UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

Nikola Perić

METODOLOGIJA UPRAVLJANJA IT PROJEKTIMA I EKSTREMNO PROGRAMIRANJE

master rad

Beograd,
2012.

UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

METODOLOGIJA UPRAVLJANJA IT PROJEKTIMA I EKSTREMNO PROGRAMIRANJE

master rad

mentor:
dr Gordana Pavlović Lažetić,
redovni profesor

kandidat:
Nikola Perić

Beograd,
2012.

Predgovor

Usled konstantnog rasta korišćenja Interneta, mobilnih i distribuiranih softverskih aplikacija, javlja se potreba da se brzo odgovori na promene i neočekivane izazove u softverskoj industriji. U tom smislu razvija se metodologija upravljanja IT projektima, kao i različiti pristupi u razvoju softvera, među kojima značajnu ulogu ima ekstremno programiranje, kao najpopularniji agilni pristup u razvoju softvera. Stoga je predmet istraživanja ovog master rada primena metodologije upravljanja IT projektima i ekstremnog programiranja.

Ovaj rad se sastoji iz pet poglavlja i dodatka. U prvom poglavlju kroz statističku analizu neuspeha IT projekata, izložena je problematika neuspelih ili delimično uspelih IT projekata. Predstavljeni su rezultati nekoliko istraživanja koja ukazuju da manje od jedne trećine započetih IT projekata završava u okviru planiranog budžeta, vremena i zacrtanih ciljeva. Opisani su i najčešći uzroci neuspeha i predloženo jedno od mogućih rešenja koje bi organizacije trebalo da usvoje da bi povećale uspešnost svojih projekata. Time bi se osiguralo da IT projekti ne propadaju i da duže traju. Drugim poglavljem se objašnjavaju ključni termini iz upravljanja projektima, koje je neophodno poznavati da bi rasprava o rešenjima problema iz ove oblasti bila jasna i bez nedoumica. Prikazane su specifičnosti IT projekata, kao i tehnike i alat metodologije upravljanja IT projektima. U trećem poglavlju dat je detaljan prikaz ekstremnog programiranja. Objašnjen je razvoj ekstremnog programiranja, definisane su vrednosti, principi, prakse i aktivnosti ekstremnog programiranja, a pojašnjena je i upotreba praktičnih preporuka. Takođe, navedeni su alat, primena i proširenja ekstremnog programiranja. Četvrto poglavlje je studija slučaja kojom je prikazan primer dobre prakse razvoja softvera primenom metodologije upravljanja IT projektima i ekstremnog programiranja. U petom poglavlju data su zaključna razmatranja o uspešnoj primeni metodologije upravljanja IT projektima i ekstremnog programiranja. Takođe, nameće se zaključak da bi, iz neuspelih IT projekata koje su drugi doživeli, trebalo izvući pouke za buduće poduhvate. Ne bi trebalo ponavljati iste greške koje su drugi pravili. U Dodatku je kratak vodič za rad u programskom paketu Microsoft Office Project, zvanično najzastupljenijem softverskom rešenju za efikasno upravljanje projektima u svetu. Na kraju ovog rada nalazi se spisak korišćene literature.

Iskreno hvala mojoj mentorki dr Gordani Pavlović Lažetić, redovnoj profesorki Matematičkog fakluteta, Univerziteta u Beogradu, koja mi je stručnom podrškom, korisnim sugestijama, kao i ukazanim poverenjem i strpljenjem, pomogla u realizaciji ovog master rada.

Takođe, hvala uvaženim članovima komisije za odbranu master rada - dr Vladimiru Filipoviću i dr Saši Malkovu, nastavnicima Matematičkog fakulteta u Beogradu, na iskazanoj veri u mene, moje ideje i istraživačke napore.

Iznad svega, neizmernu zahvalnost dugujem mojim roditeljima, Radoslavki i Miroljubu, koji su me podržali i motivisali da istrajem i u ovom poduhvatu.

U Beogradu, decembra 2012. godine Autor

Sadržaj:

Spisak slika:

Spisak tabela:

Roditeljima,
i svima koje volim

1 UVOD

1.1 Statistika i uzroci neuspelih IT projekata

U celom svetu IT (akronim, eng. *Information technology*) projekti i dalje probijaju planirane budžetske i vremenske okvire i na kraju ne isporučuju ono što je planirano. Autor ovog rada analizira uzroke zbog kojih u većini IT projekata nije ostvareno planirano i vraćeno uloženo. Prema rezultatima brojnih istraživanja, IT projekti imaju vise šansi za neuspeh, nego za uspeh. U ovom radu su predstavljeni najčešći uzroci neuspeha IT projekata, bez obzira na oblast poslovne primene. Mudro bi bilo koristiti iskustva drugih, te ne činiti iste greške.

Jedna studija iz 1995. godine koju je napravila *KPMG Consulting* grupa izveštava o zrocima i posledicama neuspeha projekata [37]. Glavni razlozi za neuspeh softverskih projekata:

- ciljevi projekta nisu u potpunosti definisani,
- loše planiranje i procene,
- tehnologija je nova za organizaciju,
- metodologija upravljanja IT projektima nije adekvatna ili uopšte ne postoji,
- u projektnom timu nema dovoljno iskusnih članova.

Ako je suditi po statističkim podacima, šanse za neuspeh IT projekta su veće od onih za uspeh. Brojke govore sledeće: oko 80-90% investicija u IT industriji ne ostvaruje očekivane rezultate, a razlozi su retko tehničke prirode. U proseku oko 80% novih IT projekata je isporučeno sa probijanjem vremenskih i/ili finansijskih ograničenja, od čega je neuspelih proje-kata oko 40%, a toliko je u proseku i delimično uspelih, s tim što je svega 10-20% IT projekata implementirano i zadovoljilo sve kriterijume uspešnosti, podaci su *OASIG*–a iz 1996. godine. Međutim, kada se govori o (ne)uspešnom projektu, treba imati na umu da je uspešan projekat onaj koji je kompletiran na vreme, u okvirima predviđenog i odobrenog budžeta, sa svim funkcijama i karakteristikama koje su prvobitno specificirane. Delimično uspešan IT projekat je onaj koji je kompletiran i operativan, ali je probio okvire finansijskog i vremenskog plana, i ponudio je manju funkcionalnost nego što je bilo planirano. Neuspešan projekat je onaj koji je otkazan u nekom trenutku tokom razvojnog procesa.

Na osnovu podataka istraživanja američkog *The Standish Group - The CHAOS Report*, koji su prikupljeni u periodu od 1994. do 2004. godine, reklo bi se da se situacija izvesno popravila, mada ne mnogo (videti tabelu 1).

Kao bitni uzroci neuspeha IT projekata navode se:

- nejasni zahtevi klijenata,
- neodgovarajuće kontrolisanje promena u projektu,
- nedostatak učešća klijenata u samom razvoju projekata,
- nekontrolisane promene vremenskog plana projekta,
- neodgovarajuće korišćenje tehnika i alata za upravljanje projektima,

- neodgovarajuće upravljanje troškovima projekta,
- neodgovarajuća komunikacija na projektu,
- neodgovarajuće testiranje,
- neodgovarajuće upravljanje rizicima,
- neodgovarajuće osiguravanje kvaliteta,
- neadekvatno planiranje,
- slab osećaj vlasništva nad projektom,
- nekompletna specifikacija,
- nerealna očekivanja,
- drugi ljudski faktori, ...

godine	uspeli IT projekti	delimično uspeli IT projekti	neuspeli IT projekti
1994.	16 %	53 %	31 %
1996.	27 %	33 %	40 %
1998.	26 %	46 %	28 %
2000.	28 %	49 %	23 %
2002.	34 %	51 %	15 %
2004.	29 %	53 %	18 %

tabela 1 - *The Standish Group - The CHAOS Report*

Prilikom diskutovanja o uspelim IT projektima, treba praviti jasnu razliku između uspeha u upravljanju projektima i uspeha projekta u celini. Projekat je uspešan u kontekstu upravljanja projektom ako je završen u zadatom roku, u okvirima propisanog budžeta i ako ima propisane karakteristike. Da bi taj isti projekat bio uspešan u celosti, on mora ostvariti očekivani profit.

1.2 Uspešnost i održivost IT projekata

Više od pola veka organizacije su upravljanje projektima smatrale procesom čija primena može biti korisna za poslovanje, a ne kao proces koji je neophodan za njihov opstanak. To je ujedno bio i osnovni razlog za nedovoljno investiranje u treninge i obuke zaposlenih i njihovo detaljnije upoznavanje sa postojećim znanjima o planiranju, praćenju i kontroli projekata. Upravljanje projektima je uglavnom predstavljalo pretnju već ustaljenim linijama autoriteta. Zbog toga su u mnogim organizacijama praktikovni samo pojedini delovi metodologije upravljanja projektima i tehnika koje se vezuju za ovu disciplinu.

Većina organizacija je živela u lažnom ubeđenju da je vreme luksuz koji mogu sebi da priušte, a ne jedan od ključnih faktora koji ih ograničava. Krajem prošlog veka, ovakav stav je u značajnoj meri oslabio jer se težilo ka stavranju što kvalitetnijih proizvoda i usluga, u što je moguće kraćem vremenskom periodu. Važnost uspostavljanja dugoročnih i pouzdanih odnosa sa klijentima našla se u prvom planu. Time je metodologija upravljanja projektima postala glavno oružje za stvaranje prednosti u nemilosrdnom konkurentskom okruženju.

Ovaj master rad ima za cilj predstavljanje osnovnih teorijskih elemenata metodologije upravljanja projektima i mogućnosti primene odgovarajućih organizacionih i upravljačkih tehnika koje se metodologijom podrazumevaju.

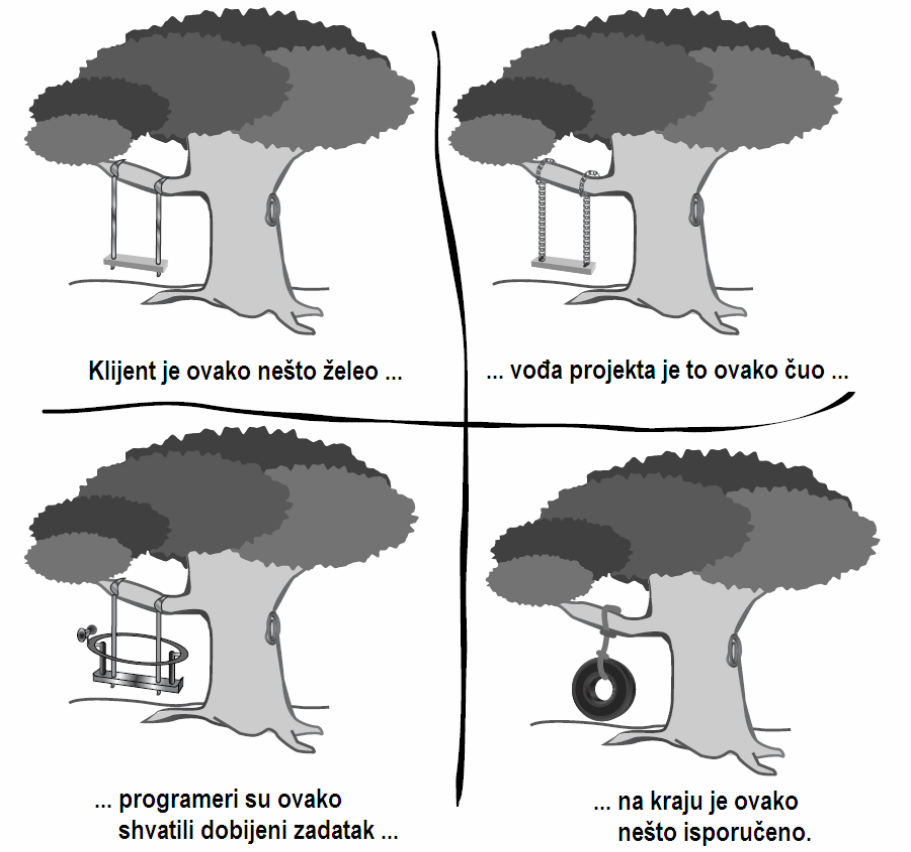
Shvatanje metodologije upravljanja projektima značajno je počelo da se menja pre desetak godina [10]. Međutim, nije se samo pojavila potreba za implementacijom procesa formalnog vođenja projekata, već se sve više uvode rešenja karakteristična za pojedine oblasti u kojima se projekti odvijaju. Metodologija upravljanja projektima, iako se razvila iz tehničkih disciplina, vremenom je pod uticajem drugih oblasti sve više postala multidisciplinarna. Tako da za uspešan rad na celom projektu treba uzeti u obzir, osim uže oblasti upravljanja projektom, pre svega organizacijsku strukturu, okruženje projekta, znanja oblasti primene projekta, standarde i pravne okvire, kao i veštine

poslovnog upravljanja i međuljudskih odnosa. Svaki od tih činilaca može imati veliki uticaj na uspešnost projekta.

Kod organizacionih sistema koji funkcionišu u nestabilnom okruženju promene su stalne. U uslovima brzih promena i visoke neizvesnosti opstaju samo organizacije koje su u stanju, ne samo da brzo reaguju na promene već i da proaktivno iskoriste prednosti promena. Najbitniji i najiskorišćeniji resursi postaju ljudi i vreme koje je ujedno i nenadoknadiv resurs. Agilne metodologije, a pre svih ekstremno programiranje, stavlja poseban akcenat na ljude u projektu i brzo reagovanje na promene, kao i poštovanje ugovorenih rokova. Timski rad i direktna komunikacija, kao i evolutivan pristup dizajnu su njihove osnovne odlike. Razne greške mogu proisteći i iz drugačije vizije koju ima klijent od one koju nameće softver pa su klijenti često nezadovoljni softverom i uglavnom ne cene trud koji se u njega uloži. I ovakvu vrstu problema na odgovarajući način tretira ekstremno programiranje. Razvoj vođen testovima, male i česte isporuke, uključivanje klijenta u razvojni tim i testiranje, kao i drugi principi u ekstremnom programiranju nude sofver optimalnog kvaliteta uz poštovanje rokova i promena zahteva.

Karakteristike dobre metodologije upravljanja projektima su: optimalni nivo detaljnosti, upotreba šablona, standardizovane tehnike planiranja, raspoređivanja i kontrole troškova, standardizovan format izveštavanja, fleksibilna primena u svim projektima, lako za razumevanje i praćenje od strane klijenata, korišćenje standardizovanih faza životnog ciklusa. Više se ne postavlja pitanje treba li primenjivati metodologiju upravljanja projektima, nego:

- Koliko uspešno je moguće upravljati projektima?
- Koliko brzo se dostiže zrelost u upravljanju projektima?
- U kojoj meri je moguće iskoristiti primere dobre prakse kako bi se što pre osetila korist upravljanja projektima?



slika 1 – *Stablo (ne)razumevanja*

Projekat lako može da izmakne kontroli ukoliko ne postoji strategija razvoja koja je otvorena i prilagodljiva. Projekti koji delimično ili potpuno ne uspevaju obično počinju sa dobrim šansama za uspeh, ali se stvari preokrenu u toku realizacije. Razvoj softvera je rizičan posao u kome postoji mnogo neizvesnosti i promena. Postoje i neki opšte poznati rizici, poput:

- Greške u terminu se javljaju tokom rada kada postaje jasno da se ne može postići ugovoreni datum isporuke.
- Prekidanje projekta je čest slučaj kod velikih projekata, u situacijama kada klijenti u projekat ulože ogromna finansijska sredstva, za koje se ispostavi da im se nikada ne mogu vratiti.
- Softversko rešenje nije prihvaćeno, jer se projekat razvijao daleko od klijenata i ne radi onako kako bi oni želeli, što je inače najčešće posledica loše komunikacije i nerazumevanja (videti sliku 1).
- Tehnička složenost se javlja usled neblagovremene provere mogućnosti integrisanja izabranog alata i platformi.
- Da softversko rešenje ima mnogo grešaka, najčešće otkrivaju krajnji korisnici softvera, jer su tokom razvoja greške zanemarene ili nisu uočene, te je kvalitet softvera jako mali. Zbog toga dolazi do parničnih postupaka, ponovnog razvoja softvera ili finansijskih problema.

Ovakvi i mnogi drugi rizici kontrolišu se izborom dobre metodologije upravljanja IT projektima u kombinaciji sa ekstremnim programiranjem. Samo ekstremno programiranje ove rizike smanjuje tako što se u softver ugrađuje kvalitet, a program koji radi se isporučuje u veoma kratkim iterativnim ciklusima. O korisnosti metodologije upravljanja IT projektima i ekstremnom programiranju biće reči u narednim poglavljima ovog master rada. Pre toga, autor ovog rada želi da ukaže kako ne bi trebalo da se kreće u projekat razvoja softvera, o čemu govori sledeća studija slučaja.

1.3 Studija slučaja - Primer loše prakse

Policija Novog Zelanda je 1994. godine dodelila IBM-u posao razvoja projekta pod nazivom INCIS – Integrisani nacionalni računarski informacioni sistem [3]. Planirano je da INCIS bude prvi sistem kriminalističke službe u svetu. On je trebalo da, na nacionalnom nivou, poveže sve aspekte policijskog rada. Posle pet godina i blizu 68 miliona dolara, projekat je odbačen. Budžet je bio premašen za 15 miliona dolara, a rezultata je bilo vrlo malo. Kao što se može očekivati kod katastrofe ovakvih razmera, postojalo je više faktora koji su doveli do neuspeha:

- Projekat je tehnički i operativno bio izuzetno složen, dok je nivo složenosti bio potcenjen. Mogući rizik nije bio uočen.
- IBM je svoje klijentske mašine razvio pod operativnim sistemom OS/2. Zbog dužine trajanja projekta, tehnologije su zastarele, te se 1996. godine prešlo na Windows NT. Cena prelaska i ponovnog rada je bila enormno velika, te su tehnički problemi zahtevali dodatna ulaganja od blizu 4 miliona dolara.
- Vođe projekta su bile preterani optimisti, te nisu reagovali adekvatno na upozorenja projektnog tima.
- Rokovi nisu bili zasnovani na realnim procenama. Politika i marketing su zadali rokove koje nije bilo moguće ostvariti.
- Skoro sav posao je trebalo da se završi pre nego što korisnici budu videli sistem. Ideja je bila da se stari sistem zameni u jednom potezu, što je takođe bilo nerealno.
- Tokom razvoja sistema korišćen je neadekvatan pristup upravljanju i razvoju projekta. Specifikacija funkcija je imala više od 4000 strana, a u toku projekta je bilo preko 900 varijacija. Promena nije bilo sve dok korisnici nisu razjasnili šta zaista piše na tih 4000 strana. Prirodno je da su tokom realizacije projekta zatražene mnoge promene, ali su one bile jako skupe i teške za izvođenje.

Iz ove studije slučaja se jasno vidi da se razvoj softvera mora odvijati u dinamičkom okruženju, koje može da prihvati promene. Sam softver je najbolji oblik specifikacije. Odgovorni za projekat INCIS su napravili greške pomešavši dokumentaciju i komunikaciju. Nemogućnost realne procene realizacije projekta je jedan od najznačajnijih razloga neuspešnosti IT projekta, jer dovodi do problema u upravljanju, loših projektnih planova, kašnjenja s isporukom, prekoračenja proračuna, neefikasnog korišćenja resursa, nezadovoljstva klijenata, lošeg kvaliteta projekta i smanjenog profita [27].

2 METODOLOGIJA UPRAVLJANJA IT PROJEKTIMA

2.1 Istorijski kontekst upravljanja projektima

Razvoj formalnog upravljanja projektima je pokrenut pedesetih godina prošlog veka, kao potreba Ministarstva odbrane Sjedinjenih Američkih Država za razvojem složenih vojnih sistema. Time se potvrđuje činjenica da je metodologija upravljanja projektima nastala iz tradicionalnih inženjerskih disciplina [8]. Najveći uticaj na dalji rast metodologije upravljanja imala je težina projekata unutar različitih inženjerskih zanimanja [10].

Desetak godina kasnije, IT industrija je započela snažan uticaj na poslovna okruženja, što je dovelo do sve većeg korišćenja računara za poslovne svrhe. U tim ranim počecima za sve projekte koristio se isti pristup, koji je bio nasleđen od drugih inženjerskih disciplina. Međutim, brzi rast IT industrije često je za posledicu imao neuspešne projekte, zbog neispunjenih rokova i zbog različitih očekivanja klijenata. Tako se došlo do zaključka da tradicionalni formalni pristup upravljanja nije prikladan za takve projekte.

Paralelno s upotrebom tradicionalnog formalnog pristupa upravljanja počeli su se razvijati i novi pristupi. Nasuprot tradicionalnom inženjerskom pristupu, razvijen je dinamički pristup [6], koji je morao da odgovori na veće zahteve za kontrolom troškova i za bržim ostvarivanjem ciljeva projekta. Kao krajnji korak u razvoju pojavio se ekstremni pristup, kroz koji se vodilo računa o svim bitnim ograničenjima u pogledu vremena, troškova, ljudi, brzine razvoja i neplaniranih iznenadnih promena unutar projektnog plana [6].

Tokom devedesetih godina prošlog veka, došlo je do integracije sledećih procesa u jedinsvenu metodologiju upravljanja projektima [10]:

- upravljanje projektima – osnovni principi planiranja, raspoređivanja i kontrole rada,
- upravljanje kvalitetom – treba da osigura da krajnji rezultat projekta ispuni očekivanja klijenata u pogledu kvaliteta,
- paralelna realizacija – treba da omogući paralelno izvršavanje posla u cilju realizacije zadataka u kraćem vremenskom roku, bez uljkučivanja dodatnih rizika,
- upravljanje promenama – treba da omogući kontrolu rezultata kako bi se obezbedila dodatna vrednost za korisnike,
- upravljanje rizikom – treba da omogući identifikovanje, ocenjivanje i reagovanje na moguće projektne rizike, bez uticaja na ciljeve projekta.

Krajem prošlog veka organizacije su počele da uviđaju da je primena upravljanja projektima više stvar potrebe, a ne izbora. Upravljanje projektima se proširilo gotovo na sve grane privrede, te su najbolji rezultati postignuti u istraživanjima svemira, u vojnoj industriji, građevinarstvu, automobilskoj industriji, telekomunikacijama, IT industriji, zdravstvu [10].

2.2 Pojmovi u upravljanju projektima

Postoji više definicija pojma projekta koje u određenoj meri odražavaju njegovo značenje, a ovde će biti navedene samo neke od njih, koje po mišljenju autora ovog rada najsveobuhvatnije definišu sam pojam. Budući da je većina autora kompetentnih za upravljanje projektima učestvovala na izradi Vodiča za upravljanje projektima (eng. *A Guide to the Project Management Body of Knowledge - PMBOK Guide*) u izdanju američkog međunarodnog Instituta za upravljanje projektima – PMI (akronim, eng. *Project Management Institute*), ili se u svojim radovima pozivaju na njega, definicije iz Vodiča uzete su kao najuopštenije. Standard PMI-a je prihvaćen kao ANSI standard i kao takav se primenjuje u svim njegovim ograncima, kojih ima širom sveta, te i u Srbiji[*]. Osim ovog Instituta, postoji i nekoliko asocijacija koje se bave promocijom upravljanja projektima, od kojih su u Evropi najpoznatije Internacionalna asocijacija za upravljanje projektima – IPMA (akronim, eng. *International Project Management Association*) i Asocijacija za upravljanje projektima – APM (akronim, eng. *Association of Project Management*). Pored PMI-a, i navedene asocijacije obučavaju i sertifikuju vođe projekata.

Prema PMI, projekat (eng. *Project*) se definiše kao određeni vremenski poduhvat usmeren ka stvaranju jedinstvenog proizvoda, usluge ili nekog drugog određenog rezultata [7]. U ovoj definiciji može se uočiti nekoliko bitnih stvari koje razlikuju projekt od svakodnevnih poslovnih rutina. Pre svega, projekat je privremena aktivnost, što znači da projekt ima svoj početak i svoj kraj. Takođe, tim koji radi na određenom projektu okupljen je tokom trajanja projekta, a nakon toga se raspušta. Drugim rečima, ljudski resursi se oslobađaju za druge projekte i poslove. Pored toga, svaki projekat je jedinstveni proizvod.

Međutim, projekat je moguće definisati i kao višefunkcionalan jer predstavlja bilo koji niz aktivnosti i zadataka koji imaju određeni cilj, određene uslove koje treba ispuniti, definisan početak i kraj, ograničena finansijska sredstva, kao i resurse (ljudske i tehničke) [10]. Takođe, projekt je moguće sagledati i kao skup različitih aktivnosti koje se obavljaju u logičkom nizu kako bi se došlo do određenog rezultata [9]. Ovom definicijom se ističe da svaka aktivnost, kao i projekat u celini, imaju definisan početak i kraj.

