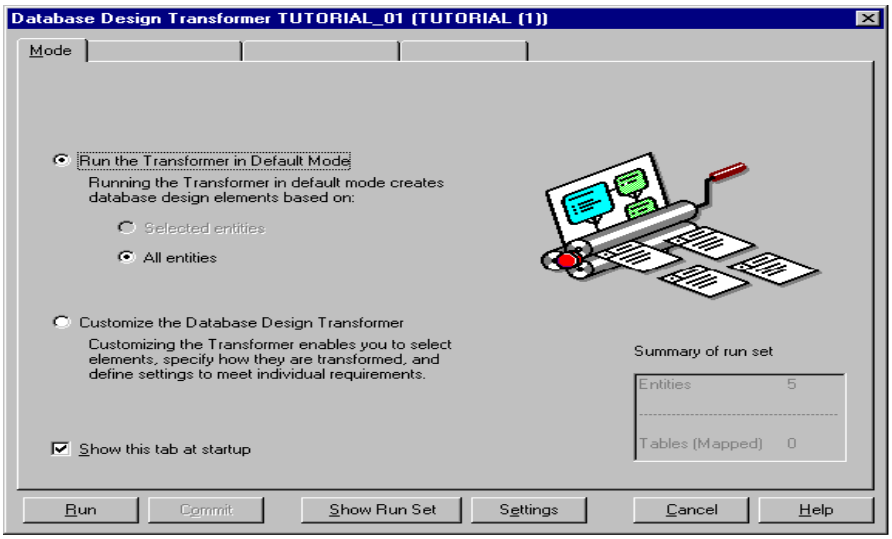
određuje se **tok podatak**, **okidače** i **rezultat**, nakon toga model se sprema u skladište podataka.



Slika 17. Modeliranje procesa (process modeller)

Izvor: http://baks.gaz.ru/oradoc/Designer/tuturial/dsgnr_tutle01_65.htm

Sljedeći korak je kreiranje **entiteta**, **odnosa** između njih i dodavanje **atributa** pomoću Entity Relationship Diagrammer-a.



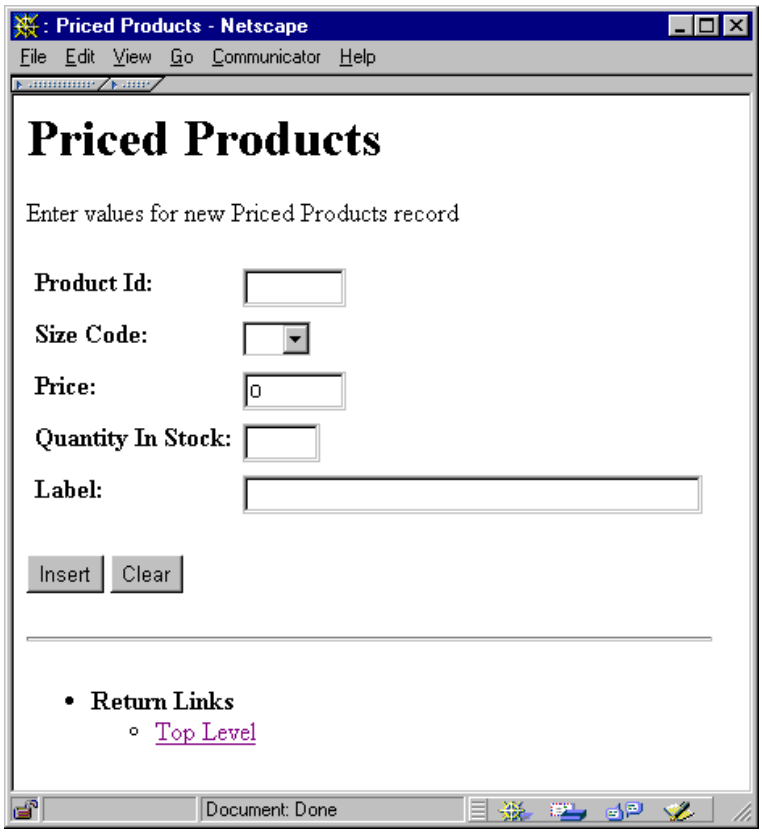
Slika 18. Transformacija modela podataka u tablični dizajn

Izvor: http://baks.gaz.ru/oradoc/Designer/tuturial/dsgnr_tutle04_65.htm

Nakon što je su u modelu određeni entiteti, okidači, odnosi i atributi pomoću Database Design Transformer-a (slika 18) vrši se transformacija, odnosno kreiraju se tablične definicije, dizajn modela baze podataka iz dijagrama entiteta i odnosa. Kada je kreirana prva verzija dizajna

baze podataka mogu se izvršiti dorade, poboljšanja. Pomoću Design Editor opcije generira se serverska verzija komponenti za sustav. Server generator može kreirati objekte na lokalnoj ili udaljenoj bazi podataka iz definicija spremljenih u Oracle Disigner rječniku (repository).