Imajući u vidu navedene definicije, kao i definicije termina projekat koje su objavljene u časopisima različitih konferencija koje su za temu imale upravljanje projektima, nameće se zaključak da se PMI definicija može smatrati formalnom definicijom projekta. Ovoj formalnoj definiciji projekta, autor ovog rada dodao bi načelo dobrog projekta koje potiče iz prakse, a na osnovu kog se insistira da projekat bude realizovan u predviđenom roku, u okviru planiranog i odobrenog budžeta, sa očekivanim rezultatima kojima će klijenti biti zadovoljni. Stoga se kritičnim elementima projekta smatraju resursi (ljudi, materijali, oprema), vreme koje je predviđeno za realizaciju projekta i sredstva koja su na raspolaganju za realizaciju projekta.

Uzimajući u obzir PMI definiciju projekta kao formalnu i najuopšteniju, upravljanje projektom (eng. *Project Management*) se definiše kao primena znanja, veština, alata i tehnika u projektnim aktivnostima da bi se ispunili projektni zahtevi [7]. Upravljanje projektima obuhvata strogo definisane zahteve, jasno postavljene ostvarljive ciljeve, uspostavljenu ravnotežu između suprotstavljenih zahteva kvaliteta, vremena i troškova, usaglašenost specifikacija, planova i očekivanja različitih zainteresovanih strana u projektu.

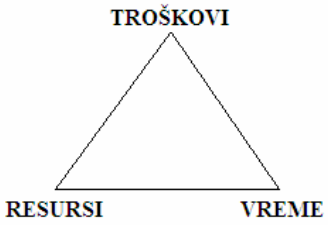
Iako većina autora slično definiše proces upravljanja projektima [4][9][10][12], značajan je i stav da je upravljanje projektima i nauka i veština i umetnost [11]. Veština i umetnost, jer je potrebno voditi ljude na projektu sa ciljem da ljudi nešto novo naprave, a nauka zbog definisanja i koordiniranja samog projekta. Takođe, vrlo je uopštena definicija da je upravljanje projektima skup metoda i tehnika utemeljenih na principima upravljanja planiranjem, procenom i kontrolom aktivnosti u svrhu postizanja željenog cilja na vreme i u okvirima predviđenog budžeta [8].

U većim organizacijama koje odlikuje veliki broj samostalnih, a i međusobno zavisnih projekata, sam proces upravljanja projektima odvija se u širem kontekstu od jednog projekta. Zbog toga se u novije vreme javljaju i pojmovi poput programa i projektnog portfolio menadžmenta, te respektivno tome pojmovi upravljanje programom i upravljanje portfoliom. Pod programom (eng. *Program*) se podrazumeva skup međusobno povezanih projekata organizovanih da realizuju predviđeni cilj koji ne bi bilo moguće ostvariti nekim samostalnim projektima [7]. Projektni portfolio menadžment – PPM (akronim, eng. *Project Portfolio Management*) je skup samostalnih projekata ili

[*] više informacija o Institutu za upravljanje projektima - ogranak Srbija na adresi www.pmi-serbia.rs

programa i sekundarnih aktivnosti koji su grupisani zajedno kako bi se uspešno upravljalo tim projektima u svrhu postizanja strateških poslovnih ciljeva [7]. Drugim rečima, PPM je upravljački proces dizajniran da pomogne da organizacija obezbedi i vidi informacije o svim svojim projektima, da ih zatim sortira i prioritetizuje prema određenim kriterijumima, kao što su strateška vrednost, uticaj na resurse, troškove itd. Projektni portfolio se redovno pregleda (mesečno, kvartalno, ...) od strane portfolio menadžment tima, da bi se odredilo koji projekti zadovoljavaju svoje ciljeve, kojima je potrebna šira podrška, ili koje treba sužavati ili potpuno stopirati i ukinuti.

Vođa projekta ili projektni rukovodilac (eng. *Project Manager*) je odgovoran za upravljanje projektom i postizanje projektnih ciljeva. Vođe projekata često navode da se ključ uspešnog upravljanja projektima i ispunjenja svih projektnih zahteva sagledava u „trouglu ograničenja" koji grafički prikazuje osnovne parametre upravljanja projektima – vreme, troškove i resurse (videti sliku 2.) [8] [11][12]. Kvalitet projekta se postiže adekvatnom ravnotežom ova tri parametra. Visokokvalitetni projekti za ishod imaju zahtevani proizvod, uslugu ili drugi rezultat u zahtevanom obimu, na vreme i u okviru predviđenog budžeta. Odnosi između navedenih parametara su takvi da u slučaju promene jednog od njih, najverovatnije dolazi do promene bar još jednog ili sva tri.



slika 2 - Osnovni parametri projekta prikazani kao „trougao ograničenja"

Osnovno pitanje je na koji način biti dobar vođa projekta? Generalno, pre početka upravljanja projektom, vođa projekta mora imati jasnu i preciznu viziju šta je smisao projekta, kako da ga započne i koje vreme je najpogodnije za početak realizacije, kada i kako će ga završiti, na koji način će voditi projektni tim ka ostvarivanju cilja projekta, ... Uspešno vođenje projekta zahteva mnogo više od korišćenja najnovijih i najboljih tehnika i alata. Uspeti u vođenju projekata znači uspeti u rukovođenju.

Uspešan vođa projekta je optimista, pesimista ili realista? Ipak, za uspeh u vođenju je potrebno pomalo od svega navedenog. Optimista, zato što bi uspešno vodio svoj projektni tim i upravljao resursima. Pesimista, naravno diskretno, zato što bi mogao predvideti situacije koje mogu poći u pogrešnom, ne planiranom smeru, te ih tako na vreme preduhitriti. Realista, zato što bi projekat sagledao iz svih aspekata, bez emocija i potpuno trezveno. Generalno, uspešan vođa projakta mora da poseduje:

- stručno znanje o temi projekta,
- veštinu upravljanja projektima,
- veštinu za uspostavljanje i održavanje dobrih odnosa sa članovima tima,
- sposobnost da motiviše druge u kriznim situacijama,
- veštinu slušanja i razumevanja,
- podršku nadređenog i podređenog rukovodstva,
- dobre organizacione sposobnosti,
- visok stepen inovativnosti,
- potrebno iskustvo,
- izraženu komunikativnost,
- intuitivnost određivanja prioriteta,
- smelost donošenja odluka i preuzimanja inicijative u kriznim situacijama,
- upornost, ...

Projektni tim čine ljudi koji su angažovani na projektu. Stepen njihove angažovanosti zavisi od njihovih znanja, kompetencija i veština, kao i od vrste projekta na kom su angažovani. Odabir ljudi za projektni tim uključuje definisanje njihovih uloga i odgovornosti.

Plan projekta je dokument koji svim članovima tima omogućava uvid u to gde treba ići, kada krenuti i kada stići, šta je potrebno učiniti da bi se ostvarili definisani ciljevi projekta. Plan projekta može da bude sasvim jednostavan popis zadataka i rokova, ali može da bude i vrlo složen dokument koji uključuje čitav niz potplanova (manjih planova) važnih za opis svega onoga što tokom projekta treba ostvariti. Aktivnosti na projektu ne započinju se bez plana projekta koji su potpisali (i tako se s njim složili) naručilac i finansijer projekta i vođa projekta ispred svih članova projektnog tima.

Plan projekta nastaje kao rezultat zajedničke aktivnosti i napora, uloženog znanja i iskustva svih članova tima. Tako bi barem trebalo da bude. Međutim, vrlo često se u praksi događa da projektni timovi „naslede“ već isplanirane projekte i planove ili da planove izrađuju same vođe projekata bez konsultovanja ostalih članova tima. Stručnjaci za upravljanje projektima slažu se u konstataciji da se tako gubi jedan od ključnih elemenata procesa planiranja. Naime, tokom planiranja članovi tima smišljaju svaki korak, mentalno prolaze kroz projekat, kroz trenutke njegove realizacije, te ih unapred proživljavaju, a to im kasnije omogućava brzu reakciju, razumevanje i spremnost na svaku nepredviđenu aktivnost pri realizaciji projekta, ali i usklađenost sa ličnim planovima. Osim toga, tokom procesa planiranja članovi tima „usvajaju projekat“, planiraju njegov razvoj i izgled rezultata, te projekat tako postaje njihova „vlastita tvorevina“.

Studija izvodljivosti projekta (eng. *Project feasibility*) je utvrđivanje da li projekat vredi pokretati i realizovati. Projekat, idejno, može da deluje kao nešto što vredi realizovati, ali ograničenja u vremenu i ceni mogu da spreče da takav projekat ugleda ”svetlost dana”.

Životni ciklus projekta (eng. *Project life cycle*) čine događaji neophodni za kompletiranje projekta, od početka do kraja projekta. Svi projekti su podeljeni u faze i nebitno da li su veći ili manji, složeni ili jednostavni, imaju sličnu strukturu životnog ciklusa. Svaki projekat ima početnu fazu, srednju (ili više njih) i završnu fazu. Broj faza u projektu zavisi od složenosti projekta, kao i oblasti kojoj projekat pripada.

Metodologija (eng. *Methodology*) prema definiciji PMI-a predstavlja skup praksi, tehnika, procedura i pravila koja se koriste u određenoj disciplini, gde je procedura niz koraka koji se, da bi se nešto postiglo, odvijaju po redosledu [7]. Metodologija se definiše i kao skup smernica i principa koji se mogu prilagoditi i primeniti u specifičnoj situaciji [13]. Međutim, mora se naglasiti da metodologija nikako nije privremeno rešenje.

Metodologija upravljanja projektima je skup međusobno zavisnih aktivnosti koje se izvode kako bi se postigao predviđeni skup proizvoda, rezultata ili usluga [7]. Dobra metodologija sadrži sve važne faze upravljanja projektima. Takođe, karakteristike dobre metodologije su srogo definisani detalji projekata, standardizovane tehnike planiranja vremena i kontrole troškova, standardizovani izveštaji, fleksibilnost prilikom primene na svim projektima, fleksibilnost za brzi razvoj, razumljivost korisniku, prihvaćenost i upotrebljivost, kao i visok stepen funkcionalnosti [10]. Metodologijom se mogu smatrati uloge i zaduženja u timu, veštine, tehnike, standardi i alat koje koristi projektni tim [14]. Na osnovu toga, može se reći da ukoliko to primenjuje jedna ili dve osobe, onda se može nazvati metodom, a ukoliko to primenjuje ceo projektni tim, onda je reč o metodologiji.

2.3 Upravljanje projektima

Upravljanje projektima je naučna disiplina koja je tokom vremena evoluirala od skupa procesa smatranih kao poželjnim u organizacijama, pa sve do struktuirane metodologije koja se danas smatra neophodnom za opstanak svake organizacije. Svi su danas svesni da se kompletno poslovanje, uključujući i najveći broj rutinskih aktivnosti, može posmatrati kao serija projekata. Jednostavno rečeno, upravljanje organizacijama se sve više svodi na upravljanje projektima.

Tradicionalno upravljanje projektima zasniva se na upravljanju ljudima [8]. Na taj način je uključen niz tehnika za planiranje, predviđanje i kontrolu aktivnosti radi postizanja željenog rezultata prema unapred strogo definisanim pravilima u određenom vremenu i sa određenim troškovima [10] [14]. Prema PMI, tradicionalni pristup uzet je i kao temelj za podelu na faze upravljanja [7]:

- pokretanje (eng. *Initiating*) – definisanje i odobravanje projekta;
- planiranje (eng. *Planning*) – planiranje akcija za postizanje cilja;

- upravljanje (eng. *Executing*) – koordiniranje ljudskim i drugim resursima u svrhu sprovođenja definisanog plana projekta;
- praćenje i kontrola (eng. *Monitoring and Controlling*) – praćenje i kontrolisanje napretka projekta radi uočavanja odstupanja od plana i preduzimanja potrebnih akcija;
- zatvaranje (eng. *Closing*) – prihvatanje proizvoda, usluge ili rezultata i završetak projekta.

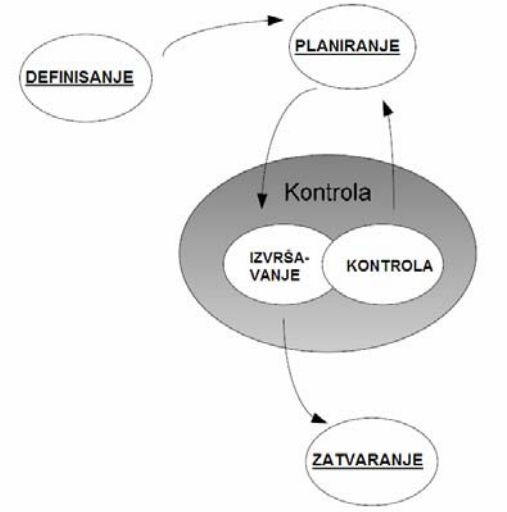
Kao prvi zadatak, koji se stavlja pred vođu projekta i projektni tim je definisanje zadataka. U nekim slučajevima, definisanje zadataka potrebno je uraditi čak i pre formalnog početka projekta.

U tradicionalnom pristupu plan je temelj svega jer predstavlja ne samo opis poslova i vremena potrebnog za njihovo sprovođenje nego i alat za donošenje odluka. Prilikom planiranja se određuju zadaci i dodjeljuju kompetentnim članovima tima. Planiranje smanjuje nesigurnost, jer pruža mogućnost ispravljanja pogrešnih koraka u postizanju željenoga cilja, povećava razumevanje ciljeva i zadataka projekta, a samim tim povećava i učinak uzimanjem u obzir moguće raspodele posla u odnosu na vremenski plan i dostupnost resursa. Za dobro i uspešno planiranje potrebno je odgovoriti na pet ključnih pitanja:

- O kom problemu se radu?
- Koji je cilj projekta?
- Koji se zadaci moraju obaviti kako bi se došlo do cilja?
- Kako će se odrediti uspešnost projekta?
- Postoje li neke pretpostavke, rizici ili prepreke koje mogu uticati na uspešnost projekta?

Izvršavanje, kao naredni korak, u principu je dozvola za obavljanje dodeljenih zadataka. Tokom izvršavanja potrebno je pratiti i kontrolisati da li se sve odvija prema planu, kako bi se na vreme i uz što manje gubitaka ispravile uočene nepravilnosti da bi se projekat završio na vreme i uz planirane troškove. Naravno, pored vremena i troškova, preduslov je i uspešno obavljen zadatak.

Na kraju, projekat je potrebno i formalno završiti, odnosno zatvoriti. Time se ukazuje da su svi projektom planirani zadaci uspešno završeni. Gotov projekat se predaje klijentu, a korišćeni resursi se oslobađaju kako bi se mogli dodeliti nekom novom projektu. Takođe, tokom zatvaranja projekta obavlja se evaluacija, koja u određenim situacijama može poslužiti kao vredan izvor znanja, naročito ako se dogodilo nešto nepredviđeno, a uspešno je savladano. To ukazuje da je završetak projekta važan deo, ali se često zanemaruje, najčešće zbog pritiska da se odmah počne rad na sledećem projektu.



slika 3 - Koraci u tradicionalnom upravljanju projektima

Svi opisani koraci se nazivaju fazama životnog ciklusa projekta (videti sliku 3) [8][10]. Takođe, moguće je iskoristiti i samo one faze koje su potrebne za određeni projekat, ali su u tom slučaju prve tri faze (definisanje, planiranje, izvršavanje) uvek obavezne dok se faze kontrole i

završetka mogu dodati u zavisnosti od veličine i vrste projekta. Iako je tradicionalni pristup upravljanja projektima vrlo robustan, moguće ga je primeniti i u najjednostavnijim, a i u najkompleksnijim projektima, jer se tokom projekta uvek primenjuju isti koraci.

Težnja za stalnim inovacijama i borba za smanjenjem troškova, koje su bile prisutne u svim granama industrije, potstakle su grupu autora da osmisle nov pristup u razvoju softvera. Iako nov, ipak se zasniva na nekoliko već poznatih principa: neprekidna inovacija, prilagođavanje proizvoda, smanjenje vremena isporuke, prilagođavanje ljudi i procesa, pouzdani rezultati [16]. Sve je to uključeno u pojam agilnosti, koji predstavlja više stav nego proces. Agilnost je sposobnost i da se stvori i da se odgovori na promenu kako bi se ostvarila dobit u turbulentnom poslovnom okruženju [16]. Takođe, agilnost je sposobnost da se balansira između fleksibilnosti i stabilnosti.

Tako je 2001. godine stvoren Manifest za agilni razvoj softvera [17] u kom su postavljene četiri dragocene vrednosti agilnog razvoja:

- članovi razvojnog tima i odnosi među njima moraju biti ispred procesa i alata,
- sam razvoj softvera je bitniji od opsežne dokumentacije,
- saradnja i stalna komunikacija sa klijentima stavlja se ispred dugih i sporih pregovaranja i pisanja ugovora,
- brza i efikasna reagovanja na promene u mnogome su korisnija od krutog i ne fleksibilnog poštovanja zacrtanog plana.

Međutim, time se ne ističe da su procesi, dokumentacija, ugovori, plan i alat nevažni, nego da su manje važni. Iako je Manifest napisan, na prvom mestu, za agilni razvoj softvera, ove bitne karakteristike mogu se primeniti neposredno, uz neznatne izmene, i na agilno upravljanje projektima.

Agilno upravljanje projektima temelji se na tradicionalnom pristupu i njegovih pet faza, ali su te faze modifikovane kako bi bile prikladnije za primenu u praksi [16]:

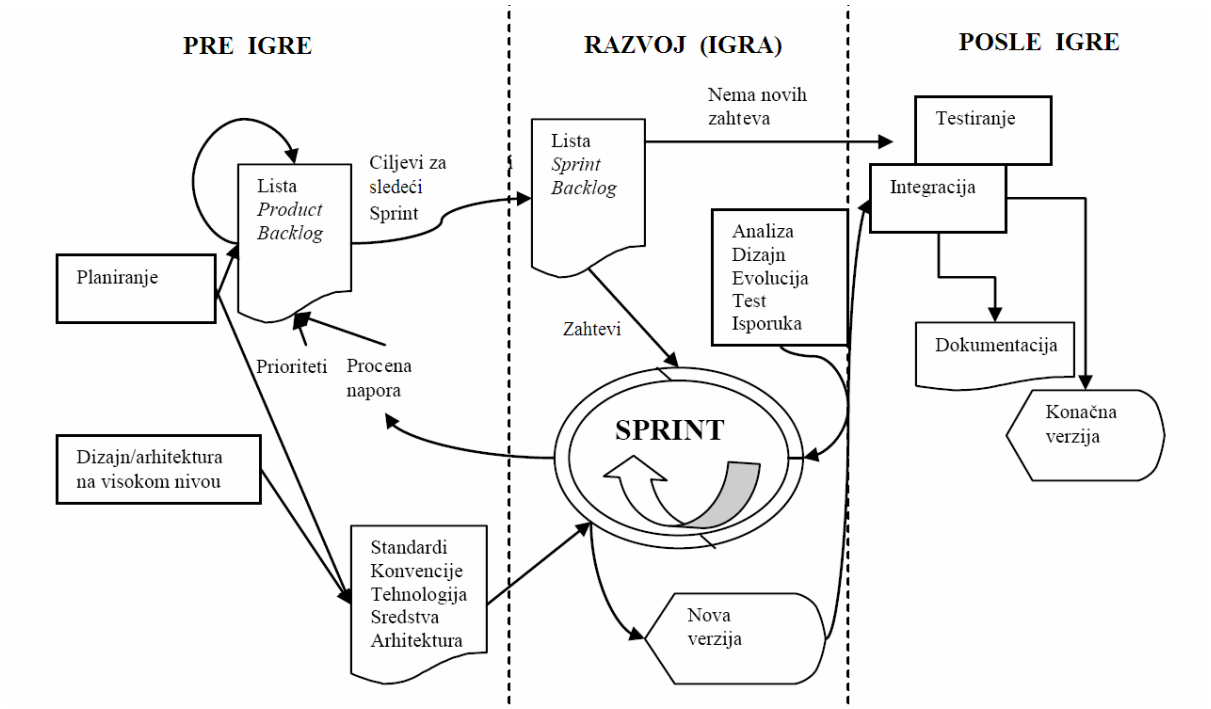
- planiranje (eng. *Envision*) – odrediti viziju projekta i projektnu organizaciju (umesto inicijalne faze, da se naglasi važnost vizije);
- nagađanje (eng. *Speculate*) – razviti model koji je određen karakteristikama, funkcionalnošću, vremenskim odrednicama, planom iteracija za ostvarivanje vizije (umesto planiranja, da se naglasi nesigurnost jer je plan povezan sa određenošću);
- istraživanje (eng. *Explore*) – isporučivati testirane delove u kratkim periodima i stalno tražiti načine za smanjenje rizika i nesigurnosti unutar projekta (umesto upravljanja, jer je istraživanje iterativno, odnosno nelinearno);
- prilagođavanje (eng. *Adapt*) – pregledati isporučene rezultate, trenutnu situaciju i ponašanje projektnog tima i sve što je potrebno prilagoditi na adekvatan način;
- zatvaranje (eng. *Close*) – završiti projekat i naglasiti ključne stvari koje su naučene u toku samog rada na projektu.

Agilno upravljanje naglašava iterativni pristup projektu, pa je zbog toga primenljiv na projektima različitih veličina. Mogu se očekivati dodatna usklađivanja ukoliko se radi o izuzetno velikim projektnim timovima. Po mišljenju autora ovog rada, ipak je metodologiju agilnog upravljanja projektima bolje primenjivati na projektima čiji su timovi manje ili srednje veličine, kako bi se postigli najbolji rezultati i najveći stepen produktivnosti, i očuvala stabilnost.

Ekstremno upravljanje projektima (eng. *Extreme Project Management*) je nastalo na temeljima ekstremnog programiranja - XP (akronim, eng. *eXtreme Programming*). Glavni cilj XP-a je da se završi projekat, primjenjujući niz jednostavnih principa. Životni ciklus projekta se sastoji od pet faza, kojima je malo promenjen naziv [18]:

- istraživanje,
- planiranje,
- iteracije do završnog međuproizvoda,
- pretvaranje u proizvod,
- održavanje i smrt.

SCRUM[*] kao razvojni pristup uključuje nekoliko bitnih karakteristika okoline i tehničkih karakteristika koje imaju veliku mogućnost promene u toku samog projekta. Akcenat je stavljen na organizaciju i motivaciju projektnog tima. Faze životnog ciklusa su pre igre, razvoj i posle igre (videti sliku 4).



slika 4 - *Scrum*

Najbolji efekat primene agilnog upravljanja je primetan u projektima s velikim rizikom i visokim stepenom promena. To nikako ne znači da u drugim projektima nije moguće primeniti neki od agilnih pristupa upravljanja, već da je izvesno da će biti potrebne određene dopune delovima iz tradicionalnog pristupa.

Ne zadugo posle početnog oduševljenja agilnim upravljanjem postalo je očigledno da agilnost ne odgovara svim vrstama projekata, kao što ni tradicionalni pristup u potpunosti ne odgovara. U pokušaju iznalaženja dobrog balansa između dve krajnosti, nastalo je adaptivno upravljanje (eng. *Adaptive Project Framework*) [8][19]. Adaptivni pristup temelji se na planu koji se sledi u potpunosti, ali za razliku od tradicionalnog pristupa, plan nije izrađen na početku već ciklično u toku samog projekta, dok su vreme i troškovi definisani na početku projekta i sprovode se kroz pet faza:

- realizacija dogovorene verzije,
- plan ciklusa,
- izgradnja ciklusa,
- provera od strane klijenata,
- osvrt na verziju.

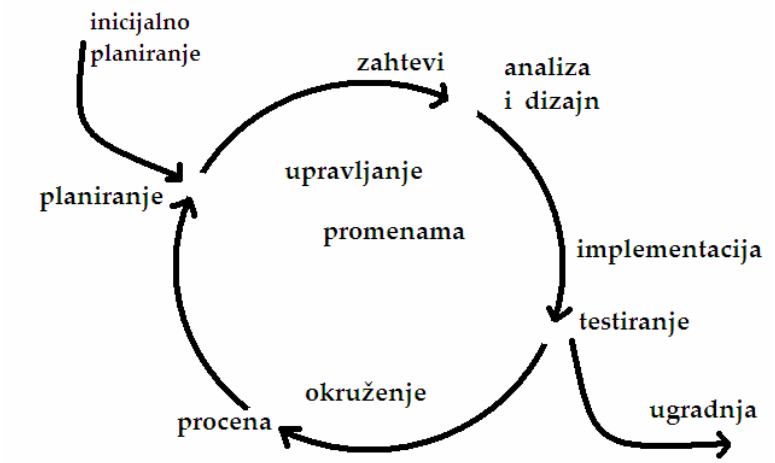
Cilj ovog pristupa je da se u plan uvrste samo one aktivnosti za koje se sa sigurnošću zna da će biti deo konačnog rešenja. Na taj način se u narednim iteracijama dolazi do detaljnijeg rešenja.

Osim navedenih pristupa upravljanja projektima, razvijeno je još nekoliko formalizama, koji se uglavnom temelje na tradicionalnom pristupu. Razlike su primetne u pogledu upravljanja rizikom na projektu, jednostavnosti implementacije i upravljanja, veličine projektnog tima i podrške učestalim promenama. Svi novi oblici su, što je i karakteristika tradicionalnog pristupa, pouzdani, dokumento-

[*] Čest utisak da je to akronim je pogrešan. *Scrum* je termin preuzet iz ragbija i označava momenat kada se protivnički timovi skupljaju na gomilu i bore za posed lopte. To ovde nije vezano, osim simbolički, za softverski projekat, jer se više odnosi na agilno upravljanje softverskim projektima, nego na agilno projektovanje softvera.

vani, pružaju podršku promenama. Jedan od poznatijih je *PRINCE2* (eng. *Projects in Controlled Environments*) koji je postao standard za IT projekte u Velikoj Britaniji. Glavne osobine su definisana struktura upravljanja projektima, fleksibilno donošenje odluka, planiranje resursa, skup kontrolisanih procedura, fokusiranost na proizvode kroz ceo projekat [13]. Naravno, ovakav pristup moguće je primeniti i na ostale projekte izvan oblasti informacionih tehnologija.

Takođe, značajan pristup upravljanja projektima je RUP (akronim, eng. *Rational Unified Process*), koji je prvenstveno namenjen razvoju softvera. Sadrži osnovne karakteristike modernog razvoja softvera poput: iterativnog razvoja, arhitekture utemeljene na komponentama, vizuelnog modeliranja, upravljanja zahtevima, stalne kontrole kvaliteta, kao i kontrolisanja promena (videti sliku 5) [13].



slika 5 - *RUP*

Metodologija upravljanja projektima je alat u rukama veštog i iskusnog vođe projekata, jer on kao odgovorno lice rukovodi planiranjem, realizacijom i kontrolom projekta za koji je zadužen. Zbog toga njome mora da se obuhvati:

- planiranje svih aktivnosti na projektu, uključujući dostupnost materijalnih, ljudskih, finansijskih, tehničkih i drugih resursa;
- efikasno praćenje i kontrolu svih radova, materijalnih i finansijskih tokova na projektu, s ciljem pravovremenog preduzimanja potrebnih akcija i sprečavanja „neprijatnih iznenađenja“ po okončanju projekta;
- pravovremeno vođenje kompletne dokumentacije o projektu.

Danas su osnovne funkcionalne oblasti metodologije upravljanja projektima međusobno povezane i zavisne (videti sliku 6). Svaka od tih oblasti je ”priča za sebe”, a zbog teme i obima ovog rada neće se ulaziti u detalje svake oblasti ponaosob. Sledi navođenje tih oblasti, bez prioriteta koja je bitnija a koja ne, jer naprosto takvo rangiranje među njima nije moguće dati precizno:

- upravljanje obimom projekta

  Upravljanje obimom (eng. *project scope management*) uključuje procese potrebne da se utvrdi da projekat uključuje sav potreban rad, i samo taj traženi rad, da bi se uspešno realizovao. Projekat je uspešan samo kada se ostvari ono što je traženo, ništa manje, ali i ništa više.
- upravljanje ljudskim resursima

  Upravljanje ljudskim resursima (eng. *human resource management*) uključuje procese neophodne za organizovanje i upravljanje projektnim timom.
- upravljanje ugovaranjem

  Upravljanje ugovaranjem (eng. *contracting management*) uključuje procese neophodne za donošenje odluke o realizaciji projekta i obezbeđivanju finansijskih sredstava, izradu projektne dokumentacije i izradu ponude.

- upravljanje nabavkom

  Upravljanje nabavkom (eng. *procurement management*) uključuje procese neophodne za nabavku proizvoda ili usluga potebnih za realizaciju projekta.

- upravljanje komunikacijom

  Upravljanje komunikacijom (eng. *project communication management*) uključuje procese neophodne za garantovanje pravovremene i odgovarajuće komunikacije: generisanje, prikupljanje, distribucija, skladištenje i preuzimanje projektnih informacija.

- upravljanje promenama

  Upravljanje promenama (eng. *change management*) uključuje procese neophodne za uočavanje potrebe za promenom i stvaranje nove vizije, upravljanje i održavanje promene.

- upravljanje rizikom

  Upravljanje rizicima (eng. *risk management*) uključuje procese neophodne za planiranje, identifikaciju, analizu, praćenje i kontrolu rizika, kao i procese za planiranje reakcija na rizik. Glavni ciljevi ovog upravljanja su povećati verovatnoću i uticaj pozitivnih događaja, a smanjiti verovatnoću i uticaj nepovoljnih događaja za projekat.

- upravljanje konfliktima

  Upravljanje konfliktima (eng. *conflict management*) uključuje procese neophodne za određivanje vrste konflikta, planiranje principa za rešavanje konflikta i izveštavanje o ishodima konflikta.

- upravljanje kvalitetom

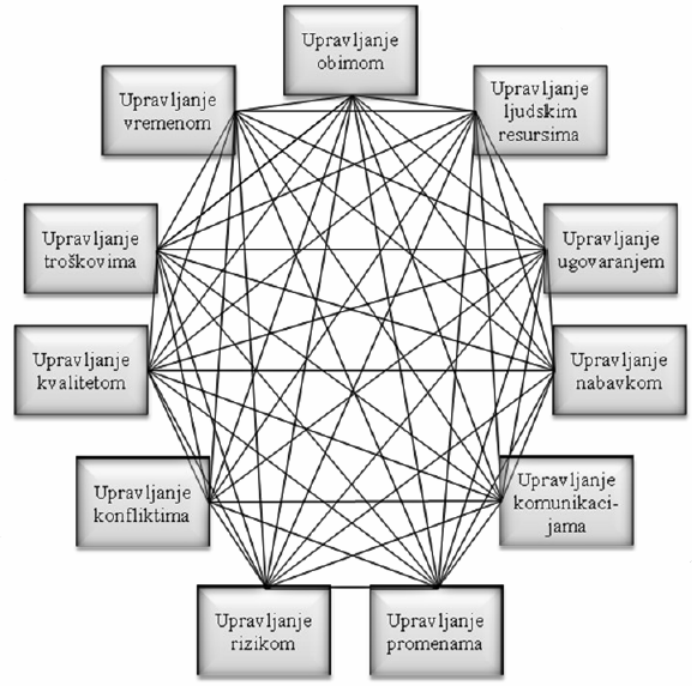
  Upravljanje kvalitetom (eng. *quality management*) uključuje procese neophodne za planiranje, osiguranje i kontrolu kvaliteta sa težnjom na unapređenju tih procesa.

- upravljanje troškovima

  Upravljanje troškovima (eng. *cost management*) uključuje procese neophodne za planiranje, predvidjanje i kontrolisanje troškova tako da projekat može biti realizovan u okviru dozvoljenog budžeta.

- upravljanje vremenom (rokovima)

  Upravljanje vremenom (eng. *time management*) uključuje procese neophodne da se projekat završi na vreme, tj. u zadatom roku.



slika 6 - *Osnovne funkcionalne oblasti metodologije upravljanja projektima*

2.4 Specifičnosti IT projekata

Premda je polje informacionih tehnologija izrazito veliko, u današnje vreme se pod IT projektima podrazumevaju i projekti iz domena hardvera (na primer: nabavka računara i računarskih komponenata, povezivanje računara sa perifernim uređajima, fizičko instaliranje računarskih mreža, zamena grafičke karte, ...) i softvera (na primer: projektovanje i implementacija različitih vrsta softverskih aplikacija, informacionih sistema, baza podataka, web portala, ...), kao i projekti koji obuhvataju kombinovanje hardvera i softvera. Međutim, zbog zastupljenosti, osetljivosti na različite vrste promena, specifičnosti i kreativnosti, u daljem izlaganju u ovom radu pod IT projektom podrazumevaće se isključivo softverska rešenja.

Upravljanje IT projektima moguće je definisati kao primenu aktivnosti upravljanja planiranjem, koordinacijom, merenjem, praćenjem, kontrolom i izveštavanjem, kako bi se osiguralo da razvoj i održavanje softvera bude sadržajan, disciplinovan i kvalitetan [20].

Iako se može reći da je upravljanje u softverskoj industriji moguće sprovesti na isti način kao i u drugim (složenim) procesima, postoje neki aspekti koji su specifični za sam softver i procese životnog ciklusa softvera, a koji upravljanje čine izrazito složenim [21]:

- U percepciji klijenata često je prisutno nerazumevanje složenosti koja je svojstvena procesu razvoja softvera, posebno u odnosu na posledice koje može imati promena zahteva.
- Gotovo je sigurno da sami procesi razvoja softvera dovode do potrebe za novim ili izmenjenim zahtevima.
- Zbog toga se softver često radije razvija iterativno, nego u nizu zatvorenih celina.
- Razvoj softvera neophodno uključuje aspekte kreativnosti i discipline. Održavanje primerene ravnoteže između ova dva aspekta često je vrlo složeno.
- Stepen inovativnosti i kompleksnosti softvera često je ekstremno veliki.
- Prisutne su brze promene u tehnologijama i alatu koji se koriste u samom razvoju softvera.

Prema tome, aktivnosti upravljanja u softverskoj industriji odvijaju se u tri pravca:

1. organizacijsko upravljanje i upravljanje infrastrukturom,
2. upravljanje projektima,
3. planiranje i kontrola softvera.

Nažalost, uobičajena je praksa da se softver isporučuje sa kašnjenjem uz prekoračenje svih postavljenih ograničenja i proračuna, a često i lošeg kvaliteta i funkcionalnosti. Upravljanje informacijama koje se dobijaju merenjem može promeniti ovakvu stvarnost.

Upravljanje razvojem softvera može se posmatrati kao organizacijski proces koji uključuje pojam procesa i projektnog upravljanja. Pojam upravljanje projektima koristi se da bi se: definisao projektni plan i ciljevi, odabrao i sačinio projektni tim, definisali procesi razvoja, pratilo i kontrolisalo napredovanje projekta.

Pod pojmom produktivnosti u kontekstu razvoja softvera misli se na meru kojom ljudi, uključeni u razvoj, stvaraju softver i odgovarajuću prateću dokumentaciju. Prema tome, kada govorimo o produktivnosti, zapravo želimo izraziti realizovanu funkcionalnost po jedinici vremena.

Faze u upravljanju IT projektima [21]:

- inicijacija i definisanje opsega - odluka o započinjanju projekta,
- planiranje projekta - aktivnosti koje je potrebno sprovesti kao pripremu za uspešan projekat iz perspektive upravljanja,
- izrada projekta - aktivnosti upravljanja koje se mogu realizovati tokom razvoja softvera,
- pregled i evaluacija - osiguravanje da softver zadovoljava postavljene kriterijume (funkcionalnost i kvalitet),
- zatvaranje - aktivnosti na projektu koje se obavljaju po završetku projekta,
- merenje – procena efikasnosti razvoja i implementacije softvera.

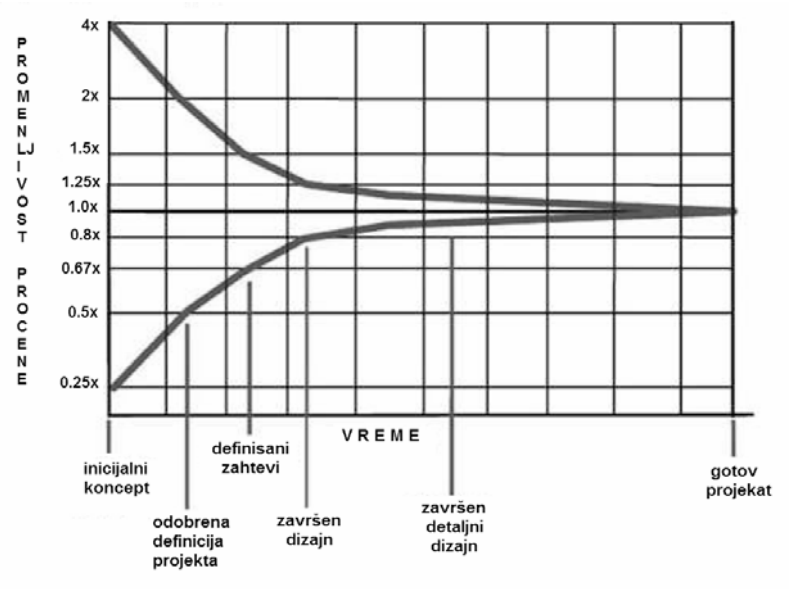
Softver koji treba razviti u velikoj meri moguće je proceniti u odnosu na detaljne specifikacije zahteva samog softvera. Specifikacije zahteva su osnova za izradu liste značajnih elemenata koje softver mora sadržati, a koje se u datom raspoloživom vremenu moraju razviti, testirati i isporučiti.

U tradicionalnom pristupu upravljanja IT projektima, procenu treba napraviti barem tri puta [23]:

- faza studije izvodljivosti – ”gruba” procena,
- faza specifikacije zahteva - detaljna procena,
- faza dizajna - dorada procene.

Procena troškova razvoja softvera na samom početku projekta u fazi studije izvodljivosti treba da bude toliko pouzdana da može da pomogne pri donošenju odluke o detaljnom definisanju projekta. Detaljno definisanje softvera i projekta u ovom trenutku nisu dostupni, pa se procena radi sa visokim rizikom (“*u grubo*”). Procena u fazi specifikacije zahteva je kritična tačka i zbog toga mora biti napravljena sa velikom i preciznom tačnošću. U ovom trenutku se zapravo donosi odluka o ulaganju značajnih resursa, a sve na osnovu procene troškova. Potrebna pouzdanost zavisi i od toga da li se radi o internom projektu, nekoj softverskoj aplikaciji, informaciono komunikacionom sistemu ili projektu za spoljnog klijenta. Jednom kada je napravljen detaljan dizajn i kada je veliki deo nepoznanica otklonjen, moguće je napraviti razumno tačnu procenu i projektni plan za realizaciju softvera.

Kako bi projekat procenili s većom pouzdanošću moramo razumeti razloge neizvesnosti u proceni i znati njome upravljati. Veličina neizvesnosti procene zavisi od toga koliko smo blizu kraju projekta. Barry Boehm je 1981. godine objavio klasičan prikaz pouzdanosti procene u različitim fazama životnog ciklusa IT projekta koji je postao poznat kao ”konus neizvesnosti” (videti sliku 7) [24].



slika 7 - *Konus neizvesnosti*

Sa slike 7 se jasno vidi da tokom inicijalne faze projekta faktor pouzdanosti procene iznosi ±4, zbog pritisaka tržišta, preteranog optimizma i ”sigurnosti”. Primera radi, ako je objektivna procena projekta 24 čovek-meseci, može se dogoditi da na početku naša procena ”u grubo” bude između 6 i 96 čovek-meseci. Kako projekat napreduje po fazama životnog ciklusa, pouzdanost procene se povećava, pa tako u fazi detaljnog dizajna taj faktor iznosi približno ±1.25, a ne više početnih ±4. Kako projekat napreduje, i kako zahtevi postaju jasniji i konačno se precizno definišu, tada i procene postaju sve pouzdanije. U trenutku kada se na projektu zaista počne sa implementacijom, procene su obično unutar 10% finalnih vrednosti [25].

Važno je napomenuti da se ove vrednosti zasnivaju na procenama koje prave vešti, iskusni stručnjaci za procene, koristeći se formalnim metodama procene. Ukoliko su procenitelji manje iskusni ili ako koriste manje precizne tehnike procene, njihove procene veoma lako mogu značajnije odstupati.

Na neizvesnost realizacije IT projekta utiče pouzdanost procene, poznati rizici i eventualni neplanirani događaji. Pouzdanost procene se temelji na verovatnoći i samo je pitanje koliko je velika i dobro procenjena. Poznate rizike treba eksplicitno označiti u planu za upravljanje rizicima. Nepoznati događaji su prema tome najveći problem jer ih je nemoguće predvideti. Najčešća strategija dobre prakse je angažovanje eksperata za predviđanje nepredvidivih događaja. Time bi se smanjio broj nepoznatih događaja. U slučaju kada ne postoje druge informacije koje bi upućivale na nepredvidljive događaje, po mišljenju autora ovog rada preporučljivo je povećati resurse, na primer za 10-30%.

Zbog neizvesnosti koja je svojstvena svim projektima, ponekad će biti potrebna ponovna procena zbog novih okolnosti i činjenica. Uvek, kada se dogode neki značajani nepredviđeni događaji, treba dobro razumeti kako će oni uticati na raspored projekta i uraditi novu procenu. Na osnovu ranih i učestalih procena mogu se preduzeti adekvatne akcije upravljanja, uvek kada se otkrije da projekat ne prati planirani proračun ili projektni plan [23].

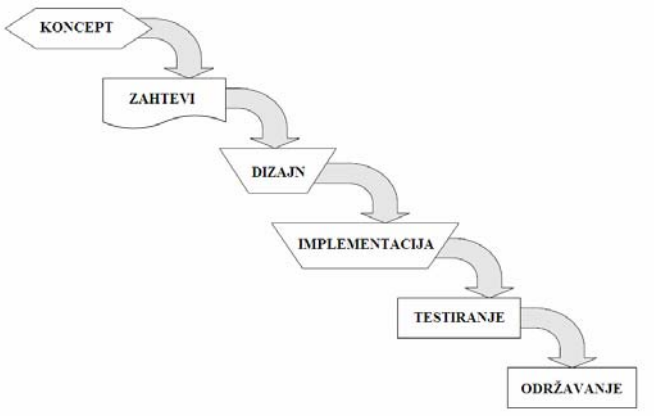
Upravljanje IT projektom moguće je sagledati u kontekstu konkretnog IT projekta, organizacije i tehničkih uslova. Skoro svaki pristup upravljanja ima argumente za i protiv, a na projektnom timu i vođi projekta je da odluče koji pristup je najprikladnije primeniti na konkretnom projektu. Ponekad se najbolje rešenje dobija kombinovanjem različitih pristupa.

Sledi kratak pregled najčešće korišćenih pristupa upravljanja IT projektima od kojih je svaki primenljiv za određeni tip projekta i problema koji treba rešiti:

- Vodopad (eng. *Waterfall*) zahteva striktno pridržavanje propisanog redosleda faza, dok se informacije kreću od jedne ka drugoj fazi projekta. Propisane faze ovog pristupa su (videti sliku 8) [3]:

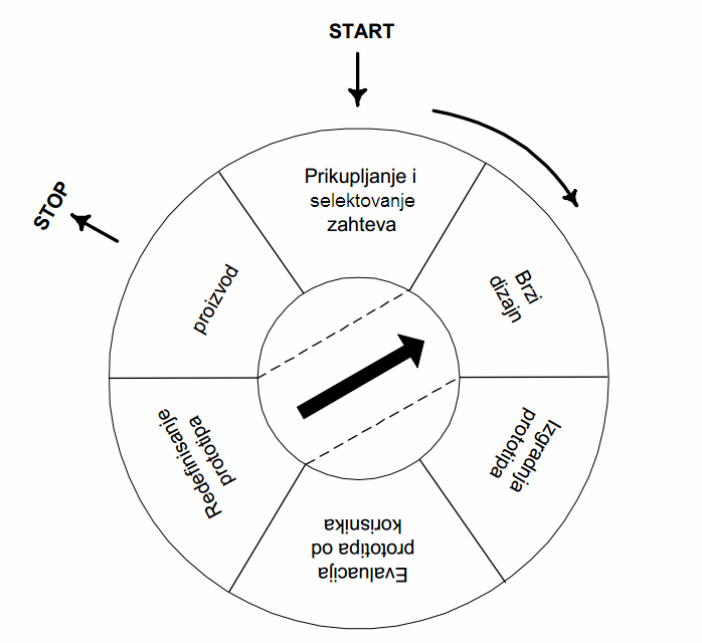
    - koncept,
    - prikupljanje zahteva,
    - dizajniranje sistema,
    - implementacija rešenja,
    - testiranje rešenja,
    - održavanje sistema.

    U slučaju da je potrebno izvršiti neke izmene, vrlo je teško vratiti se na neki od prethodnih koraka. Akcenat se stavlja na planiranje unapred čime se smanjuje potreba za stalnim planiranjem u toku realizacije projekta. Ovaj pristup je najkorisniji kada su zahtevi kupca strogi i do detalja unapred poznati. Njegova primena se preporučuje kod projekata koji imaju visoko rizične komponente, poput medicine, bezbednosti, ...



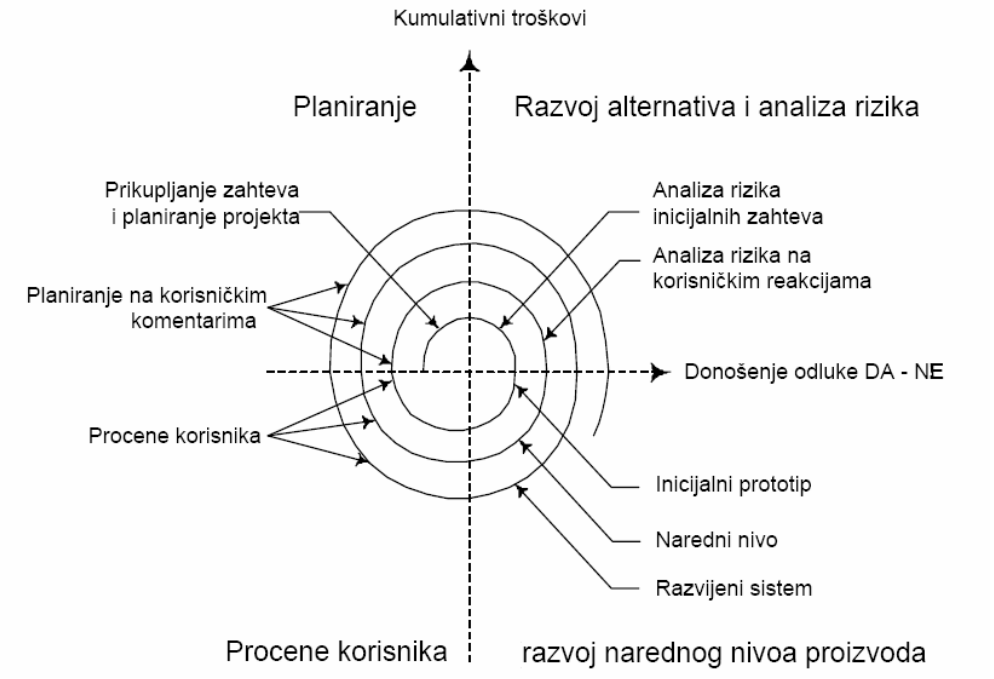
slika 8 - *Waterfall pristup projektovanju softvera*

- U prototipskom pristupu, pre finalizovanja projekta gradi se prototip (videti sliku 9), koji se koristi za dalji razvoj stabilnijih rešenja. Ovaj pristup je veoma koristan kod projekata koji nemaju jasno definisane zahteve.



slika 9 – *Prototipski pristup*

- Spiralni pristup kombinuje karakteristične delove vodopada i prototipskog pristupa (videti sliku 10). Zasniva se na kontinuiranom poboljšavanju zahteva, dizajna i implementacije. Svakom iteracijom nastaje bolja verzija rešenja projekta. Naglasak je na upravljanju rizicima.



slika 10 – *Spiralni pristup*

- U iterativnom pristupu projekat se razvija u iteracijama, a svaka iteracija rezultira rešenjem koje radi. Ovaj pristup ne zahteva da na početku budu definisani svi zahtevi, dozvoljava da se povratne informacije iz ranijih iteracija koriste u sledećim iteracijama i smanjuje rizik, jer se funkcionalnosti isporučuju kako projekat napreduje.
- RUP (akronim, eng. *Rational Unified Process*) obuhvata četiri faze izvršavanja projekta – početak, elaboracija, konstrukcija i tranzicija, od kojih se svaka završava s jasno definisanim izlazima. Sam model se može izvoditi iterativno, a zahtevi, dizajn, koodiranje i testiranja su aktivni tokom celog projekta, iako njihov intenzitet varira od faze do faze.

- Agilni pristup (*eXtreme Programming - XP*, *Scrum* i dr.) zasniva se na principima razvoja projekta u malim iteracijama, gde je mera napredovanja zapravo projekat koji radi, a promena je moguća u bilo kom trenutku. Uopšteno, projekat započinje kratkim korisničkim pričama čiji detalji se prikupljaju tokom iteracije u kojoj se one razvijaju. U samoj iteraciji, razvoj se odvija u timovima (parovima, grupama od 7±2 članova, i sl.) sledeći učestalu integraciju i korišćenje jednostavnog dizajna koji se prema potrebi refaktoriše.