Kad je dizajn stabilan i zadovoljava sve zahtjeve može se krenuti sa izradom modela aplikacije. Oracle Disigner CASE alat omogućava nam izradu dizajna aplikacije kroz modul Application Design Transformer. Izvršava se konverzija i kreiraju se zadane definicije modula, modificiraju se zadani moduli kako bi ih bilo moguće koristiti za generiranje funkcionalnih formi. Naknadno se može modificirati karakteristike formi kao što su padajući meni, pregled po vrijednostima itd.



Slika 19. Web prikaz jednostavnog modela aplikacije izrađene Oracle Designer CASE alatom

Izvor: http://baks.gaz.ru/oradoc/Designer/tuturial/dsgnr_tutle10_65.htm

Nakon što se prođe kroz konverziju pomoću Oracle Web PL/SQL generator-a može se generirati funkcionalnu web aplikaciju u kojoj je moguće pretraživati, mijenjati, dodavati ili brisati informacije.

5. Prednosti i nedostaci prototipskog pristupa

Prednosti prototipskog pristupa:

Pristupi razvoja softvera s uključenim razvojem prototipa postigli su uvaženost pošto su pokazali da imaju mogućnost dinamički reagirati na promjene korisničkih zahtjeva, smanjuju količinu prerada i rizika od nepotpunih zahtjeva korisnika.

Istraživači su također uvidjeli prednost prototipskog razvoja koji omogućava da se proces razvoja raspodijeli u manje segmente razbijanjem kompleksnog i često loše definiranog problema na nekoliko razumljivih, manjih i jednostavnijih dijelova koji su jeftiniji za razvoj, poboljšavaju komunikaciju između osoblja uključenog u razvoj, omogućava tehničku izvedivost, omogućava dobru tehniku upravljanja rizicima i rezultira većim uključivanjem korisnika u razvojni proces.

U dobro vođenom projektu koji uključuje prototipski pristup u razvoju informacijskih sustava nesporazumi između korisnika, klijenata i razvojnog tima identificiraju se kroz prikazivanje prototipa, umjesto na kraju izrade sustava kao kod predvidivih, tradicionalnih metoda razvoja.

Značajke koje nedostaju identificiraju se u početku razvoja a poteškoće u korištenju ili zbunjujuća svojstva mogu se brzo identificirati i riješiti brzo. Nepotpuni i proturječni zahtjevi identificiraju se kod implementacije u prototip i korisnici onda mogu lakše razjasniti svoje zahtjeve. Funkcionalan prototip se može predstaviti kako bi se dokazala isplativost projekta i osigurao daljnji razvoj i financiranje.

Prototip može poslužiti kao baza za razvoj naknadnog potpunijeg i složenijeg sustava što bitno pridonosi manjoj cijeni razvoja potencijalno novog sustava. Informacijski sustavi izrađeni pomoći prototipa uglavnom zahtijevaju manje održavanja i jeftiniji su za naknadnu nadogradnju.

Korištenjem prototipa može se postići brz i efikasan razvoj kvalitetnog informacijskog sustava koji zadovoljava potrebe korisnika, organizacije a samim time povećava se i zadovoljstvo klijenta.

Nedostaci prototipskog pristupa: Nije uvijek lako planirati i upravljati procesom razvoja prototipa, zbog brzih i moguće mnogih ponavljanja teško je definirati koliko ponavljanja (iteracija) će biti u procesu razvoja i koliko dugo će svako ponavljanje trajati. Iz toga se može zaključiti da nije moguće pružiti točnu informaciju klijentu koliko će sami razvoj informacijskog sustava trajati.

Dokumentacija zahtjeva konstantno ažuriranje kako razvoj napreduje. Zbog brzih ponavljanja u procesu razvoja neki pristupi i metode imaju veća mogućnost stvaranja nekompletne dokumentacije.

Od klijenata se traži aktivna uključenost u proces razvoja koji ovisi o njihovim validacijama prototipa. Moguće je da klijenti nisu pripremljeni ili nemaju mogućnosti da podrže takav projekt što dovodi do loših prototipa i odugovlačenja u procesu razvojnih iteracija a samim time i odugovlačenje projekta i skupi razvoj.

Rani prototip koji jasno ocrtava željene specifikacije može podignuti očekivanja klijenta u vezi isporuke konačnog sustava i na osnovu toga klijent može zahtijevati implementaciju takvog ranog prototipa koji nije gotov. Uvjeravanje klijenta da je potrebno doraditi sustavi ili implementacija ranog prototipa mogu biti jednako štetni za povjerenje klijenta.