2.5 Tehnike u upravljanju projektima

Metodologija upravljanja projektima obuhvata skup različitih tehnika. Neke od najčešće korišćenih su:

- strukturni dijagrami
- Gant dijagram
- metod kritičkog puta
- tehnika evaluacije i analize projekta
- Ishikawa dijagram

2.5.1 Strukturni dijagrami

Strukturni dijagrami se koriste da se stvore uslovi, olakša i poboljša planiranje, praćenje i kontrola projekta. Postoje tri strukturna dijagrama:

- PBS (akronim, eng. Product Breakdown Structure)

  Koristi se za podelu projekta na faze. Na ovom dijagramu se ne vidi koji zadaci u okviru projekta treba da se urade i ko će ih raditi, nego se pokazuje kojim redosledom treba da se realizuje projekat.
- WBS (akronim, eng. Work Breakdown Structure)

  Dijagramom strukture projektnih poslova se prikazuje hijerahijska struktura rada koja opisuje obim posla na projektu. Drugim rečima, za razliku od PBS dijagrama, WBS dijagramima se pokazuje koji se sve zadaci moraju odraditi u okviru projekta. Koristi se kako bi se zadaci u okviru projekta raščlanili na sastavne elemente (najčešće do trećeg nivoa dubine, s tim da je na poslednjem nivou potrebna angažovanost manja od 80h) sa prikazom odnosa između elemenata i celine.
- OBS (akronim, eng. Organization Breakdown Structure)

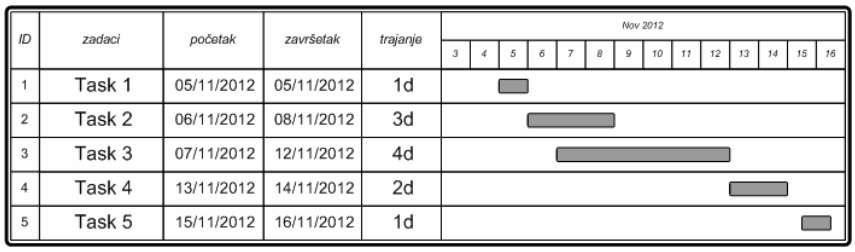
  Dijagramom organizacione strukture projekta se daje prikaz ko je kompetentan da uradi zadatke koji su prikazani WBS dijagramom.

Pored strukturnih dijagrama, često se koristi i matrica odgovornosti – RACI (akronim, eng. Responsability, Accountability, Comunication, Information). Spajanjem WBS i OBS dijagrama dobija se RACI matrica koja pokazuje odgovornost za izvršenje zadataka u okviru projekta. Njome se povezuju zadaci koje treba uraditi, a koji su prikazani WBS dijagramom, sa stručnjacima koji će te zadatke uraditi, a koji su predstavljeni OBS dijagramom. Matricom odgovornosti se precizira: odgovornost za ceo projekat, odgovornost za sve zadatke, komunikacija i međusobno informisanje u realizaciji projekta.

2.5.2 Gant dijagram

Gant dijagram (tzv. gantogram) se koristi za predviđanje ishoda vremena, troškova, kvaliteta i kvantiteta. To je horizontalni dijagram koji grafički predstavlja vremenske odnose između koraka u projektu, zadatke koji se nadovezuju, kao i zadatke koji se paralelno izvršavaju. Zadaci se unose

prema redosledu izvođenja, a shodno svom vremenskom trajanju. Na taj način se sagledava minimalno ukupno vreme potrebno za realizaciju projekta, potreban niz koraka, kao i zadaci koji će se izvršavati paralelno.



slika 11 – *Gant dijagram (gantogram)*

2.5.3 Metod kritičkog puta

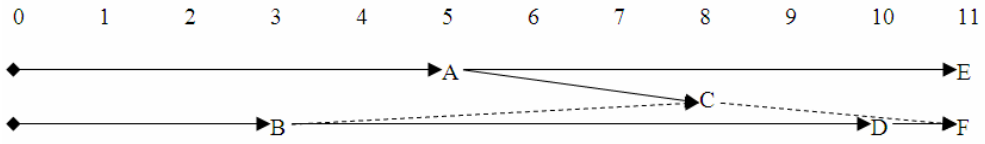
Metod kritičkog puta – CPM (akronim, eng. *Critical Path Method*) koristi se za izradu i kontrolu vremenskih planova projekta, kada je vreme zadataka u projektu poznato. Nakon procene vremena svih zadataka na projektu, prelazi se na predviđanje ukupnog vremena potrebnog za realizaciju projekta. Ovaj metod može značajno da pomogne u borbi protiv prekoračenja planiranih rokova. Prilikom crtanja dijagrama kritičkog puta, ispod vremenske ose na kojoj su prikazani dani, meseci ili godine trajanja projekta, usmerenim strelicama se predstavljaju trajanja zadataka, a isprekidanim usmerenim strelicama se označava vreme neoperativnosti u toku trajanja paralelnog zadatka (videti sliku 12).

Metod kritičkog puta ne koristi se samo da bi se prikazali odnosi između različitih zadataka u projektu, već se koristi i za izračunavanje kritičnog puta. Kritični put projekta je najduži put u CPM dijagramu kojim se utvrđuje najraniji završetak projekta. Projekat je završen onda kada su završeni svi zadaci predviđeni tim projektom. Ako jedan ili više zadataka na kritičkom putu traju duže nego što je planirano, ćitav plan projekta će kasniti, osim ako vođa projekta preduzme korektivne mere.

zadaci	preduslovi	trajanje
A		5
B		3
C	A, B	3
D	B	6
E	A	4
F	C, D	1

tabela 2 – *Lista projektnih zadataka*

Radi ilustracije CPM dijagrama, u tabeli 2 navedena je lista projektnih zadataka, preduslovi za početak zadataka, kao i njihovo trajanje izraženo, na primer, u danima. Ukoliko bi projekat obuhvatao zadatke od A do E, tada bi kritični put projekta bio put A-E dužine 11. To bi značilo da je najkraće procenjeno vreme za realizaciju projekta 11 dana. Sve više od toga bi se smatralo vremenskim prekoračenjem plana projekta. Međutim, ako bi u projekat bio uključen i zadatak F, tada bi bilo više od jednog kritičnog puta: A-E, A-C-F, B-C-F, B-D-F.



slika 12 – *CPM dijagram*

Prema tome, može postojati više kritičnih puteva na jednom projektu. Vođa projekta treba da pažljivo prati izvršavanje zadataka na kritičnom putu da bi se izbeglo kašnjenje završetka projekta. Ako postoji više kritičnih puteva, projektni timovi moraju pratiti svaki od njih, jer se oni mogu menjati tokom rada na projektu.

2.5.4 Tehnika evaluacije i revizije projekta

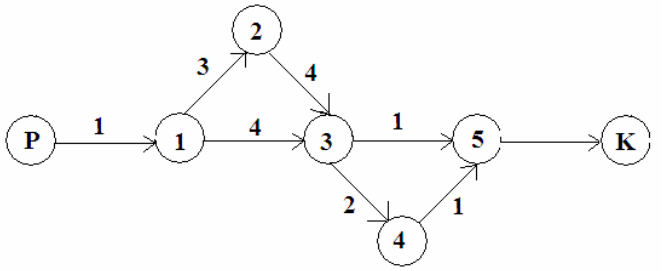
Tehnika evaluacije i revizije projekta grafički se predstavlja PERT (akronim, eng. *Project Evaluation and Review Technique*) dijagramom. U pogledu planiranja, ovi dijagrami su rafiniraniji od gantograma, jer su pogodni za projekte koji sadrže veliki broj koraka. Za razliku od CPM dijagrama, PERT dijagrami prikazuju napredak u svakom zadatku ili u celom projektu.

Tehnika evaluacije i analize projekta koristi se za procenu trajanja projekta u situacijama sa visokim stepenom nesigurnosti procene trajanja pojedinačnih zadataka. Ova tehnika primenjuje metod kritičnog puta na ponderisani prosek procene trajanja, u oznaci E(t), a izračunava se po formuli:

$$E(t) = \frac{\text{optimističko} + 4 \times \text{najverovatnije} + \text{pesimističko vreme}}{6}$$

Iz formule se vidi da se koriste tri verovatne procene vremena trajanja projektnih zadataka – optimističko, pesimističko i najverovatnije procenjeno vreme, umesto jednog procenjenog trajanja kao kod CPM-a.

U PERT dijagramima zadaci su prikazani krugovima, aktivnosti sa strelicama koje povezuju krugove, a periodi neaktivnosti između dva zadatka pokazani su kao isprekidane strelice. Redni brojevi zadataka se upisuju u krugove, a procenjeno vreme koje je potrebno za realizaciju sledećeg zadatka zapisuje se na strelici. Na slici 13 prikazan je primer PERT dijagrama na kome su zadaci označeni brojevima 1, 2, ..., 5, a slovima P i K su naznačeni, redom, početak i kraj projekta.



slika 13 – *PERT dijagram*

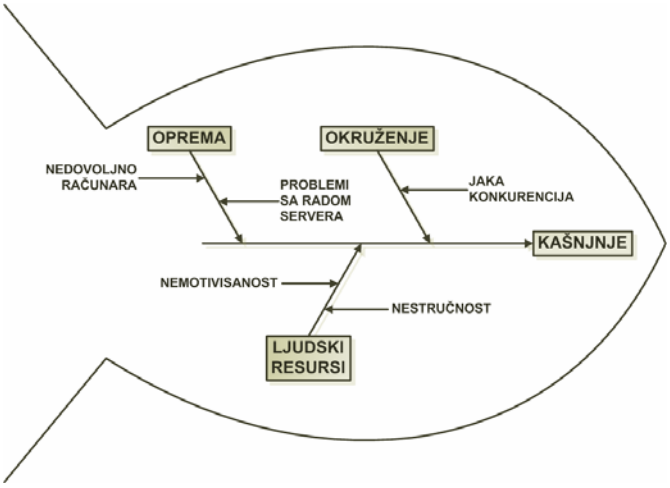
Kako se u velikom broju projekata prekoračuje procenjeni vremenski plan, PERT može biti od pomoći za izradu realističnijih vremenskih planova. Međutim, ovom tehnikom se ne uzima u obzir šta se dešava ukoliko se promeni kritični put projekta, što je u praksi čest slučaj.

2.5.5 Ishikawa dijagram

Ishikawa dijagram se koristi za analizu uzroka i posledica, odnosno za analiziranje složenih problema koji imaju više povezanih uzroka. Ovaj dijagram je nazvan po japanskom profesoru Kaoru Ishikawi, koji je prvi koristio ovu vrstu dijagrama 1943. godine. Zbog specifičnog izgleda, ovaj dijagram se često naziva „riblja kost“ (videti sliku 14).

U postupku pravljenja dijagrama treba:

- opisati problem na krajnjoj desnoj strani dijagrama,
- identifikovati potencijalne uzroke i grupisati ih u kategorije,
- za svaku kategoriju definisati preciznije uzroke na kosim crtama,
- ići, ukoliko je potrebno, zbog preciznosti, do dubine od tri nivoa poduzroka uzroka, koji su takođe povezani kosim crtama sa uzrocima kojima pripadaju,
- paziti da dijagram ne bude prenatrpan uzrocima.



slika 14 – *Ishikawa („riblja kost“) dijagram*

2.6 Softver za upravljanje projektima

Velika zastupljenost metodologije upravljanja projektima dovela je do brzog razvoja i primene različitog softvera u ovoj oblasti. Generalno, postoje dva tipa softvera za upravljanje projektima - desktop aplikacije i veb aplikacije, kojima se pristupa preko Interneta. U nastavku rada sledi uporedni prikaz softvera prema komponentama koje sadrže (videti tabelu 3). Osnovne analizirane komponente su: mogućnost saradnje, praćenje, raspoređivanje, projektni portfolio menadžment, upravljanje resursima, upravljanje dokumentima, sistem toka rada (eng. *workflow system*), izveštavanje i analiza, baziranost na mreži (eng. *web-based*) i licenciranost.

Saradnja je mogućnost da tim koji radi na projektu uspešno i blagovremeno razmenjuje informacije i dokumente. Takođe i menadžeri imaju mogućnost redovnog obaveštavanja o statusu projekta, kao i mogućnost da kreiraju izveštaje koji će biti dostupni svim učesnicima u projektu.

Sistemi za praćenje kao delovi softvera za upravljanje projektima omogućavaju praćenje projekata po određenim elementima, poređenje i upozoravanje na moguća ili nastala odstupanja u odnosu na planirane veličine, kao i upravljanje i rešavanje nastalih odstupanja.

Raspoređivanje se odnosi na proces raspoređivanja resursa na planirane zadatke. Pod projektnim portfolio menadžmentom se misli na analiziranje i upravljanje grupom trenutnih ili predloženih projekata na osnovu brojnih karakteristika. Osnovna svrha ove komponente je pomoć menadžerima da na optimalan način iskombinuju projekte koji će omogućiti da organizacija na najbolji mogući način postigne svoje ciljeve.

Upravljnje resursima pruža mogućnost da se na najbolji način rasporede postojeći resursi i da se postignu ciljevi u okviru planiranog. Sistem toka rada predstavlja postavljanje tokova posla i pravila koja povezuju niz zadataka. Ovaj sistem omogućava upravljanje, kontrolu, kao i uvid u efikasnost izvršenja. Smanjenje potrebnog vremena, povećanje produktivnosti, poboljšanje kvaliteta i smanjenje troškova su dokazi značaja ove komponente.

Sledi lista popularnog softverskog alata za upravljanje projektima koji su poređeni prema navedenim komponentama (videti tabelu 3):

- *Cooper project* je veb alat kompanije *Element Software. Cooper* se isporučuje besplatno određenim neprofitnim organizacijama (uključujući univerzitete, humanitarne organizacije, itd.). Godine 2007. *Apple* je kompaniju *Element Software* prepoznao kao vodećeg preduzetnika u industriji softvera za upravljanje projektima.
- *HP Project and Portfolio Management Center* je softver kompanije *Hewlett-Packard* iz grupe IT menadžment proizvoda.
- *JIRA* potiče iz *Atlassian*-a koji je isporučuje besplatno za: projekte otvorenog kôda koji ispunjavaju određene kriterijume, organizacije koje su ne profitne, ne vladine, ne akademske, ne komercijalne, ne političke. *JIRA* se koristi u više od 15000 organizacija u preko 120 zemalja širom sveta [5] i može se slobodno reći da je jedan od najpopularnijih informacionih sistema ove vrste.

- *Merlin* je komercijalni softver kompanije *Project Wizards* iz Nemačke za *Mac OS X*.
- *Microsoft Office Project*[*] je alat za planiranje, praćenje, izveštavanje i donošenje odluka o projektu. Postoji verzija *Standard* i *Professional*. Opcije ovog softvera su unapređene uvođenjem *Microsoft Office Project Server*-a.
- *Onepoint Project*, kompanije *Onepoint Software*, je softverski paket koji kombinuje projektno planiranje, praćenje napretka, monitoring projekta, kontrolu i izveštavanje. *Onepoint Project* integriše formalne, agilne i *JIRA* projekte u jedinstveni portfolio projekata i bazu za korišćenje resursa. Takođe ovo rešenje uključuje standarde *IPMA*-e i *PMI*-a.
- *Open ERP Project Management* je softver kompanije *Open ERP* koja u okviru svog *open source* sveobuhvatnog opsega poslovnih aplikacija razvija ovu aplikaciju za upravljanje projektima.
- *Primavera Project Planner* je softversko rešenje kompanije *Primavera Systems*. Godine 2009. kompanija *Oracle Corporation* pripaja *Primavera Systems* jer je po njihovom mišljenju to kompanija koja je razvila najbolja softverska rešenja na tržištu projektnog portfolio menadžmenta. Ovo Primaverino rešenje koristi više od 450000 korisnika u preko 60000 organizacija širom sveta [38]. Trenutno su aktuelni verzije *Enterprise* i *Professional*.
- *Project Open* je sveobuhvatan alat za upravljanje projektima u potpunosti integrisan sa ERP (akronim, eng. *Enterprise Resource Planning*) sistemom. Ovaj softver spada u softver otvorenog kôda i korisnicima je omogućeno da biraju elemente u okviru modula koji su im potrebni za upravljanje projektima.
- *SAP Portfolio and Project Management* je razvila kompanija *SAP*, čiji je cilj bio da razvije aplikaciju koja će na nivou organizacije pomoći upravljanje brojnim procesima, uključujući kontinuirano stvaranje proizvoda i usluga, integrisani razvoj proizvoda, upravljanje sredstvima i informacionim sistemom organizacije. Korisnici mogu koristiti podatke iz različitih sistema uključujući ljudske resurse, finansije, projektni menadžment i desktop sisteme, omogućavajući nove funkcionalne poslovne procese i obezbeđujući ažuran uvid u poslovanje.
- *Work PLAN Enterprise* spada u grupu ERP softverskih rešenja kompanije *Sescoi*.

	1)	2)	3)	4)	5)	6)	7)	8)	9)	10)
Cooper project	♦	♦	♦	♦	♦	♦		♦	♦	♦
HP Project and Portfolio Management Center	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
JIRA	♦	♦	♦				♦	♦	♦	♦
Merlin	♦	♦	♦	♦	♦	♦		♦	♦	♦
Microsoft Office Project	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Onepoint Project	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Open ERP Project Management	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Primavera Project Planner	♦	♦	♦	♦	♦	♦			♦	♦
Project Open	♦	♦	♦	♦	♦	♦	♦	♦	♦	
SAP Portfolio and Project Management	♦	♦	♦	♦	♦	♦			♦	♦
Work PLAN Enterprise	♦	♦	♦	♦	♦	♦	♦	♦		♦

legenda:

1) – saradnja
2) – sistem za praćenje
3) – raspoređivanje
4) – projektni portfolio menadžment
5) – upravljanje resursima
6) – upravljanje dokumentima
7) – sistem toka rada
8) – izveštavanje i analiza
9) – baziranost na mreži
10) – licenciranost

tabela 3 – *Uporedni prikaz softvera za upravljanje projektima*

[*] upravljanje razvojem softvera u *Microsoft Office Project*-u pojašnjeno je u Dodatku ovog rada

Međutim, na tržištu je velika ponuda softvera za upravljanje projektima koji je besplatan ili je dostupan po veoma niskoj ceni. Besplatan softver je dostupan ili za korišćenje putem Interneta ili za preuzimanje u obliku otvorenog kôda, čime se ostavlja mogućnost programerima da ga prilagode svojim potrebama i da ga integrišu u već korišćene informacione sisteme organizacija. Zbog teme i obima ovog rada, neće se ulaziti u detalje i suštinske razlike softvera ove vrste, već autor ovog rada navodi samo onaj koji je više u upotrebi: *jxProject, dotProject, GanttProject, ProjectPier, OpenProj, Clocking IT, TaskJuggler, iTeamWork, php-collab, PHProjekt, TeamSCOPE, Project Engine, LightHouse, Springloops, CreativePRO Office, NO Kahuna, Basecamp, Trac Project*.

Na osnovu analiziranja navedenog softvera, nameće se generalni zaključak da danas većina softvera sadrži ne samo osnovne već i napredne opcije koje omogućavaju organizacijama dobru osnovu za uspešno upravljanje projektima i poslovanjem.

3 EKSTREMNO PROGRAMIRANJE

3.1 Razvoj ekstremnog programiranja

Koncepte ekstremnog programiranja u praksi je prvi primenio Kent Beck. Zbog toga se on smatra tvorcem ovog novog pristupa u razvoju softvera. Njegova knjiga "Ekstremno programiranje - Objašnjenja" nije praktični vodič, već pre manifest ovog novog pristupa u razvoju i implementaciji softverskih aplikacija.

Ekstremno programiranje – XP (akronim, eng. *eXtreme Programming*) je nastalo tokom realizacije projekta C3 (eng. *Chrysler Comprehensive Compensation*) [1]. Zamisao tog projekta je bila da se niz različitih sistema sa spiskovima plata zaposlenih sastave u jednu softversku aplikaciju. Prvi pokušaj razvoja je propao usled složenosti pravila koja vladaju u toj oblasti, kao i teškoća u integraciji [1]. Korišćeni pristup u razvoju softvera - *Waterfall* nije dao rezultate. U kritičnom trenutku 1996. godine, Kent Beck sa svojim timom je pozvan u kompaniju *Chrysler* da preuzme projekat C3. Oni su morali da ga praktično počnu od početka.

XP tim u *Chrysler*-u je prvu radnu verziju sistema isporučio za nepunih godinu dana, tako da je već 1997. godine prvih 10000 službenika primalo platu pomoću novog sistema [1]. Razvoj sistema je nastavljen i u naredne dve godine, kada su nove funkcije dodavane u malim količinama. Projekat C3 je prekinut 1999. godine, jer je pažnja *Chrysler*-a bila usmerena u drugim pravcima [1].

Nakon svega, ostala je činjenica da je XP tim sastavljen od samo 8 programera napravio sistem, sa 2000 klasa i preko 3000 metoda, koji je funkcionisao bez većih grešaka [3]. Stoga je XP bio testiran, otklonjeni su početni nedostaci, tako da je bio spreman za široku upotrebu. XP su napravili programeri i on je namenjen programerima. Zbog toga, iako XP nije bio prvi agilni pristup u razvoju softvera, on popularizuje i omasovljava upotrebu agilnosti. Po podacima *Cutter* konzorcijuma[*] iz 2001. godine, 38% ispitanika koristilo je XP u svakodnevnom razvoju softvera, što je više od svih ostalih agilnih pristupa zajedno [3].

XP se koristi u timovima male i srednje veličine (od 6 do 20 ljudi) koji prave programe u uslovima kada se zahtevi brzo menjaju [1]. Timovi blisko sarađuju sa klijentima, izbegavajući planiranje unapred i veliku dokumentaciju na račun brzih iteracija koje klijentu daju realan rezultat. Zbog toga su prednosti korišćenja XP-a velike [3]:

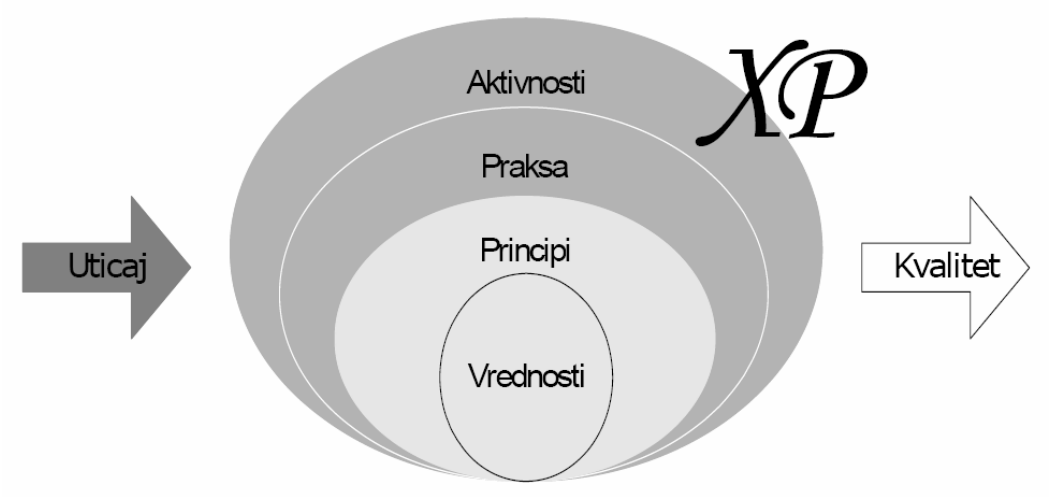
- vreme razvoja softvera je znatno kraće,
- želje klijenata su pravilno implementirane,
- promene se lakše dodaju u projekat,
- uvek postoji funkcionalna verzija,
- greške su znatno smanjenje tokom programiranja,
- odsustvo prekovremenog rada,
- testovi za svaki deo programa, ...

[*] za više informacija o *Cutter* konzorcijumu posetiti zvanični sajt www.cutter.com

Imajući u vidu navedene prednosti, sve veći broj softverskih kompanija prelazi na XP. Njihovo organizovanje može da bude drugačije u svakom konkretnom slučaju i za svaki pojedinačni zahtev, jer XP ne predstavlja krut pristup razvoju softvera, već dozvoljava izmene u izvođenju pojedinih aktivnosti i obaveznih praksi koje su detaljno opisane u sledećim poglavljima ovog rada.

3.2 Definisanje ekstremnog programiranja

XP je kompleksan pojam koji sadrži četiri dela - vrednosti, principe, aktivnosti i prakse koji nadograđuju jedan drugog [1]. Jezgro XP su osnovne vrednosti, koje se postižu prihvatanjem principa, oni se dalje sprovode kroz skup mehanizama koji se upotrebljavaju u praksi. Sve to se ogleda u aktivnostima rada tima (videti sliku 15) [1].



slika 15 – *Delovi ekstremnog programiranja*

3.2.1 Vrednosti ekstremnog programiranja

Vrednosti XP-a predstavljaju primenu i nadogradnju osnovnih vrednosti agilnog razvoja softvera [17]. U XP-u je od izuzetnog značaja da one budu poznate i prihvaćene u timu [3]. Kent Back ističe značaj prihvatanja ovih vrednosti i upozorava da je njihovo neprihvatanje osnovni uzrok eventualnog neuspeha realizacije projekta pomoću ekstremnog programiranja [1]. Slede vrednosti XP-a i njihovo objašnjenje [33]:

- komunikacija

  Loša komunikacija ne dolazi slučajno. Mnoge okolnosti utiču na to. XP pokušava da korišćenjem praktičnih mehanizama u radu prevaziđe probleme u komunikaciji. Jako je važno da se znanje i informacije nesmetano kreću, jer bez toga nema većeg uspeha.

- jednostavnost

  Postići jednostavnost nije lako. XP je posvećen jednostavnosti i neguje princip da je bolje posao završiti što jednostavnije, pa ga kasnije nadograditi ako treba, nego odmah uraditi na komplikovan način trošeći više vremena, iako se ta složena funkcionalnost možda nikada neće koristiti i isplatiti. Ovakav stav poznat je kao YAGNI princip[**].