Upravljanje procesom izrade prototipa ponekad je vrlo zahtjevno i potrebno je dobro organizirati razvojne timove i njihovu interakciju s korisnicima, klijentima kako bi se postigao efikasan razvoj.

6. Zaključak

Veliki broj razvojnih stručnjaka dijeli mišljenje da bi se razvoj pomoću prototipa trebao koristiti uvijek u nekom od oblika. Iako se ističe da od primjene prototipskog pristupa najveću koristi imaju sustavi s mnogo korisničkih interakcija. Što je veća interakcija između korisnika i sustava to je veća korist od primjene razvoja sustava pomoću prototipa, a kao primjer takvih sustava mogu se uvrstiti transakcijske sustave (bankomati), ručna računala itd.

Prototipskim razvojem uvelike se smanjuje mogućnosti greški i propusta u razvoju sustava, a samim time i cijena ispravljanja grešaka i propusta u implementaciji i korištenju sustava. Prototipski pristup razvoju informacijskih sustava omogućavaju konkretni prikaz idejnog dizajna i omogućavaju dizajnerima sustava, razvojnom timu, klijentima, korisnicima, rani uvid u izgled budućeg sustava.

Njegova primjena omogućava evaluaciju i prilagodbu u cilju dobivanja željenih specifikacija i izgleda završnog proizvoda. Odmah, u startu korisnik na temelju prototipa procjenjuje izvedivost projekta u dizajnu i željenom budžetu. Njegova primjena omogućava razvojnim timovima da budu kreativni, te da eksperimentiraju kako bi dobili najbolje moguće rješenje. Zavisno o veličini projekta i resursa može se odabrati odgovarajuća kombinacija modela razvoja i alata za njegovu potporu.

U današnje vrijeme prisutan je konstantan pritisak na produktivnost, funkcionalnost, stabilnost, isplativost informacijskih sustava pa se može zaključiti da je prototipski pristup razvoju informacijskih sustava efikasan onda kad je informacijski sustav opsegom mali, ima veliki broj korisnika, kada je promjenjiv i treba omogućiti jednostavno održavanje i nadogradnju uz male troškove.

7. Literatura

Knjige:

1. Gary B. Shelly, Harry J. Rosenblatt.: „System Analysis and Design 9th Edition“, Shelly Cashman Series, 2012.
2. Davis, G. B. and M. H. Olson.: „Management Information Systems: Conceptual Foundations, Structure, and Development“, McGraw-Hill, 1985.
3. Alan Dennis, Barbara Haley Wixom, Roberta M. Roth.: „Systems Analysis and Design“ 5th Edition“, John Wiley & Sons, 2012.
4. Kenneth E. Kendall, Julie E. Kendall.: „Systems Analysis and Design 8th Edition“, Pearsons, 2011.
5. John W. Satzinger, Robert B. Jackson, StephenD. Burd.: „System Analysis and Design in a Changing World“ 5th Edition, Course Technology, 2010.
6. Jeffrey L. Whitten, Lonnie D. Bentley.: „Systems Analysis and Design Methods“ 7th Edition, McGraw-Hill, 2010.

Internet stranice:

7. Nauman J.D. and Jenkins M., „Prototyping: The New Paradigm for Systems Development“, MIS Quarterly, http://misq.org/cat-articles/protoyping-the-new-paradigm-for-systems-development.html, 28.01.2015.
8. Moe R., Jensen D., Wood K., "Prototype partitioning based on requirement flexibility”, http://www.sutd.edu.sg/cmsresource/idc/papers/2004-_Prototype_Partitioning_ASME_DTM_2004_published.pdf, 28.01.2015.
9. Jakob Nielsen., „Usability Engineering“, http://www.nngroup.com/articles/guerrilla-hci/, 28.01.2015.
10. Helpful Rapid Prototyping Methods and tools to bring Digital Ideas to Life Fast, http://chiefdisruptionofficer.com/helpful-rapid-prototyping-methods-and-tools-to-bring-digital-ideas-to-life-fast/, 28.01.2015.
11. Make Testing Easy, http://prototypesapp.com/, 28.01.2015.
12. Methods for Software Prototyping, http://sce.uhcl.edu/helm/REQ_ENG_WEB/My-Files/mod4/Software_Prototyping.pdf, 28.01.2015.
13. Evolutionary Prototype model, http://crackmba.com/evolutionary-prototype-model/, 28.01.2015.

14. Oracle Designer Tutorial
http://baks.gaz.ru/oradoc/Designer/tuturial/dsgnr_tuttoc_65.htm

8. Popis slika