[**] YAGNI (akronim, eng. *You Ain't Gonna Need It!*) ili u prevodu *„Neće ti trebati“* je princip koji upućuje da se ne radi nešto samo zato jer se misli da će to nekada možda zatrebati. Velika je verovatnoća da će se utrošiti resursi na rešavanje problema koji ne postoji i da će se tako ugraditi neka funkcionalnost koju klijent nije tražio. Raditi sa ograničenjem, jedna je od najtežih stvari koje programeri moraju da nauče kada prelaze na XP. Treba raditi samo ono što je trenutno potrebno, i ništa više od toga!

- povratna sprega

  Kontkretno i brzo saznanje o stanju sistema je neprocenjivo. Optimizam je nepotrebni rizik programiranja i čest izvor grešaka. Povratna sprega je lek za to. Ona se odvija na relacijama: klijent-programer, klijent-sistem, programer-sistem i programer-programer. Ona je krajnje praktična, jer se na konkretnom primeru pokazuju problemi, što vredi više nego bilo kakav opis ili rasprava.

- hrabrost

  Ovo je vrednost koja podrazumeva stalnu borbu sa promenama i rizicima kako na tržištu, tako i unutar tima. Tu spadaju prihvatanje i primena novih tehnoloških rešenja, kao i hrabrost da se deo urađenog posla odbaci i počne iznova, ako je to potrebno. Ova vrednost je vezana i za samopouzdanje i veru u rad tima.

- poštovanje

  Sve ove vrednosti obuhvata i prožima međusobno poštovanje članova tima. Niko nesme da se oseća odbačen ili zapostavljen, jer takva osećanja članova tima vode neuspehu.

Ove vrednosti su u velikoj meri povezane. Jednostavnost i komunikacija imaju veliku zajedničku snagu. Što se više komunicira jasnije se sagledavaju potrebe i više se zna šta se treba činiti, a šta ne. Što je bolja povratna sprega, to se lakše komunicira. A sama hrabrost bez ostalih vrednosti nema nikakvog smisla jer sa njima ona daje ekstremnu vrednost projektu.

3.2.2 Principi ektremnog programiranja

Principi XP-a proizilaze iz osnovnih principa agilnog razvoja softvera, proširujući i konkretizujući njihovo značenje u svakom pojedinačnom projektu. Vrednosti XP-a pružaju osnovne kriterijume po kojima se sagledava uspešnost posla, ali su suviše široke da bi se na osnovu njih mogli definisati praktični mehanizmi za osiguranje uspešnosti. Zato principi, zajedno sa vrednostima, čine jezgro ekstremnog programiranja. Iz njih proizilaze odluke koje se sprovode obaveznim praksama, kojih se tim mora držati disciplinovano do ekstrema. Zato se ovaj pristup u programiranju i naziva ekstremnim [1][3]. Slede principi XP-a i njihovo objašnjenje [33]:

- brza povratna sprega

  Umesto da se na odgovor od klijenta čeka mesecima ili godinama, on informaciju vraća u roku od nekoliko dana ili nedelja. S druge strane, programeri saznaju u roku od nekoliko minuta ili sekundi da li je dizajn dobar i da li program radi, umesto da na to čekaju danima ili nedeljama. To pojačava razumevanje i učenje.

- pretpostavljena jednostavnost

  Vreme koje se uštedi pretpostavljajući da je svaki problem trivijalno jednostavan za rešavanje, daleko prevazilazi vreme koje se izgubi u retkim slučajevima kada ova pretpostavka nije istinita.

- postepene promene

  Sve promene treba sprovoditi u malim pomacima, te se zbog toga svaka velika promena razbija u niz minimalnih koraka.

- prihvatanje promena

  Najbolja strategija je da se rešavaju prvo veći problemi, pa tek onda oni manji.

- kvalitetan rad

  Niko ne voli da radi loše. Jedine moguće vrednosti kvaliteta su odlično i perfektno. U suprotnom se ne uživa u poslu, što izaziva pad kvaliteta rada.

3.2.3 Obavezne prakse ekstremnog programiranja

Obavezne prakse u XP-u predstavljaju skup praktičnih mehanizama pomoću kojih se sprovode aktivnosti i realizuje razvoj softvera pomoću XP-a. Postoji dvanaest ovih mehanizama koji se sprovode ekstremnom disciplinom. Vrednosti ekstremnog programiranja moraju biti osnovni motiv i pokretač rada. Jedino je tako moguće uspešno primenjivati ovih dvanaest praktičnih mehanizama [3]. Sledi navođenje mehanizama obaveznih praksi u XP-u i njihovo objašnjenje [33]:

- programiranje u paru

  U XP-u programeri rade u paru. Dva programera sede za jednim računarom radeći na istom zadatku.

- planiranje igre

  Ne treba se predugo zadržavati oko plana koji treba da da osnovne smernice. Treba brzo i grubo planirati unapred.

- razvoj vođen testovima – TDD (akronim, eng. Test Driven Development)

  Ne treba pisati ni jednu liniju kôda pre nego što se napiše test koji taj kôd treba da zadovolji. Cilj kodiranja se svodi samo na zadovoljavanje testova. Klijent takođe testira rešenja i pomoću rezultata testa upravlja razvojem.

- celokupnost tima

  U timu bi trebao da se nađe i predstavnik klijenta koji bi u timu imao ulogu stručnog savetnika. On piše priče korisnika, bavi se funkcionalnim testiranjem i razjašnjava nejasnoće u hodu.

- stalna integracija

  Kako se komponenta završi, sistem se integriše. Mora se imati odgovarajući način za praćenje promena u kôdu koji dozvoljava poništavanje izmena, ako je potrebno.

- poboljšanje dizajna (refaktorisanje)

  Svaki put kada se unapređuje neka komponenta, to se radi pomoću refaktorisanja. Trebalo bi povremeno pregledati ceo kôd i refaktorisati ga kako bi se kôd održao čitljivim.

- male verzije

  Razvoj se odvija u malim iteracijama. Na taj način se smanjuje cena promena i odbacivanja loših rešenja. Svaka nova verzija se daje klijentu na testiranje i korišćenje, što ga uključuje u proces razvoja.

- standardi u kodiranju

  Da bi tim funkcionisao bez zastoja zbog prilagođavanja na tuđi stil programiranja uspostavljaju se praktični standardi kojih se pridržavaju svi članovi tima.

- kolektivno vlasništvo nad kôdom

  Svi su vlasnici svega, pošto svi rade sve. Tako se izbegava posvećenost pogrešnim idejama i rešenjima, otpor promenama i olakšava ujedinjavanje znanja i veština u timu.

- jednostavan dizajn

  Posvećenost jednostavnosti olakšava buduće promene i refaktorisanja, jer jednostavan dizajn proizvodi čist i samo neophodan kôd. Time je YAGNI princip pravac koji treba slediti.

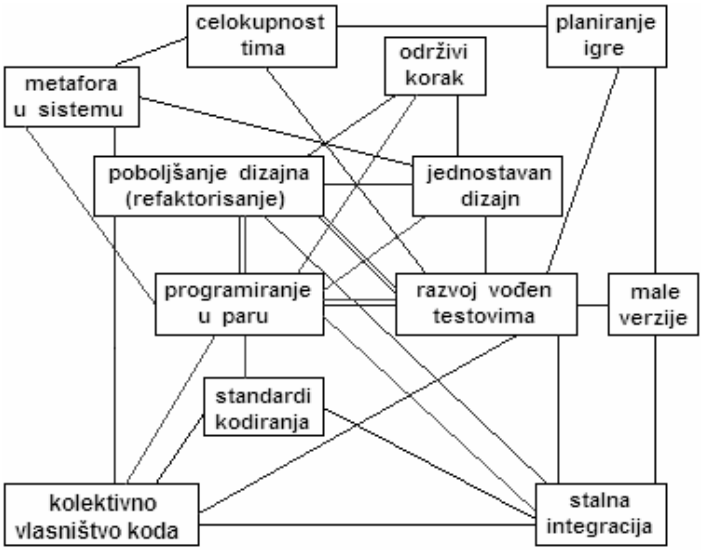
- metafora u sistemu

  Olakšava komunikaciju i pojačava osećanje pripadnosti grupi. Tim razvija svoj rečnik i termine za obeležavanje karakteristika sistema i svega ostalog.

- održivi korak

  Potrebno je uspostaviti ritam u radu koji treba da bude održiv na duge staze. Treba odrediti jednake intervale trajanja iteracija koji se dobijaju praćenjem dosadašnjih aktivnosti.

Na slici 16, može se primetiti da ni sve prakse nisu međusobno direktno povezane. Ima praksi koje direktno utiču na više drugih, a ima ih koje direktno utiču na manje njih. Ipak zapostavljanje neke prakse sa manje veza, vremenom vodi opadanju performansi tima u svim aspektima, jer indirektne veze takođe postoje, te su tako sve prakse međusobno povezane. Mora se priznati da ipak jezgro obaveznih praksi čine razvoj vođen testovima, programiranje u paru i refaktorisanje [1]. Ovo ne znači da je efikasan rad moguć bez drugih praksi, nego predstavlja samo jednu opservaciju stanja XP-a kao sistema. Veze ovih obaveznih praksi, predstavljaju dvosmerne linije uticaja, pa njihov broj koji polazi od (odlazi ka) elementu na neki način predstavlja nivo uticaja tog elementa na celokupni sistem. Ipak, ovaj način rangiranja praksi nije pouzdan. Na primer, praksa male verzije održava učestalu komunikaciju i povratnu spregu, pa bi njen izostanak, mnogo brže doveo do kolapsa od izuzimanja prakse metafora u sistemu. Prema tome, zaključuje se da broj veza nije najbitniji pokazatelj važnosti obaveznih praksi. Od broja veza važniji je odnos prema vrednostima i principima koji ima pojedina obavezna praksa XP-a.



slika 16 – *Povezanost obaveznih praksi XP-a*

Ovakva podela obaveznih praksi je rezultat višegodišnje primene. U početku je podela bila nešto drugačija. Određeni praktični mehanizmi su zamenjeni novim, neki su preimenovani, ali je većina ostala ista kao što je bila u osnovnoj podeli iz 1999. godine [1]. U verziji iz 2000. godine, skup obaveznih praksi, sastojao se iz sledećih praktičnih mehanizama [3]:

- programiranje u paru,
- planiranje igre,
- testiranje (postalo je razvoj vođen testovima),
- kupac na licu mesta (postalo je celokupnost tima),
- stalna integracija,
- poboljšanje dizajna (refaktorisanje),
- male verzije,
- standardi u kodiranju,
- kolektivno vlasništvo nad kôdom,
- jednostavan dizajn,
- metafora u sistemu,
- radna nedelja od 40 sati (postalo je održivi korak).

3.2.4 Aktivnosti ekstremnog programiranja

Na kraju se, u razvoju softvera, sve svodi na konkretne aktivnosti koje se sprovode u timu i koje vode visokom nivou kvaliteta u što kraćem vremenu i uz minimalne troškove, ostajući pri tome u domenu rešenja. Aktivnosti su ono što proizilazi iz vrednosti i principa. Imaju smisla jedino kada odražavaju te vrednosti i principe. Važne su za definisanje ponašanja tima u određenim situacijama. Takođe, moraju biti sprovedene na način striktno propisan obaveznim praksama [3]. Slede aktivnosti XP-a i njihovo objašnjenje [33]:

- kodiranje

  Ovo je oblik komuniciranja i predstavlja preslikavanje ideja u softver. Izvorni kôd je osnovni element softverskog programa. Preko njega se materijalizuju ideje, ali i vrši čista i nedvosmislena komunikacija, kako sa mašinom, tako i sa kolegama programerima. On može izražavati ideje za rešavanje problema, ali i testove. Može se koristiti i za opis algoritama, a može i ukazivati na moguća mesta daljeg razvoja sistema. Najbolja mera ispravnosti je kôd koji radi.

- testiranje

  Gotovi jedinični (eng. *unit*) testovi bi trebalo da budu potreban i dovoljan uslov za početak kodiranja, jer se bez testa ne može znati kada je kodiranje završeno, niti koliko tačno je sprovedeno. Takođe, na osnovu funkcionalnih testova, klijent određuje pravac daljeg rada. Na taj način se poboljšava i komunikacija među klijentima i programerima.

- slušanje

  Polazi se od saznanja da programeri ne znaju ništa o poslovnoj logici, te oni moraju aktivno da slušaju klijenta. Aktivno slušanje se odnosi na ukazivanje klijentu šta je teže, a šta lakše izvesti. Iz priče klijenta mora se izvući što više korisnih informacija, koje će kasnije pomoći u razumevanju sistema. Potrebna je česta komunikacija sa klijentom, kome se postavljaju pitanja i pažljivo slušaju odgovori.

- dizajn

  U XP-u dizajn teži jednostavnosti i smanjenju suvišnog kôda. Dijagrami i planovi nisu centralni elementi dizajna, već je to izvorni kôd. Ovo je još jedna revolucionarna promena u načinu razmišljanja programera.

Ove aktivnosti se moraju sprovoditi na smislen i strukturiran način. Ne mogu se izvoditi haotično i proizvoljno. Za uspostavljanje planskog i organizovanog izvođenja ovih aktivnosti koriste se obavezne prakse.

3.3 Praktične preporuke ekstremnog programiranja

Kao što je pokazano u prethodnom poglavlju, XP ima dvanaest obaveznih praksi. One se smatraju osnovnim praktičnim preporukama za pravilno primenjivanje XP-a u svakodnevnom radu, jer se njima opisuju načini na koji tim razvija sistem. Može se primetiti da se praktične preporuke već koriste, ali ono što je novo je upotreba preporuka na usaglašen način. Dobar primer za to je kolektivno vlasništvo nad kôdom, koje kada bi se koristilo nezavisno, došlo bi do potpunog haosa jer bi to značilo da svako, po ličnom nahođenju, može da menja kôd bez ikakvih ograničenja. Međutim, kod XP-a programiranje u parovima, standardi u kodiranju i testiranje predstavljaju ravnotežu kolektivnom vlasništvu nad kôdom. Prema tome, slabosti određene preporuke se prevazilaze primenom drugih praktičnih preporuka XP-a.

Još jedan aspekt praktičnih preporuka koji odlikuje XP je način na koji se primenjuju preporuke. Opšta praksa je da se preporuke primenjuju do krajnjih granica [3]. Za potrebe razvoja softvera, praktične preporuke je moguće podeliti u tri kategorije [33]:

- programiranje

  jednostavan dizajn, razvoj vođen testovima, poboljšanje dizajna (refaktorisanje), standardi u kodiranju;

- tim

  kolektivno vlasništvo nad kôdom, stalna integracija, standardi u kodiranju, održivi korak, programiranje u parovima, male verzije;

- procesi

  celokupnost tima, razvoj vođen testovima, male verzije, planiranje igre.

Vidi se da postoje izvesna preklapanja između kategorija, jer XP ima specifične praktične preporuke koje opisuju kako se kodira. Zbog toga će u nastavku biti objašnjene pojedine praktične preporuke. Autor ovog rada se prilikom izbora preporuka i redosleda njihovog navođenja vodio stepenom otpora uvođenja primena u svakodnevni rad.

3.3.1 Održivi korak

U prvoj verziji XP-a ova obavezna praksa se zvala ”radna nedelja od 40 sati”. Međutim, vremenom se došlo do zaključka da nije važno samo osmočasovno radno vreme, već je bitnije održati ujednačeni ritam rada koji ne opterećuje ni programere, ni sam projekat.

U drugim pristupima razvoja softvera praksa je zahtevala prekovremeni rad, posebno pri kraju projekta, kada se bližilo ugovorenim rokovima. Ovakav način rada je dodatno iscrpljivao programere, koji su, zbog smanjenja dnevnog i nedeljnog odmora, pravili sve više grešaka. Ni XP nije potpuno bez prekovremenog rada, ali se preporučuje da on, kada je neophodan, nikada ne traje dve nedelje za redom [1], jer preopterećeni tim ne može raditi brzo i prihvatati promene koje se javljaju ”u letu”.

Održivi korak se odnosi i na pravovremeno pripremanje svih pretpostavki za rad, planiranje iteracija, razvijanje priča korisnika i sprovođenje eventualnih dodatnih istraživanja, kako bi na sva pitanja postojali spremni odgovori.

3.3.2 Celokupnost tima

XP tim se ne sastoji samo iz programera i njihovih rukovodilaca, već se u taj tim uključuje i predstavnik klijenta koji bi u timu imao ulogu stručnog savetnika. On piše priče korisnika, bavi se funkcionalnim testiranjem i razjašnjava nejasnoće u hodu. U zavisnosti od veličine softverskog projekta i broj predstavnika klijenta varira od jednog stručnjaka, pa do dodatnog tima koji opslužuje potrebe svih razvojnih programerskih timova.

Prisustvo predstavnika klijenta je suštinski važno. Često je ovo teško obezbediti jer klijent nije siguran u nivo doprinosa koji njegov predstavnik može doneti razvoju softvera. Ljudi koji bi trebali biti u razvojnom timu često su klijentu neophodni na drugim mestima u okviru njegove organizacije. Ukoliko predstavnik pruži doprinos softveru i pomogne njegov brži razvoj, to je značajna korist. Ali ukoliko projekat propadne, nije izgubljeno samo vreme i rad koji je predstavnik klijenta uložio u softver, već je nepovratno izgubljen i rad koji bi taj čovek uložio u svoju matičnu organizaciju da nije radio na softveru [33][35]. To je nekada preveliki rizik. Ukoliko klijent ne delegira svog predstavnika, a projekat propadne, to je i njegova krivica isto koliko i krivica programera, jer mu je ponuđena mogućnost da doprinese razvoju projekta, a on to nije prihvatio [3][36].

Za razvoj komercijalnog softvera klijenta praktično nije moguće obezbediti u timu. Tad ulogu klijenta ipak neko mora da preuzme, jer je celokupnost tima obavezna praksa koje se XP tim mora pridržavati. Ovu ulogu najčešće preuzima trener u timu ili neko ko ima manje obaveza u timu. Ta osoba, pored uloge klijenta u timu, mora dodatno da istraži sve aspekte poslovanja klijenta kako bi timu mogao da pruži sve potrebne odgovore [35][36].

Uloga trenera u timu je bitno različita od uloge vođe projekta, jer je on tehničko lice koje je najposvećenije uspehu projekta. Mora imati visok nivo potrebnih tehničkih znanja, veština dizajna, potpuno razumevanje XP-a, kao i znanja o sprovođenju posla. On se s vremena na vreme uključuje u rad tima kroz programiranje u paru preuzimajući ulogu mentora [33][35]. Vođa projekta ugovara poslove, bavi se planiranjem i problemima funkcionisanja tima u širem okruženju. Takoreći, on krči put za rad tima, dok trener daje sve od sebe za omogućavanje sprovođenja obaveznih praksi XP-a. Ove dve funkcije mogu biti objedinjene i u jednoj osobi, ali je njihovo postojanje neophodno za uspešan

rad tima [33][36]. Pored navedenih osnovnih uloga u XP timu, može se pojaviti potreba za još nekim, poput konsultanta, analitičara, podrška klijentima, stručnjaka za procenu rizika, tehničara alata, ...

3.3.3 Praćenje razvoja

Neko u timu treba da se bavi merenjem i praćenjem napretka razvoja softvera. Jako je važno, a i u skladu je sa principima XP-a, da programeri imaju stalnu povratnu spregu. Praćenje ostvarenog je najbolji način za to. Ne insistira se ni na jednoj posebnoj metrici, ali se ne preporučuje više od pet parametara koji se stalno prate i čiji se rezultati ističu na vidno mesto u radnoj prostoriji. Najčešće se koriste:

- broj linija kôda softvera,
- broj linija kôda testa,
- broj pronađenih grešaka,
- broj ispravljenih grešaka,
- broj sati utrošenih na posao,
- broj klasa sistema,
- broj završenih iteracija,
- broj iteracija koje je klijent prihvatio, ...

3.3.4 Jednostavan dizajn

Uvek neko mora biti posvećen dizajnu sistema. S vremena na vreme (bar jednom nedeljno) trebalo bi proći kroz ceo postojeći kôd i promeniti ono što ne doprinosi jednostavnosi, jasnoći i čitljivosti kôda. Hijerarhija jednostavnog dizajna [1]:

- izvršiti sve testove,
- nema duplog kôda,
- jasno se iznosi namera koju je programer imao sa nekim kôdom,
- postoji najmanji mogući broj klasa i metoda.

3.3.5 Kolektivno vlasništvo nad kôdom

Jedna od bitnih novina koje donosi XP je ta da svaki član tima može u svakom trenutku da pristupi bilo kom delu programa i da ga menja, bez potrebe da bilo kome objašnjava svoje namere. Pri svakoj integraciji sprovodi se i testiranje čitavog sistema kako bi se otkrili eventualni propusti proistekli iz ovakvih izmena.

Ideja ove prakse je da treba dati svima mogućnost da menjaju sve delove programa, ukoliko im se učini da je to potrebno [35]. Ova praksa se površno može činiti kao pretnja po stabilnost i sigurnost softvera. Međutim, kolektivno vlasništvo nad kôdom u sprezi sa ostalim praksama XP-a, a pre svega sa stalnom integracijom, ne prouzrokuje ovakav rizik [3].

U XP-u svi preuzimaju odgovornost za ceo sistem. Ne znaju svi sve podjednako dobro, ali svi znaju bar po nešto o svakom njegovom delu. Ako programeri koji rade u paru uoče mogućnost poboljšanja koje bi im olakšalo rad, onda se to i sprovodi [1]. Ukoliko se napravi neki deo na komplikovan način, verovatno će ga neko iz tima pojednostaviti u nekom trenutku ili će se složiti da to mora da se uradi upravo na komplikovan način [3]. U svakom slučaju, svaki programer će dobro razmisliti o mogućim jednostavnim i elegantnim rešenjima, jer će i drugi pregledati i koristiti taj isti kôd.

Bez razvoja vođenog testovima i stalne integracije ova praksa nema nikakvu vrednost [3]. Testovi su bitni da ukažu na globalne greške koje nisu odmah uočljive, pri promeni nekog dela sistema,

a njihovo često sprovođenje pretvara rizike u prednosti, jer se svaki put saznaje nešto više o sistemu i to znanje se širi na ceo razvojni tim.

3.3.6 Razvoj vođen testovima

Prolaženje kroz niz testova nije dovoljna garancija da neće biti grešaka u isporučenom softveru, osim naravo ako je moguće testirati sve moguće kombinacije ulaza, čime će se zasigurno u mnogome uvećati broj potrebnih testova, kao i troškovi testiranja. Iz ovoga se može zaključiti da testiranje ne treba koristiti za dokazivanje pouzdanosti. Racionalnije je koristiti ga za presretanje grešaka prilikom njihovog nastanka.

Greške u softveru se svrstavaju u jednu od tri kategorija [3][35]:

- pogrešno u softveru,
- nedostaje u softveru,
- nepotrebno u softveru.

Jedino je kategorija pogrešno urađenog u softveru relevantna za otkrivanje grešaka pomoću testiranja. Najčešća strategija koja se koristi za odbranu od grešaka je metod verifikacije i validacije [3]. Verifikacija daje odgovor na pitanje: "U kojoj meri se softver realizuje na korektan način?", dok validacija odgovara na pitanje: "U kojoj meri se realizuje dobar softver?".

Različite vrste testiranja moguće je svrstati u neku od sledećih grupa [3]:

- testiranje modula (eng. *unit testing*),
- integraciono testiranje (eng. *integration testing*),
- sistemsko testiranje (eng. *system testing*),
- testiranje korisničke prihvatljivosti (eng. *acceptance testing*).

Testiranje modula je nasleđeno još iz proceduralnog programiranja. Objektno orjentisano progamiranje donosi nove probleme koji do tada nisu postojali, ali zadržava se isti termin. U osnovi testiranje modula nije ništa drugo do programi napisani za grupno izvršavanje i testiranje klasa. Najčešće se testiranje sprovodi nakon dizajna i pisanja kôda. Zna se da je testiranje nakon kodiranja jako teško, a kada se rokovi isporuke softvera približe onda se često testiranje preskače. Razvoj vođen testovima pokušava da reši ovaj problem pružajući bolji kvalitet softvera, na taj način što se testovi pišu pre pisanja samog izvornog kôda. Ovo predstavlja jednu od suštinskih praksi XP-a [35].

Integraciono testiranje je proces provere interakcija između sistemskih komponenti koje su već modularno testirane. To je kontinuirani proces u kom se akcenat stavlja na arhitekturu sistema, odnosno na nivo sistema na kom se vrši integracija pri čemu se niži nivoi sistema zanemaruju. Izuzetak su mali i jednostavni sistemi u kojima se testiraju sve komponente zajedno.

Tokom prethodna dva nivoa testiranja je većina funkcionalnih grešaka već otklonjena, te se sistemskim testiranjem ispituje kompletno ponašanje sistema sa posebnim osvrtom na identifikovanje funkcionalnih sistemskih karakteristika, kao što su sigurnost, brzina, pouzdanost, ... Na ovom nivou se uočavaju i testiraju interakcije softvera sa operativnim sistemima, različitim sklopovima hardvera, testira se rad u različitim uslovima, ...

Testiranje korisničke prihvatljivosti se kod klasičnog pristupa razvoju softvera sprovodi posle implementacije, kroz uočavanje nedostataka i dodavanje novih rešenja korisničkog interfejsa. Kod XP-a ovo testiranje se odvija za vreme realizacije projekta tako što klijent aktivno učestvuje u timu korišćenjem i testiranjem urađenog funkcionalnog dela softvera.

Umesto dizajniranja, kodiranja, pa tek onda testiranja, razvoj vođen testovima upućuje da se prvo počne testiranjem modula. Jednostavno rečeno, ne piše se ni jedna linija izvornog kôda sve dok se nedobije neuspeli test [3]. Redosled koraka je sledeći:

1. Piše se i pokreće test. Ovo je pvi neuspeli test. Javljaju se greške prilikom kompajliranja jer kôd koji se testira još uvek ne postoji.
2. Piše se osnovni skelet klase, kako bi se test mogao iskompajlirati.

3. Ponovo se pokreće test. Ni ovaj put test ne uspeva, iako se sada kompajlira.
4. Implementira se kôd klase za uspešni prolaz testa. Ova implementacija ne bi trebala da sadrži više kôda nego što je neophodno da bi se prošao test. Sve sigurnosne dodatke i vizuelna poboljšavanja korisničkog interfejsa treba izbeći u ovom koraku. Najbolje je slediti opšteprihvaćena pravila XP-a: KISS* i YAGNI.
5. Opet se pokreće test, koji ovaj put prolazi uspešno. Ukoliko ne prođe, vraća se na prethodni korak gde se vrše neophodne dorade i ispravke.
6. Ponovo se kreće sa novim testom i ponavlja se ceo proces. Sve dok se ne kompletira klasa ovaj ciklus se nastavlja dodajući sve kompleksnije i zahtevnije testove, što za rezultat ima novi kôd i novu funkcionalnost.
7. Tokom prethodnih koraka se često, kao rezultat dodavanja linija kôda kako bi test proradio, pojavljuje mnogo dupliranog i neelegantnog kôda koji ne sadrži dovoljno komentara [35]. Ovi propusti se prevazilaze refaktorisanjem, koji se može smatrati poslednjim korakom u ovom ciklusu.

3.3.7 Stalna integracija

Po pravilu, integraciju izvodi onaj par programera koji je završio predviđeni deo posla i u mogućnosti je da se posveti integraciji. Kent Beck predlaže da radna stanica za integraciju bude u sredini prostorije radi lakše komunikacije svih članova tima [1].

Stalna integracija se sastoji iz kompajliranja, otklanjanja grešaka, modularnog, integracionog, a često i sistemkog testiranja. Glavne prednosti stalne integracije su [3]:

- problemi integracije se pravovremeno otkrivaju i popravljaju, te nema panike u poslednjim momentima, pred istek rokova,
- rano upozoravanje na nepotpun ili nekompatibilan kôd,
- neposredno modularno testiranje za svaku promenu,
- dostupnost trenutnog izdanja za testiranje, prezentovanje ili objavljivanje.

Integracija se sprovodi nekoliko puta dnevno i to samo ukoliko su svi testovi uspešno proradili. Integracija sistema često može biti otežana ili preduga za često ponavljanje, ali ni tada se od nje ne treba odustajati [1][3].

3.3.8 Poboljšanje dizajna (refaktorisanje)

Refaktorisanje je vrsta reorganizacije izvornog kôda programa [2]. Svrha ove prakse XP-a je izmena strukture zapisa izvornog kôda kako bi bio lakši za razumevanje i jednostavniji za kasnije izmene, ali bez izmene funkcionalnosti softvera. Time se dobija prečišćen i pojednostavljen kôd koji je lako čitljiv i zahvalan za održavanje [2].

Neki od sigurnih pokazatelja da je na nekom mestu u kôdu potrebno refaktorisanje su višestruko pojavljivanje istih ili sličnih linija kôda, što zahteva uvođenje nove metode ili čak klase. Takođe, pokazatelj je i kada se pojavljuju konkretne vrednosti neke veličine na više različitih mesta, što se rešava uvođenjem nove konstante i njenom zamenom na odgovarajućim mestima. Bez obzira šta ukazuje na potrebu za refaktorisanjem, ne treba zaboraviti da sve celine kôda treba da budu propraćene odgovarajućim komentarima, shodno prihvaćenim standardima kodiranja.

U praksi se pokazalo da je refaktorisanje tokom programiranja u paru daleko efikasnije [3]. Dok jedan programer kodira, drugi prati, kontroliše i osmišljava još bolje varijante. Međutim, nije potrebno refaktorisati posle svake izmene u kôdu i nije uvek moranje refaktorisati sve dok ima šta da se refaktoriše, jer to često nema smisla, premda refaktorisanje nije samo sebi cilj.

* KISS (akronim, eng. *Keep It Simple, Stupid*) princip kaže "*Neka bude jednostavno, glupo*"

3.3.9 Programiranje u paru

Programiranje u paru obezbeđuje visok nivo povratne sprege, a delom pomaže i uspostavljanje visokog nivoa kvaliteta kôda. Ono predviđa istovremeni rad dva programera na jednoj radnoj stanici koji naizmenično pišu kôd. Dok jedan kodira, drugi progamer pažljivo prati rad, kontroliše sintaksu, predlaže rešenja i, po potrebi, komunicira sa ostatkom tima. Programeri bi trebalo, u periodima od svakih trideset minuta do jednog sata, da zamene uloge u paru.

Programiranje u paru predstavlja tehniku stalne kontrole dizajna, koja se odvija preko aktivnog pregledanja kôda [35]. Prednosti ovog mehanizma obaveznih praksi u XP-u su sledeće [29] [31][35]:

- povećana disciplina

  Prave se manje pauze u radu, jer se partneri smenjuju u radu.
- bolji kôd

  Programeri u paru zajedno ređe greše u dizajnu i sintaksi i teže kvalitetu, te je time i napisani kôd znatno bolji.
- manje ometanja

  Mnogo se bolje prihvataju ometanja tokom rada, jer jedan programer otklanja ometanja, dok drugi programira. Informcije se razmenjuju kako između programera u paru, tako i između ostatka tima.
- više programera doprinosi razvoju rešenja

  Ako je česta promena parova, više programera će biti uključeno u razvoj neke osobine i tako dati svoj dopinos.
- poboljšan moral

  Programiranje u paru je interesantnije od individualnog programiranja, te je moral poboljšan.
- kolektivno vlasništvo nad kôdom

  Kada svi u timu primenjuju ovu praksu i parovi se često menjaju, svi stiču pojam o celom kôdu, a ne samo o nekom njegovom delu.
- mentorstvo

  Ovo je najbezbolniji način da se pojedinačna znanja izmešaju.
- povezanost tima

  Ljudi se u timu bolje upoznaju kroz ovu praksu.
- jedna radna stanica manje

  Pošto par radi za jednim računarom, računar koji je preostao se može upotrebiti u druge svrhe.
- manja osetljivost na izostanke

  Praksom programiranja u paru se izjednačavaju znanja o sistemu, te je izostanak nekog iz tima bezbolniji.

Nedostaci [29][31][35]:

- Iskusnim programerima podučavanje neiskusnih u paru može biti otežavajuće.
- Mnogi programeri više vole da rade izolovano nego u timu, a posebno im može biti težak rad u paru.
- Razlike u stilu programiranja mogu stvoriti konflikte.
- Kada se radi od kuće u virtualnom timu ili kada zaposleni mora da radi van radnog prostora, teško je, a ponekad i nemoguće sprovesti ovu praksu.

Istraživanja koja su sprovedena u prethodnih desetak godina, pokazuju da su posle obuke za programiranje u paru, dva programera postala više od 200% produktivnija pri radu u paru, nego što bi

svako od njih bio da radi pojedinačno [15]. Oni rade oko 15% sporije, ali prave oko 15% manje grešaka [15][29] [30]. Pošto je debagovanje, često, nekoliko puta skuplje od osnovnog programiranja, nameće se logičan zaključak da je programiranje u paru isplativije [15] [28][31]. U prilog ovome, ukazuje još jedno sprovedeno istraživanje u kom je merena produktivnost programera početnika, sa jedne strane i iskusnih programera, sa druge strane, dok rade u parovima i pojedinačno [28]. Došlo se do zaključka da parovi programera početnika imaju znatno veću produktivnost nego ukoliko bi ti programeri početnici radili samostalno. Takođe, pokazalo se da je veća i produktivnost programera početnika u parovima, u odnosu na samostalni rad iskusnih programera [15][29][30].

3.3.10 Standardi u kodiranju

Standardi u kodiranju služe da olakšaju čitljivost kôda na osnovu opšteprihvaćenih programerskih pravila. Programiranje u paru zahteva visok nivo usklađenosti i uniformnosti pri razvoju softvera, jer programeri često nisu zadovoljni ako postoje stilske razlike u pisanju programskog kôda. Oni su u stanju da satima menjaju trivijalne razlike u tuđem kôdu, umesto da daju svoj doprinos kroz povećanje vrednosti projekta.

Postavljanjem zajedničkih standarda kodiranja obezbeđuje se da se sve ostale obavezne prakse lakše sprovode. Postoje čak i alati koji se koriste za restrukturiranje već napisanog kôda po predefinisanim standardima. Ovi alati su često integralni deo razvojnih okruženja. Na taj način programeri ne treba da troše vreme na definisanje ili predefinisanje standarda. Oni samo treba da prate uputstva, a svoju kreativnost i vreme upotrebe na smišljanje najboljih rešenja problema.

3.4 Upotreba razvojnog alata za ekstremno programiranje

Osnovna preporuka za izbor dobrog alata za razvoj softvera pomoću XP-a je princip malih početnih investicija i princip ”putovanja sa malo prtljaga”. Popularan alat u svetu XP-a su proizvodi organizacije Wikimedia. Oni omogućavaju distribuirani pristup beleškama i izveštajima, s tim da svako može sve da menja i da dodaje svoje komentare, što pruža nove mogućnosti i podstrek savremenom dizajnu. Za distribuiranu saradnju koriste se i MS NetMeeting ili IBM-ov Rational ProjectConsole. CVS (akronim, eng. *Concurrent Versions System*) je najšire korišćen alat za upravljanje razvojem kôda u XP-u. Koristi se za stalnu integraciju i praćenje promena u kôdu. Njegova alternativa u MS Visual studiju je Visual Source koji se pretežno koristi u .NET okruženju. Integrisana razvojna okruženja - IDE (akronim, eng. *Integrated Development Environment*) se najviše koriste u razvoju. Nivo integracije sa drugim alatom treba da bude presudan pri odabiru IDE-a. Za Java okruženje najčešće se koriste Eclipse, IDEA, JBuilder, JDeveloper, Sun ONE Studio i drugi, dok je za .NET neprikosnoven Microsoft Visual Studio. Takođe, sve je više besplatnih razvojnih okruženja koja čak imaju i znatne prednosti nad komercijalnim, kao što je to slučaj sa Eclipse razvojnim okruženjem. Postoje čak i papirni obrasci koji se u XP-u koriste, poput CRC (akronim, eng. *Class, Responsibilities, Collaboration*) kartica, koje se koriste kao pomoć pri projektovanju klasa, kao i indeksirane kartice za beleženje priča korisnika.

Nastanak razvoja vođenog testovima je vezan, kao i nastanak XP-a, za kraj 90-tih godina prošlog veka i za Kent Becka koji je napisao okruženje za testiranje za jezik SmallTalk za vreme angažmana na Chryslerovom C3 projektu i nazvao ga SUnit. Kasnije je zajedno sa Erichom Gama konvertovao ovaj alat za Java okruženje napravivši JUnit [3]. Danas postoji čitav spektar xUnit-a za različite jezike koji su slične strukture i namene. Ovaj besplatni alat se najčešće koristi za testiranje modula. Osnovne karakteristike su mu jednostavnost korišćenja i razdvajanje test primera od izvornog kôda programa. Alati iz porodice xUnit-a je široko korišćen za modularne (jedinične) testove koji čine jezgro XP-a. Ipak, osim njih koristi se i Rational Visual Test kompanije IBM, koji omogućava testiranje korisničkog interfejsa. U sprezi sa xUnit alatom najčešće se upotrebljava automatski skript alat poput ANT (akronim, eng. *Another Neat Tool*) koji omogućava testiranje celog projekta koristeći pri tom xUnit-e. Za ANT se pišu XML skriptovi koji daju mogućnost automatske integracije, kompajliranja, testiranja, a mogu izdavati i izveštaje različitih formata. Takođe, .NET programerima se savetuje korišćenje alata NANT, koji predstavlja verziju ANT-a prilagođenu za .NET platformu. Osim ovog

navedenog, koristi se alat za praćenje grešaka, obezbeđivanje kvaliteta kôda, praćenje performansi i drugo. Ponekad je potrebno angažovati osobu iz tima da radi na podešavanju i održavanju alata.

Tokom dnevnih stojećih (15-ominutnih) sastanaka treba smišljati nove ideje i rešavati postojeće probleme. Zbog toga se u te svrhe najčešće koristi tehnika „izbacivanja ideja“ poznatija kao brejnstorming (eng. *brainstorming*). Aktivnim učešćem članova tima odmah beleže sve ideje kojih se sete, jer im slobodno povezivanje ideja omogućava da budu kreativniji. Osnovni koraci brejnstorminga kojima se podstiče iznošenje novih ideja su:

- usmerenje na kvantitet

  Od članova tima se zahteva da se usmere na kvantitet i da pokušaju da daju što veći broj različitih ideja, ne opterećujući se njihovim kvalitetom. Pretpostavka je da što se više ideja iznese, onda je veća šansa dolaska do pravog rešenja.

- bez kritikovanja u startu

  Cilj je da se kreira atmosfera u kojoj će se članovi tima osećati prihvaćeno i slobodno da kažu sve svoje ideje, bez obzira koliko to bilo neuobičajeno i možda na prvi pogled smešno.

- kreativnost je potrebna

  Neobične i neočekivane ideje su veoma poželjne jer takve ideje mogu da pokrenu tim na sasvim nov način razmišljanja i gledanja na stvari, zahvaljujući čemu može da se dođe do neočekivanih i radikalnih rešanja.

- kombinovanje i poboljšavanje već iznetih ideja

  Kombinacija više različitih ideja može da stvori sasvim novu ideju koju je moguće unaprediti.

Naime, brejnstorming nije generisanje ideja za nekog drugog koji će izvršiti njihovu procenu i odabir. Naprotiv, tim generiše ideje za sebe, kako bi došli do one koja će predstavljati rešenje problema. U ove svrhe se često koristi softver za brejnstorming, poput MindManager, OpenMind, XMind, FreeMind, i mnogih drugih.

Podela alata u XP-u, premana nameni [1][3]:

- upravljanje i praćenje grešaka (u XP-u se koriste u vreme isporuke i prihvatanja):
  Rational ClearCase, Visual Intercept, Compuware TrackRecord, BlipIt, BugZilla, Blue Tail, Ticket Tracker elementool, Mantis, Code Track, Teamatic, ClearDDTS, Test Track
- integracija tima i umanjivanje ograničenja koja se javljaju kod fizički udaljenih timova:
  Wikimedia, Whiteboard, Paper, Project Web, NetMeeting, Rational Project Console
- automatizovano jedinično testiranje, testiranje grafičkog korisničkog interfejsa i testovi prihvatanja: *JUnit, ComUnit, VBUnit, NUnit, httpUnit, Rational Visual Test*
- obezbeđenje kvaliteta kôda, usklađenost sa standardima i mogućnost praćenja:
  DevPartner CodeRewiev
- merenje performansi sistema i komponenti:
  Jmeter, jUnitPerf, PerfMon, TrueTime, RealTime, MS Visual Studio Analyzer
- upravljanje konfigurisanjem izvornog kôda, uključujući i praćenje verzija i kontrolu promena:
  CVS, Visual Source Safe, PVCS, ClearCase
- integrisana razvojna okruženja (IDE) koja omogućavaju programerima da menjaju, debaguju, kompajliraju i sklapaju: *IBM VisualAge for Java, Microsoft Visual Studio .NET, Eclipse, Sun ONE Studio, JBuilder, JDeveloper, ControlCenter, IDEA, WebObjects, C++ Bilder, CodeWarrior*
- refaktorisanje:
  jFactor, IDEA, JBuilder, CodeMorpher, ControlCenter, RefactorIT, Eclipse, Refactoring Browser, Xrefactory, Retool, NET Refactorin, Sharp Developer
- priče korisnika, kratki dnevni stojeći sastanci, prezentacije:
  Word, Paint, PowerPoint, Wikimedia, Brejnstorming

Pri odabiru alata, prema mišljenju autora ovog rada, prvenstveno se treba voditi iskustvima članova tima, a zatim i preporukama XP-a.

3.5 Uvođenje ekstremnog programiranja

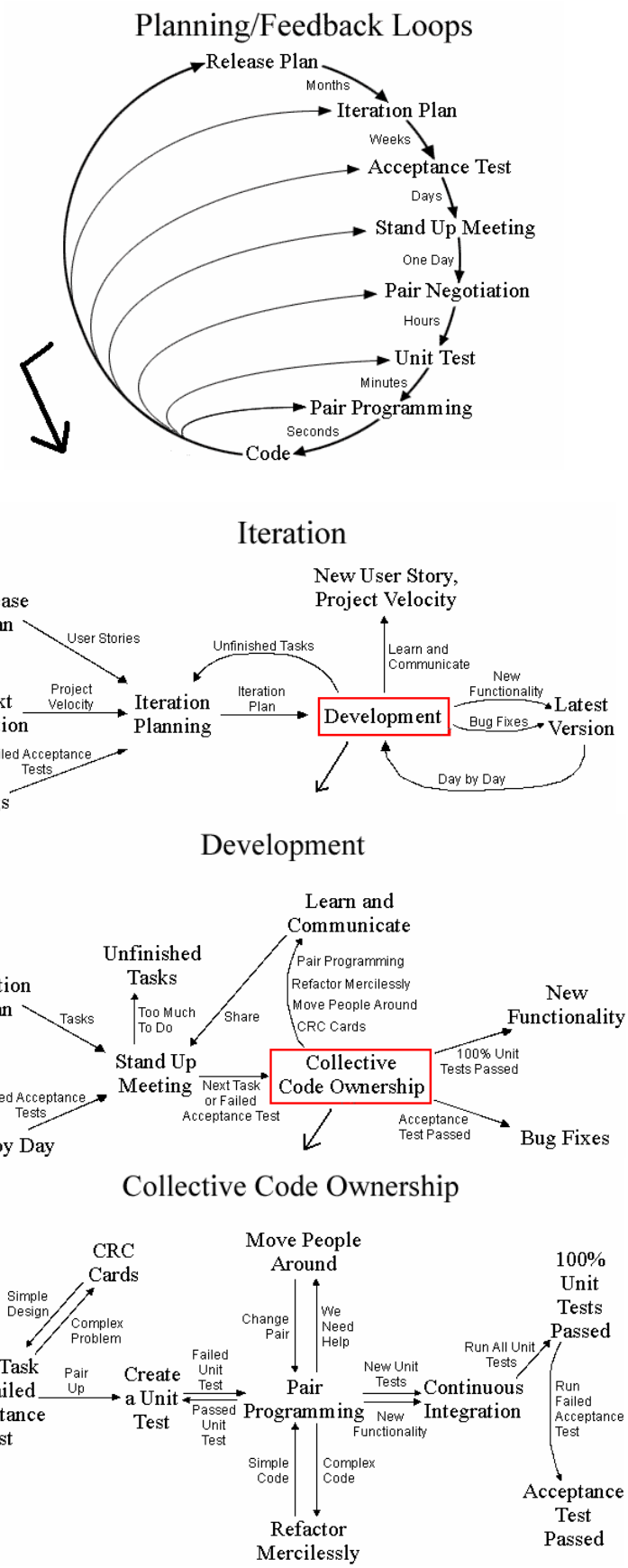
XP je rezultat analiza i prikupljanja ranije već korišćenih, ali nedovoljno iskorišćenih praksi i njihovo objedinjavanje u jednu celinu koja je sama po sebi nova. Sve ove prakse se nalaze svuda, u svakoj organizaciji, bez obzira da li se ona bavi softverom ili ne. Međutim, nijedna organizacija, koja ne razvija softver pristupom XP-a, ne koristi sve ove prakse objedinjeno kako bi se postigao maksimalni uspeh na projektu koji se realizuje.

Takav uspeh i jeste cilj kome XP teži. Do tog cilja nije ni malo lako doći. Potrebna je ekstremna posvećenost sprovođenju obaveznih praksi uz istovremeno poštovanje vrednosti i principa XP-a da bi se uspelo u ovoj nameri. Ipak, XP je istovremeno i kontradiktoran, dajući mogućnost slobodnom izgrađivanju i modifikovanju funkcija. Ova kontradiktornost proističe iz potrebe za fleksibilnošću koja zavisi od trenutne situacije, priznajući da je svaki softver poseban, a da je organizacija koja ga razvija još karakterističnija u odnosu na druge. To znači da se određene obavezne prakse koje su krojene za opšti slučaj, ponekad neće pokazati kao najbolje rešenje. U tom slučaju je na samoj organizaciji da osmisli bolji pristup, s tom razlikom da sada ima u praksi oproban šablon od koga može početi i u odnosu na koji može sagledavati rezultate internih istraživanja [3][36].

Kada organizacija koja se bavi proizvodnjom softvera želi da uvede XP javlja se čitav niz problema sa kojima se suočavaju oni koji pokušavaju da sprovedu ovu promenu. Koren tih problema je, uglavnom, u prirodnom otporu zaposlenih prema promenama.

XP unosi jedan broj suštinskih izmena, koje zahtevaju menjanje načina sagledavanja i shvatanja problema sa kojim se tim susreće u poslu. Ide se dotle da se u nekim slučajevima radi i skica radnog prostora, pa se zatim na optimalan način u radnom prostoru raspoređuje nameštaj i uređaji koji se koriste, u cilju veće produktivnosti i bolje radne atmosfere. Ništa ne sme da stane na put uspehu. Neadekvatan nametnuti nameštaj i nefunkcionalni prostor se preuređuju uz mnogo manje troškove nego što je trošak neuspeha posla. Ova promena koncepta rada često utiče i na razmatranje rukovodećih pozicija u samoj organizaciji. Neophodna su dodatna usavršavanja i prilagođavanja na nove uslove. Ovo je i najteže prevazići, ukoliko predstavlja problem pri uvođenju XP.

Ukoliko se želi preći na XP u postojećem projektu, treba doneti odluku da li sve početi iz početka ili ne. Kakva god da je ova odluka, XP se može sprovesti. Ukoliko se počinje iz početka, mora se više pažnje posvetiti ljudima koji na pravi način treba da prihvate novu praksu. Ukoliko se nastavlja već započeti posao, treba povesti računa o problemima kompatibilnosti i uložiti dodatne napore na povezivanje i istovremeno razdvajanje starog i novog dela kôda. Kvalitet se, po pravilu, poboljšava jer će sve što je urađeno od tog trenutka biti kontrolisano praksom XP-a, ali uz najbolju veru da je sve do tada urađeno dovoljno dobro. Međutim, ako i nije rađeno dovoljno dobro, delovi koji se čine pogrešnim se mogu refaktorisati i preraditi, u maniru razvoja vođenog testovima. Kada se prevaziđu ovi problemi, pažnja se ponovo usmerava ka ljudima koji sve ovo treba da sprovedu. Ni malo nije lako preneti ljudima sva znanja o XP-u. Uglavnom će se naići na otpor, ukoliko se vrednosti, principi, prakse i aktivnosti XP-a prenose zaposlenima kao imperativ rada. Sve treba da bude postepeno. Treba dopustiti ljudima da se prilagode, da se oslobode straha od nepoznatog i da prestanu sa odbijanjem promena. Ljudi se moraju angažovati na svim obaveznim praksama kako bi ih što pre prihvatili i kako bi se za njih više zainteresovali. Zbog toga se često održavaju radionice posvećene uvođenju XP-a u organizacije i što je moguće lakšem prihvatanju filozofije XP-a. Na tim radionicama, treneri tokom ovuke svojih timova demistifikuju XP kroz studije slučaja i prikaze XP-a u nekoliko slika. Na slici 17 je jedan od mogućih prikaza XP-a u nekoliko slika [22].



slika 17 – *XP u nekoliko slika*

U jednoj studiji slučaja [3], klijent je velika B2B (akronim, eng. *Business to Business*) kompanija koja je poručila novi web sajt za podršku njihovog modela za spajanje kupaca i snabdevača. Prva faza projekta je bila prema klasičnom *Waterfall* pristupu. Bilo je potrošeno i izvesno vreme za pripremu infrastrukture i planiranje. Rok je bio veštački postavljen, na bazi pogrešnih pretpostavki. Tim je zadovoljio rok, ali je kvalitet bio niži od očekivanog. U toku testa za prihvatanje bilo je više od 100 grešaka, tako da je tim nekoliko narednih nedelja proveo na stvarnom završetku programa, odnosno na otklanjanju tih grešaka.

U drugoj fazi razvoja projekta uveden je XP. Tim je počeo da koristi praktične preporuke. Vođa tima je toliko značaja pridavao programiranju u paru da je naredio da je svi, bez izuzetka primenjuju. Programiranje u parovima, standardi u kodiranju, prethodno testiranje, vodili su ka poboljšanju kvaliteta softvera. Na kraju ciklusa od tri meseca bilo je samo dve manje greške koje su otkrivene kod testova za prihvatanje, što je ogroman napredak u odnosu na više od 100 grešaka iz prve faze.

Kupcima se dopala kontrola koju su imali u procesu XP-a, kao i stalna vidljivost projekta. I kupci i snabdevači su projekat označili kao veoma uspešan. Programeri su bili zadovoljni kvalitetom kôda, kao i privrženošću tom kôdu. U ovom primeru koji ilustruje prelazak na XP u toku trajanja projekta, uočava se da su najvažniji resursi projekta ljudi, koje treba na obazriv način uvesti u XP.

U sledećoj studiji slučaja [3], kompanija *Symantec* iz Kalifornije, je nezavisni proizvođač softvera, poznata po tehnologijama vezanim za sigurnost, pomoćne programe i upravljanje na daljinu. *Symantec* je za razvoj softvera koristio Pristup usmeren ka rešenjima. Međutim, *Symantec* je ubrzo preuzeo kompaniju *Axent Technologies* koja je već koristila XP. Tako je XP ušao u Symantec.

Razvojni tim, koji je radio na novom sigurnosnom projektu Orca, baziranom na Javi, prihvatio je XP i dao mu prednost u odnosu na druge pristupe razvoja. Jedna od praktičnih preporuka XP-a, celokupnost tima, preporučuje da kupac treba da bude na licu mesta. Ovo je veoma teško ostvariti kada se razvija komercijalni softver. Ulogu pseudo kupca je u ovom slučaju imao interni menadžer proizvoda. Međutim, XP projekat je zapao u probleme jer se menadžer proizvoda nije uklopio u XP tim. Time je timu nedostajao pravi smer jer nije znao šta kupac zaista želi. Zbog toga je XP tim u nastavku realizacije projekta morao da se preorjentiše na saradnju sa osobom koja je bila zadužena za osiguranje kvaliteta komercijalnog softvera u okviru *Symantec*-a. Uloga osiguranja kvaliteta u XP projektu je bila dvostruka. Sa jedne strane, to je bilo pisanje testova za prihvatanje, a sa druge, obezbeđivanje povratnih informacija vezanih za korisnika. Programeri su radili sa svojim alatima za automatsko testiranje, kao što je *JUnit*. Bili su usresređeni na testiranje na nivou koda. Osoba zadužena za osiguranje kvaliteta je morala da specificira testove, kad god su oni mogli da smanje razlike između testova koje su radili programeri i izlaza koji bi bio zadovoljavajući po njenom mišljenju. Na taj način je XP projekat uspešno realizovan u predviđenom roku i sa očekivanim kvalitetom i funkcionalnošću.

Ovakva situacija u XP timu, ukazuje da će, ako nema kupca na licu mesta, timu nedostajati pravi smer i fokus rada. Međutim, pokazalo se da se i osobe zadužene za osiguranje kvaliteta mogu uklopiti u tim. Stoga treba očekivati mešanje postojećih tehnologija i znanja sa praktičnim preporukama i obaveznim praksama XP-a.

3.6 Proširenje primene ekstremnog programiranja

XP je počeo kao pristup koji je pogodan za primenu u razvoju softvera u timovima male i srednje veličine. Međutim, postavljaju se pitanja:

- Na koji način se XP može primeniti kod velikih razvojnih timova koji broje po nekoliko desetina, čak i stotina članova?
- Na koji način se mogu realizovati virtualni i distribuirani razvojni poduhvati fizički razdvojenih timova?
- Kako se izboriti sa rizičnim poduhvatima od kojih zavise širi razvoji?

U takvim velikim distribuiranim razvojnim projektima, XP se primenjuje u formi proširenja koje je poznato pod nazivom industrijsko ekstremno programiranje – IXP (akronim, eng. *Industrial Extreme Programming*).

IXP nastaje 2004. u konsultantskoj firmi Cutter, gde se kao autor pominje Joshua Kerievsky. Na slici 18 je prikazano pet vrednosti (komunikacija, zadovoljstvo, učenje, jednostavnost i kvalitet) koje mogu da variraju od tima do tima [33]. Ovde se umesto dvanaest obaveznih praksi iz XP-a pojavljuje čak dvadeset tri. Nekima je promenjen naziv, neke su dodate, a neke su ostale iste kao i u ranijoj verziji XP. Nove su [3]:

- pripremljenost za pristup,
- projektna zajednica,
- ugovaranje projekta,
- menadžment vođen testovima (SMART princip),
- retrospektiva,
- stalno učenje.

Poboljšane postojeće prakse [3]:

- razvoj vođen testovima priče – SDD (akronim, eng. *Storytest Driven Development*) predstavlja unapređenje TDD-a,
- dizajn vođen domenom – DDD (akronim, eng. Domain Driven Design) predstavlja unapeđenje metafore u sistemu,
- uparivanje predstavlja unapređeno programiranje u paru,
- iterativno testiranje korisnosti predstavlja unapređeno prisustvo klijenta u timu.

slika 18 – *Sadržaj IXP-a*

Nepromenjene obavezne prakse [3]:

- refaktorisanje,
- igra planiranja,
- stalna integracija,
- zajedničko vlasništvo nad kôdom,
- standardi u kodiranju,
- održivi korak,
- česte isporuke.

Implicitne nepromenjene obavezne prakse koje postaju eksplicitne [3]:

- stalno upravljanje rizikom,
- evolutivni dizajn,
- pričanje priče (korisnički zahtevi),
- testiranje priče,
- rad u zajednici,
- mali timovi.

Sve novine su došle iz prakse, analizom problema koji su se javljali tokom korišćenja XP-a. Ipak, treba biti obazriv jer nepoznavanje osnovnog koncepta XP-a vodi neuspehu u implementaciji industrijske varijante ovog pristupa.

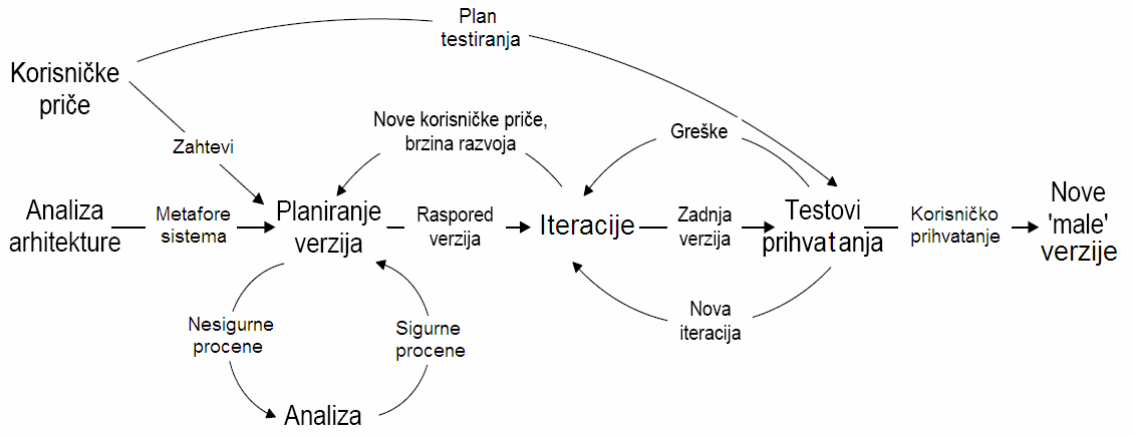
IXP je postao dosta složen u svojim preporukama, ali obećava značajna poboljšanja. Ovo usložnjavanje bi trebalo da prati značajno kvalitativno poboljšanje implementacije samog pristupa u postojeće organizacije, dok sprovođenje pristupa u praksi, kada jednom zaživi, i nije mnogo različito od klasičnog XP-a.

Sa aspekta razvoja XP-a može se videti da ovaj novi pristup u razvoju softvera još uvek traži svoj konačni oblik kroz evoluciju elemenata koji ga definišu. Mnoge njegove obavezne prakse su se razvile iz standardnih praksi XP. U IXP-u se vrednosti dinamički određuju u svakom timu na osnovu dogovora. Ovakav pristup je u neku ruku isuviše radikalan, zato što, kao što je i prikazano u ovom radu, prakse direktno proizilaze iz vrednosti i omogućavaju njihovo sprovođenje. Dinamičkim dodeljivanjem vrednosti, sve praktične mehanizme njihovog sprovođenja trebalo bi preispitati i redefinisati u duhu izmenjenih vrednosti.

4 STUDIJA SLUČAJA – Primer dobre prakse

Grupacija hotela je angažovala kompaniju za razvoj softvera da realizuje projekat razvoja veb lokacije za promociju i rad hotela Grupacije koja do tada nije postojala [26]. Iako se ovde radi o projektu srednje veličine, ipak postoji problem geografske udaljenosti predstavnika Grupacije hotela. Zbog toga je napravljen sajt na kom se blagovremeno postavljaju svi detalji koji se tiču realizacije projekta i njegovog napredovanja. Na kraju svake radne nedelje održavaju se *online* sastanci predstavnika Grupacije sa vođom projekta i članovima razvojnog tima. Na ovim jednočasovnim sastancima, utvrđuje se napredak projekta i verifikuju svi zadaci koji su odrađeni tokom te radne nedelje. Takođe, govori se i o zadacima koji se planiraju za narednu nedelju, i o čijoj uspešnosti realizacije će se govoriti za sedam dana. Pored ovih *online* sastanaka, svaki radni dan članova tima sa vođom počinje petnaestominutnim stojećim sastancima na kojima se govori o projektnim zadacima predviđenim za realizaciju tog dana.

U nastavku će biti prikazana realizacija projekta kroz korake adekvatnog upravljanja, pisanja i kontrole realizacije projektnog plana i koraka razvoja softvera primenom ekstremnog programiranja (videti sliku 19).



slika 19 – *Osnovna shema XP-a*

1 Plan projekta

Plan projekta sadrži sledeće delove:

- opšti pregled
- članovi tima
- zahtevi
- raspored projektnih zadataka
- resursi
- planovi implementacije
- planovi podrške

1.1 Opšti pregled

Svrha projekta je realizacija veb lokacije Grupacije hotela u čijem su vlasništvu četiri hotela u atraktivnim turističkim mestima. Pored brojnih zabavnih sadržaja, svaki hotel je specijalizovan za određeni vid turizma. Da bi se uklopila u zabavne sadržaje koje hoteli pružaju gostima, veb lokacija ima grafički prikaz virtuelne slot mašine. Posetioci veb lokacije imaju priliku da klikom miša na predviđeno dugme pokreću bubnjeve virtualne slot mašine. U zavisnosti od rezultata, posetioci će moći da osvoje jednu od nekoliko nagrada koje je Grupacija hotela obezbedila.

Grupacija hotela takođe u svojoj ponudi ima i usluge partnerskog avio prevoznika, tako da se u realizaciji veb lokacije mora obuhvatiti poslovna relacija između dva entiteta. Posetiocima veb lokacije Grupacije hotela je na raspolaganju link ka sajtu avio kompanije, na kom postoji mogućnost da se rezervišu letovi ka destinacijama hotela Grupacije i po povoljnijim uslovima bude gost nekog od hotela Grupacije. Avio kompanija daje svoj doprinos u finasiranju realizacije veb lokacije Grupacije hotela. Međutim, budžet projekta neće predstavljati polje interesovanja vođe projekta, jer se radi o unapred dogovorenoj fiksnoj ceni realizacije veb lokacije.

Smisao celog projekta jeste da se potencijalni gosti hotela Grupacije navedu da elektronskim putem rezervišu boravak u nekom od hotela i da pri tome koriste i usluge avio kompanije, poslovnog partnera Grupacije hotela.

1.2 Članovi projektnog tima

Tim na ovom projektu čine: vođa projekta, administrator baza podataka, dva grafička dizajnera, veb dizajner-programer, pet veb programera, predstavnik Grupacije hotela i IT menadžer avio kompanije. Ukupan broj članova projektnog tima je 12, od čega je 6 programera kojima po potrebi pomaže vođa projekta, 2 dizajnera različitih kompetencija, kao i jedan specijalista za grafički dizajn koji je angažovan po ugovoru za realizaciju ovog projekta.

1.3 Zahtevi

Zahtevi predstavnika Grupacije hotela koji moraju biti realizovani u okviru veb lokacije:

- Početna stranica veb lokacije ima grafički prikaz vitrualne slot mašine koja je slična pravim slot mašinama koje se viđaju u kockarnicama. Posetilac veb lokacije će, nakon provere njegove starosti, moći da pokrene bubnjeve virtualne slot mašine klikom miša na dugme „Okreni bubanj“. Svaki bubanj može sadržati različite simbole, koji će odgovarati marketinškim i veb temama hotela ove Grupacije.
- Poput prave slot mašine u kockarnicama, i na virtualnoj mašini moraju postojati kombinacije simbola koje donose nagrade koje su grupisane u tri različite kategorije. Virtualna slot mašina će uvek dati neku od pobedničkih kombinacija, kojom će se odrediti vrsta nagrade koju posetilac veb lokacije Grupacije hotela dobija.
- Kombinacija tri zvezde donosi najveći dobitak na virtualnoj slot mašini – glavnu premiju. Srećni posetilac veb lokacije dobija sedam dana za dve osobe u jednom od hotela, po izboru dobitnika, sa plaćenim troškovima – smeštaj u hotelu, boravišne takse i doručak u restoranu odabranog hotela u formi švedskog stola. Dobitnik takođe dobija i 50% popusta na usluge avio kompanije, partnera Grupacije hotela. U toku svakog meseca može biti dodeljena samo jedna glavna premija. Automatski će biti objavljeno na veb lokaciji čim glavna premija bude dodeljena i nastaviće se dalje sa radom i dodelom ostalih nagrada narednim posetiocima.
- Postoji osam prvih nagrada čija je vrednost manja u odnosu na glavnu premiju, ali je istovremeno veća u odnosu na nagrade druge kategorije. Prvu nagradu donosi bilo koja kombinacija tri ista simbola na virtualnoj slot mašini, izuzev tri zvezde koje su rezervisane za glavnu premiju. Programeri moraju da obezbede ispravno funkcionisanje ovog dela veb lokacije, jer se očekuje da će istovremeno biti više hiljada posetilaca. Prva nagrada je 20% popusta na celokupan račun realizovanih hotelskih usluga tokom boravka u nekom od hotela Grupacije. Biće podržane po dve takve nagrade u svakom od hotela. Dobitnik može izabrati nagradu iz liste preostalih nagrada, koje još uvek nisu dodeljene.

- Druga nagrada je znatno manje vrednosti. Predviđeno je da se podeli 20 ovakvih nagrada. Potrebno je imati kombinaciju dva ista simbola i posetioci koji uplate bar jedno noćenje dobijaju dodatno noćenje u nekom od hotela Grupacije.
- Glavna premija i svaka pojedinačna prva nagrada mogu biti dodeljene samo jednom u toku meseca. Po završetku pisanja kôda i uspešnog testiranja, klijentu treba predati dokument koji će biti garancija da svaka od ovih nagrada može biti dodeljena samo jednom.
- Posetiocima je dopušteno da posećuju veb lokaciju Grupacije kad god to požele, ali u nagradnoj igri mogu da učestvuju samo jednom. Drugim rečima, posetioci mogu doći na veb lokaciju neograničen broj puta, ali virtualnu slot mašinu mogu pokrenuti samo jednom. Zbog toga postoji stranica za prijavljivanje posetilaca prilikom ulaska na lokaciju. Pre pokretanja slot mašine biće upoređeni JMBG posetilaca, potencijalnih gostiju hotela, sa JMBG prethodnih dobitnika. Ukoliko se takav JMBG već nalazi na listi dobitnika, biće onemogućena dalja dodela nagrada.
- Svaki hotel ima svoj deo veb lokacije u okviru kog posetioci mogu da pogledaju izglede soba, restorana, bazena, teretana, kazina, sauna, ...
- U delu veb lokacije „Rezervacije“, posetioci mogu da rezervišu sobe u hotelima Grupacije. Tada će im biti zatraženo da unesu broj platne kartice, kako bi se uspešno izvršila naplata rezevacije sobe.
- U okviru dela „Rezervacije“, ispod dela za rezervaciju soba, nalazi se deo u okviru kog posetioci mogu da rezervišu i kupe avionske karte avio kompanije, partnera Grupacije hotela. Tada će im biti zatraženo da unesu broj platne kartice kako bi se uspešno realizovala naplata. Ovaj deo veb lokacije mora biti povezan sa avio kompanijom sa kojom Grupacija ima saradnju o realizaciji popusta na avio karte za goste koji odsedaju u hotelima Grupacije.
- Veb lokacija mora biti realizovana za 100 dana (uključujući i neradne dane poput vikenda, državnih i verskih praznika), kako bi bila prezentovana na proslavi godišnjice poslovanja Grupacije hotela.
- Ispunjenje pojedinačnih zahteva biće provereno korisničkim prijemnim testom (eng. *user acceptance testing*), kojim će se proveriti kako posetioci prihvataju sadržaje veb lokacije. Smatraće se da su zahtevi ispunjeni ukoliko testovi postignu 100% prolaznost.
- Po završetku uspešne realizacije projekta, programeri su u obavezi da adresu sajta postave na glavne Internet pretraživače, poput Google-a, Yahoo-a, Excite-a, ...

1.4 Raspored projektnih zadataka (WBS)

1. korak - Vođa projekta u razgovoru sa predstavnikom Grupacije hotela utvrđuje koji su krajnji rezultati projekta. Vođa projekta se koristi kombinacijom ispitivanja, diplomatije, detektivskog nastupa i psihologije kako bi utvrdio šta predstavnik Grupacije zaista želi.

2. korak - Naoružan informacijama dobijenim od predstavnika Grupacije hotela, vođa projekta počinje da uočava potrebne zadatke i utvrđuje faze projekta. To izvodi dekompozicijom krajnjih rezultata projekta u smisaone zadatke.

3. korak - Kada vođa projekta shvati koje projektne zadatke treba izvršiti, time su definisani svi elementi (događaji, zadaci i faze) projekta, koji čine WBS dijagram (videti sliku 20). To dalje ukazuje koliko je potrebno ljudi, resursa i vremena, kao i gde treba postaviti oznake zavisnosti između zadataka na gantogramu (videti sliku 20).

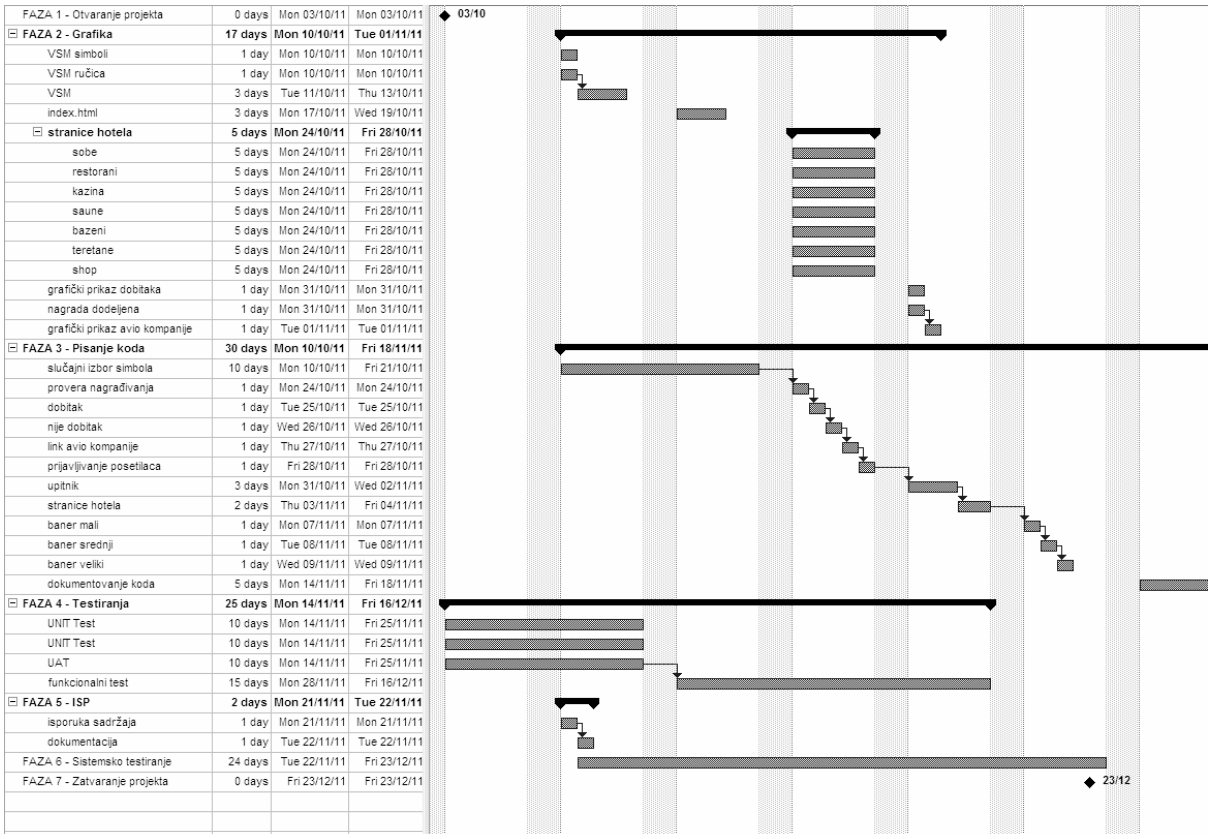
1.5 Resursi

Za realizaciju projekta, razvojni tim koristi sledeće resurse[*]:

- 7 radnih stanica, od čega po jedna za vođu projekta i administratora baza podataka, dve stanice za grafičke dizajnere i tri stanice za programere, jer programiraju u paru,

[*] navedeni resursi su opšteg tipa, jer njihovo preciziranje zavisi od kompetencija i iskustva vođe projekta i projektnog tima, kao i od pojedinačnih jedinstvenih zahteva svakog konkretnog specifičnog projekta

- opremu za pravljenje profesionalnih fotografija visokih rezolucija,
- softver za profesionalnu obradu fotografija i grafičkih elemenata,
- softver za upravljanje projektom,
- softver za intranet sajt, kao lokalna podrška projektu koji se razvija,
- programerski alat i razvojna okruženja,
- servere za testiranje funkcionalnosti veb lokacije u lokalu,
- softver za različite vrste testiranja,
- softver za distribuirano dokumentovanje.



slika 20 – *Gantogram rasporeda projektnih zadataka (WBS)*

1.6 Planovi implementacije

Zbog prirode virtualne slot mašine, najveći deo dizajnerskog posla grafički dizajneri moraju da obave pre pisanja kôda. Kada se postave sve grafičke komponente, programeri mogu da kreiraju logičke module koji će upravljati virtualnom slot mašinom. Takođe, treba uzeti u obzir i linkove ka avio kompaniji, poslovnom partneru Grupacije hotela, i ka Internet provajderu – ISP (eng. *Internet service provider*). Faze testiranja su takođe deo projekta (videti sliku 20).

Razvojni plan ima tri sloja:

- izrada grafičkih komponenata za virtualnu slot mašinu,
- pisanje kôda koji omogućava funkcionisanje virtualne slot mašine,
- hostovanje sadržaja veb lokacije Grupacije hotela na serveru ISP-a.

Sav hardver se nalazi kod ISP-a, te ne treba brinuti o kreiranju servera i administriranju. Treba osigurati da kada se kôd postavi na veb server ISP-a, veb lokacija funkcioniše ispravno. ISP je zadužen za sigurnosne aspekte veb lokacije. Međutim, programeri su u obavezi da na sve datoteke

postave NTFS dozvole tako da posetioci veb lokacije imaju samo mogućnost čitanja (eng. *read-only*) svih podataka, a da pri tome virtualna slot mašina ispravno funkcioniše. ISP tim konstantno nadgleda svoju mrežu i osigurava je od upada bilo koje vrste.

Plan testiranja je neuobičajen kod ove veb lokacije. Najpre će programeri testirati stvarni kôd kako bi se proverilo da li ispravno funkcioniše. Potom će se testirati cela veb lokacija sa korisnicima, kako bi se proverilo da li korisnik može da vidi sve što bi trebalo. Međutim, za kompletno testiranje modula koji se koristi za simulaciju rada bubnjeva virtualne slot mašine, da bi se osiguralo da neće doći do dupliranja dobitnika glavne premije, koristiće se softver koji simulira milione pogodaka veb lokacije.

1.7 Planovi podrške

U planu je da programeri pruže direktnu podršku za sve probleme koji budu nastali u vezi sa kôdom.

2 Upravljanje rizikom

Procenu rizika, zbog kompleksnosti potencijalnih problema, vođa projekta morao uraditi zajedno sa predstavnikom Grupacije hotela. Svi potencijalni rizici su objedinjeni u dokumentu koji će imati i vođa projekta i predstavnik Grupacije. Svi rizici su vrednovani na skali od 1 do 10, kako po pitanju verovatnoće pojave, tako i po pitanju opasnosti, pri čemu su sa 10 predstavljeni najverovatniji i najopasniji rizici:

- Moguće je da veb lokacija ne bude poznata širokom krugu korisnika Interneta, zbog slabe posećenosti. Ne postoji pravi odgovor na ovaj rizik, izuzev čekanja da vest o postojanju veb lokacije vremenom dođe do širokog kruga korisnika Interneta i zainteresuje ih. Rizik može biti ublažen objavljivanjem reklama u medijima. Verovatnoća rizika je 7, a opasnost od rizika je 2.
- Nagrade možda neće biti dovoljno atraktivne za posetioce. Verovatnoća rizika je 5, a opasnost od rizika je 5.
- Veb lokacija možda neće biti efikasna iz bilo kog razloga – kompleksna grafika, jasnoća prikaza, upotrebljene boje, odgovarajuća brzina, ... Veb lokacija će imati poseban deo koji će prikupljati statistiku o posećenosti. Verovatnoća rizika je 2, opasnost od rizika je 7.
- Saradnja između Grupacije hotela i ISP može postati kritična i nejsana u bilo kom trenutku. Rizik može biti ublažen i sprečen redovnim kontaktima delegiranog predstavnika Grupacije hotela sa ISP. Verovatnoća rizika je 2, opasnost od rizika je 5.
- Problemi sa proizvođačima pravih slot mašina. Ovaj rizik je nebitan po prirodi i do njega verovatno neće ni doći. Verovatnoća rizika je 0, opasnost od rizika je 7.

3 Upravljanje promenama

Upravljanje promenama, kada više programera zajedno piše kôd, zahteva upotrebu odgovarajućeg softvera za kontrolu verzija. Time se sprečava situacija da jedan programer unese izmene u kôd, a da drugi programer te izmene poništi već pri prvom narednom susretu sa takvim kôdom. Dve stvari su potrebne za rešenje ovog problema: dokumentovanje svih implementiranih izmena, i lokacija sa koje se mogu kontrolisati verzije budućeg softvera. Svakodnevne izmene koje se unose iz dana u dan tokom pisanja kôda moraju biti dokumentovane, i to na dva mesta: prvo – u knjizi izmena na sajtu projekta, i drugo – u samom kôdu. Novoizgrađeni kôd mora biti propraćen odgovarajućim komentarima, ali se podaci o izmenama (opis izmena, programer koji ih je uneo, vreme izmena, ...) moraju unositi i u neku vrstu posebnog zaglavlja. Praćenje različitih verzija kôda tokom razvoja se može olakšati upotrebom odgovarajućeg softvera, koji predstavlja skladište (eng. *repository*) kako kôda tako i mehanizama za kontrolu verzija. Jedan od takvih programa je i *Visual SourceSafe* kompanije *Microsoft*. Postoji veći broj sličnih programa, ali ovaj može poslužiti kao dobar primer, ako se uzme u obzir njegova zastupljenost u praksi.

4 Upravljanje kvalitetom

Od tri osnovna ograničenja svakog projekta – vreme, kvalitet, budžet, u ovom konkretnom slučaju kvalitet predstavlja glavno ograničenje ovog projekta. To je lako objašnjivo, jer Grupacija hotela mnogo polaže na kvalitet i zadovoljstvo svojih gostiju. Bilo kakvo, pa i najmanje narušavanje vizuelnog i funkcionalnog kvaliteta nije prihvatljivo, jer kada je reč o hotelima prvo što potencijalni gost uočava jeste vizuelni sklad a potom i funkcionalnost i udobnost. Premda je veb lokacija prvi sustret potencijalnog gosta sa hotelom, jasno je zašto na veb lokaciji ne sme biti propusta i nedostataka i zašto je kvalitet osnovno ograničenje ovog projekta.

Kontrola kvaliteta se prati tokom realizacije projekta, putem određenih kontrolnih tačaka. Vođa projekta će dokumentovati sve realizovane kontrole.

Sa stanovišta kontrole kvaliteta, vođa projekta je zadovoljan onim što su grafički dizajneri odradili. Grafički prikaz virtualne slot mašine, njena ručica i različiti elementi koji se pojavljuju u bubnjevima mašine izgledaju zadovoljavajuće. Kôd koji posetiocima veb lokacije omogućava pokretanje virtualne slot mašine dočarava rad prave mašine u kazinu. Međutim, vođu projekta su brinula dva različita aspekta virtualne slot mašine: testiranje na daljinu i testiranje opterećenja. Postavlja se pitanje da li će virtualna slot mašina funkcionisati na isti način kada se neko na veb lokaciju povezuje u lokalu i kada neko ko je geografski veoma udaljen virtualno zavrti bubnjeve slot mašine preko Interneta? Drugo pitanje je šta se dešava ako stotine ili hiljade posetilaca veb lokacije istovremeno pokuša da pokrene virtualnu slot mašinu? Da li će se ponašati isto kao kada postoje jedan ili dva korisnika istovremeno? Ovakve nedoumice su rešene pomoću softvera za testiranje opterećenja. Zaključeno je da virtualna slot mašina dobro radi, iako je bilo situacija u kojima je postojalo malo kašnjenje kod predstavljanja rezultata pokretanja mašine. Stoga je vođa projekta programere uputio da još jednom provere kôd i pogledaju da li postoji još nešto što je moguće popraviti i možda ubrzati rad bubnjeva slot mašine.

Test opterećenja virtualne slot mašine pokazao je da se rad bubnjeva virtualne slot mašine usporava kada istovremeno postoji više od 8000 korisnika. Ovakvi rezultati testa opterećenja smatraju se zadovoljavajućim, jer statistika ISP-a potvrđuje da na veb lokacijama iste ili slične sadržine u proseku bude do hiljadu istovremenih poseta.

5 Realizacija projekta po fazama

Ceo projekat razvoja veb lokacije Grupacije hotela biće realizovan po fazama projekta prikazanim u gantogramu rasporeda projektnih zadataka (videti sliku 20). Pri tome će se slediti postupak razvoja softvera primenom praktičnih preporuka, vrednosti, principa, praksi i aktivnosti XP-a (videti sliku 21) koje su dogovorile zainteresovane strane u projektu.

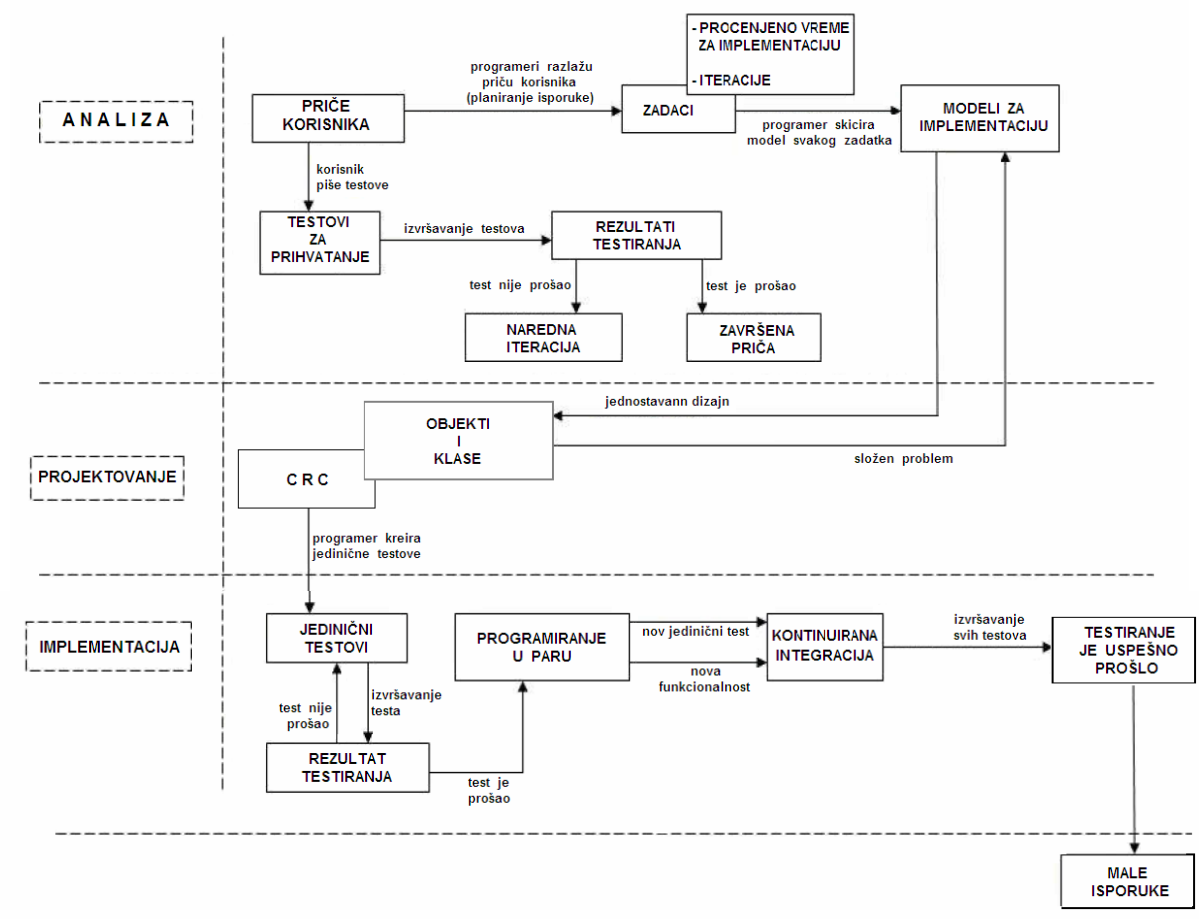
Druga faza obuhvata projektne zadatke vezane za izradu grafike za virtualnu slot mašinu:

- grafička rešenja za simbole koji se pojavljuju na sva tri bubnja virtualne slot mašine,
- grafički prikaz virtualne slot mašine,
- grafičko rešenje ručice virtualne slot mašine u nekoliko različitih položaja,
- grafika potrebna za početnu stranicu veb lokacije Grupacije hotela,
- grafika potrebna za stranice na kojima će biti prikazana unutrašnjost i spoljašnjost hotela Grupacije, zajedno sa izgledom soba, kazina, restorana, sauna, bazena, teretana, ...
- grafički prikaz dobitaka,
- grafički prikaz stranice koja obaveštava da je nagrada već dodeljena,
- grafički prikaz avio kompanije, poslovnog partnera Grupacije hotela.

Treća faza obuhvata projektne zadatke vezane za pisanje odgovarajućeg kôda veb lokacije:

- programiranje modul koji će slučajnim izborom odabrati tri grafička simbola i postaviti ih na vitrualnu slot mašinu,
- programiranje modula kojim će se proveriti da li je ostvaren dobitak na vitrualnoj slot mašini,

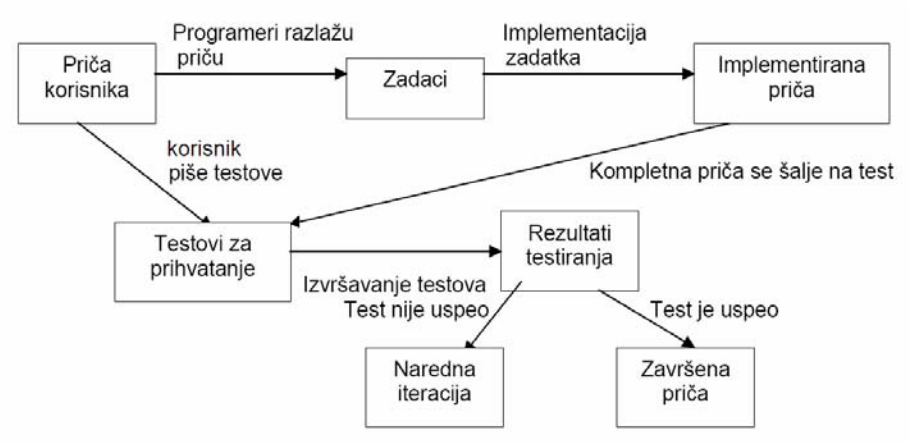
- pisanje kôda koji obaveštava posetioca veb lokacije o dobitku,
- pisanje kôd koji obaveštava posetioca da nije ništa dobio na virtualnoj slot mašini,
- programiranje dela veb lokacije u okviru kog će se posetiocu ponuditi upitnik, pozvati ga da ga popuni i time postane učesnik u izvlačenju jedne od tri nagrade – vikend noćenje u jednom od četiri hotela Grupacije, po izboru dobitnika nagrade (Prikupljeni podaci se smeštaju i čuvaju u bazi podataka. Svim učesnicima u popunjavanju upitnika zagarantovana je tajnost podataka koja je iskazana na stranici upitnika.),
- pisanje kôda koji će posetioce preusmeriti na veb sajt avio kompanije sa kojom Grupacija hotela sarađuje,
- realizacija odgovarajućih poslovnih veza – B2B (akronim, eng. *business-to-business*) prema virtuelnoj privatnoj mreži avio kompanije, radi provere statusa letova i cena,
- programiranje stranice za prijavljivanje posetilaca,
- programiranje modula koji će posetiocima predstaviti sobe i ostale karakteristike hotela Grupacije,
- izraditi reklamne banere hotela Grupacije koji će biti postavljeni na različitim veb sajtovima,
- dokumentovati kompletan kôd.



slika 21 – *Postupak razvoja softvera primenom XP preporuka*

Četvrta faza je predviđena za testiranje prema praktičnim preporukama ekstremnog programiranja:

- parcijalna testiranja programskih modula,
- korisnički prijemni testovi (videti sliku 22),
- funkcionalna testiranja.



slika 22 – *Tok rada testa za prihvatanje*

Peta faza obuhvata projektne zadatke vezane za puštanje u rad veb lokacije Grupacije hotela:

- isporuka pripremljenog sadržaja veb lokacije ISP-u,
- dokumentovanje svih informacija o veb sajtu - IP adrese, DNS, ...

Šesta faza je predviđena za sistemsko testiranje veb lokacije Grupacije hotela.

6 Zatvaranje projekta

Tokom realizacije projekta, najveći deo zadataka vođe projekta je posvećen kontrolisanju realizacije projekta. Pored svakodnevnih kraćih sastanaka sa projektnim timom, vođa projekta mora da na kraju svake nedelje saziva sastanak predstavnika Grupacije hotela, IT menadžera avio kompanije i projektnog tima, kako bi se na tom sastanku dobile potvrde za novonastale izmene na projektu i verifikovali do tada urađeni delovi projekta.

Već na početku projekta, tokom delegiranja projektnog tima, vođi projekta je bilo očigledno da će imati problema sa specijalistom za grafički dizajn. Premda po prvi put sarađuju, vođa projekta u početku nije razumeo zbog čega specijalista ima prigovarački, negativan stav prema projektu i okruženju. Umesto da ne nastavlja dalje saradnju sa takvim čovekom, vođa projekta je ipak pružio šansu specijalisti. Ubrzo je uvideo problem – specijalista za grafički dizajn je imao osećaj odbačenosti u novoj radnoj sredini i nedovoljno poštovanja njegove stručnosti. Tada je vođa projekta shvatio da verovatno u očima specijaliste deluje hladno, arogantno, rezervisano i da odaje utisak duboke sumnje u sposobnosti specijaliste. Stoga je vođa projekta odmah promenio svoj način rada s njim. Posvetio mu je više pažnje. Otvoreno mu je pokazao da ceni njegovu stručnost, jer je izuzetno bio dobar u tome što radi, na taj način što se u pauzama rada na projektu više zainteresovao za dosadašnje postignute rezultate specijaliste. Od tog trenutka specijalista je preuzeo dosta dodatnih zaduženja, postao je jedan od boljih radnika na projektu, i njegov doprinos timu i razvoju projekta je postao dragocen. Time je vođa projekta shvatio da je imao loš stav prema specijalisti, što je za posledicu imalo to da se specijalista postavio loše prema projektu i kolegama iz tima.

Međutim, zahvaljujući dobroj organizovanosti i pravovremenom reagovanju na situacije nastale tokom realizacije projekta, vođa projekta i njegov tim su projekat uspešno završili nekoliko dana pre kraja predviđenog roka.

ZAKLJUČAK

Svaki projekat je jedinstven i „priča za sebe“. Ovakvo tvrđenje predstavlja posledicu definicija projekta izloženih u ovom radu. Potvrda tome je i činjenica da je i svaki pojedinac, pa tako i onaj u projektnom timu, jedinstven, kao i svaka organizacija sa svojim pravilima, okruženjem i svojom kulturom.

Različiti pristupi metodologije upravljanja projektima nastali su upravo na tim različitostima, kada je postalo očigledno da jedinstven pristup problemu upravljanja projektima ne zadovoljava jedinstvene zahteve projekata. U takvom okruženju je nastalo više pristupa koji svaki na svoj način pokušavaju stvoriti okruženje koje se može primeniti na sve projekte unutar iste oblasti ili čak na projekte iz različitih oblasti poput IT industrije, građevinarstva, medicine, nauke, ... Međutim, jasno je da jedinstveni projekti zahtevaju i jedinstveni pristup metodologije upravljanja projektom, koji će biti svojstven samo za taj projekat.

Vodeći računa da će u budućnosti projekti biti još kompleksniji i sadržajniji, a samim tim i zahtevniji, i da će se od metodologije upravljanja projektima zahtevati primena na takvim projektima, vrlo je važan pravilan odabir pristupa metodologiji upravljanja projektima u okviru svake organizacije. Pri odabiru pristupa mora se voditi računa o primenjivosti na sve projekte unutar organizacionog sistema. Međutim, moguće je da ni jedan od postojećih pristupa neće zadovoljiti baš sve projekte. Umešno kombinovanje sistemskog i situacionog pristupa je nešto što se nameće kao opšte prihvatljiv kompromis pri razvoju metodologije upravljanja projektima unutar vlastitih specifičnih organizacionih sistema. Još jedno od mogućih rešenja je stvaranje generičke metodologije, koja bi se oslanjala na postojeće pristupe upravljanja, a koju bi bilo moguće oblikovati prema datom projektu, čak i prilagoditi u toku samog projekta. Vrlo važan korak u tom oblikovanju predstavlja učenje na iskustvima završenih projekata. Naravno, u toku nastanka generičke metodologije upravljanja trebalo bi sprovesti analizu primenjivosti metodologije na različite vrste projekata, kao i rešiti upravljanje resursima jer je moguće da bi nekoliko projekata, svaki vođen svojim pristupom metodologije, sa svojim pravilima, u isto vreme delili resurse i to različitim intenzitetom. Optimizacija deljenih resursa nameće se kao vrlo važan deo upravljanja projektima. Na kraju, ne treba zanemariti ni interna pravila organizacije koja takođe utiču na odabir pristupa metodologije upravljanja.

Područje upravljanja projektima doživelo je velike promene. Međutim, još se i dalje intenzivno razvija. U tom razvoju treba voditi računa i o potrebama budućih projekata i proces razvoja postaviti na najviši nivo zrelosti kako bi se osigurala što veća fleksibilnost. Činjenice i saznanja iz prakse pokazuju da je neophodno energičnije uvoditi i koristiti metodologiju upravljanja projektima, planski i na organizovan način, u skladu sa potrebama prakse, zahtevima vremena i savremenim trendovima.

Ostaje da se vide dalji razvoj i nova rešenja, ili bar potvrda vremena da je savremena metodologija upravljanja IT projektima dobra zamena starim konceptima. Za sada, sve je više pozitivnih argumenata, a kritike su obično motivisane otporom prema promenama.

XP je orijentisan ka ljudima pre nego ka procesima. Akcenat je na realnoj komunikaciji - komunikaciji licem u lice i “živoj reči”, pre nego na pisanoj dokumentaciji. Takođe, ovde se više ceni program koji radi. To predstavlja osnovnu meru napretka projekta. XP za rezultat ima kompletno razvijene i testirane funkcionalne celine koje su samo mali deo kompletne celine i to svakih par nedelja. Cilj je postići grub, ali upotrebljiv sistem na početku, pa ga tek onda poboljšavati. Karakteristično za timove XP-a je da rade istovremeno na različitim aktivnostima. XP stavlja akcenat na iterativno stvaranje softvera u kratkim vremenskim periodima. XP se razlikuje od iterativnih po tome što

se taj period kod njega meri nedeljama, a ne mesecima rada, a rad se odvija u duhu kolektivne saradnje.

Ekstremno programiranje je u mnogome fleksibilno, ali postoje i neki problemi za koje još uvek nisu nađena adekvatna rešenja. Visok nivo stručnosti u timu XP-a nije uvek moguć. To se relativno brzo popravlja uparivanjem iskusnih i onih manje iskusnih programera. Kent Beck napominje da se na taj način programiranje u paru pretvara u mentorski rad [1]. Tek posle nekoliko nedelja, pa i meseci jaz između programera postaje značajno manji. Ipak, organizacije nisu obrazovne isntitucije već mesta produktivnog rada i ostvarivanja profita. Neke od organizacija ne mogu da priušte svojim zaposlenima obuke od nekoliko meseci što može da predstavlja dobar motiv da se takve organizacije opredele za tradicionalni pristup razvoju softvera koji ne zahteva posebno visoku stručnost programera u timovima.

Mnogi programeri ne mogu da se uklope u sistem koji se nudi. Posebno teško može biti uvođenje programiranja u paru i rada dvojice za jednom radnom stanicom, jer ima ljudi koji to ne mogu da prihvate. Time se nameće krajnje radikalan zaključak da ko ne može da se prilagodi treba da bude isključen iz tima, što uopšte nije jednostavno.

Baze podataka predstavljaju veoma osetljivu temu u krugovima programera XP-a. Na njih se često trošilo i najviše vremena u periodu projektovanja i pripreme dokumentacije u tradicionalnim pristupima. Za sada se u domenu XP-a smatra da je to prevelika cena koja se plaća za jako malu dobit. Smatra se da je moguće projektovati i baze podataka kao i sam kôd, po principu dodavanja tabela tek kada zatrebaju.

Vođenje projekata vrlo je kompleksan zadatak. Treba uzeti u obzir sve faktore uticaja i poštovati sva ograničenja (od vremenskih, finansijskih do resursnih ograničenja). Uvek treba imati na umu konačni cilj i ne skretati s plana rada. Takođe, važno je da se ceo postupak realizacije projekta uredno i redovno dokumentuje kako bi "bilo ko" mogao nastaviti projekat kasnije na osnovu projektne dokumentacije. Važno je da vođa projekta donosi prave odluke u pravom trenutku i da uvek ima u vidu prioritete pomoću kojih bira konačni pristup rešenju. Isto toliko važna je redovna komunikacija između svih strana uključenih u projekat. I konačno, kada je projekat završen treba proveriti koliko je projektni tim bio uspešan u realizaciji konačnog rešenja projekta, u kojoj meri su ispoštovana sva data ograničenja (vremenski rokovi, budžet po fazama i u celini, ...) i u kojoj meri je odnos uloženog i dobijenog zadovoljavajući. Sva znanja i iskustvo sigurno će koristiti u vođenju nekog narednog projekta.

Bilo je mnogo uspešnih IT projekata, ali znatno više ih je propalo. Mudro bi bilo koristiti iskustva drugih u stvaranju i sprovođenju IT projekata. Nepotrebno je činiti iste greške i gubiti ogromne resurse da bi se došlo do saznanja da nešto ne valja. Ponekad je očigledno da je projekat bio potpuno uspešan ili totalno neuspešan. Ali u mnogim slučajevima, to nije toliko jednostavno oceniti. Na primer, veći projektni rezultati su dovršeni, ali projekat je probio budžet. Ili pak, projektni tim je isporučio na vreme i u okviru budžeta, ali rešenje se 80% poklapa sa poslovnim zahtevima. Ključ za definisanje uspeha jeste da se unapred definišu kriterijumi uspeha. Kriterijumi mogu biti sledeći:

- budžet projekta,
- trajanje projekta,
- iskorišćenost resursa,
- nivo kvaliteta,
- brzina povratka investicija,
- stepen inovativnosti,
- uticaj na tržište,
- zadovoljan klijent, ...

Na uspeh i kontrolu projekta bitno utiču:

- činjenica da vođa projekta ne upravlja, ne naređuje, već koordinira, usmerava i sugeriše rad na projektu,
- inicijalizacija projekta,
- plan projekta,

- vremenski plan projekta,
- praćenje i izveštavanje o statusu projekta,
- verifikacija procesa,
- identifikovanje problema,
- izbegavanje problema.

XP u potpunosti menja dosadašnju praksu timskog programiranja. Do sada je postojao sistem koji se sastojao od planiranja, u okviru kojeg se smišljala čitava struktura programa, analizirale potrebe korisnika, kreirali postupci pri izradi softvera, zatim se obavljalo kodiranje i na kraju - testiranje i ispravljanje grešaka. Autor ovog rada smatra da je najveći doprinos primene XP-a u tome što se prvo testira, a tek onda kodira. Testiranjem na ovaj način programeri stiču mnogo veću sigurnost u ispravnost svog kôda. Implementacija funkcionalnosti je time olakšana. Primenom integrisanih testova, pregledom kôda u realnom vremenu i malim ciklusima isporuke, XP osigurava dobijanje kvalitetnog kôda. Kvalitetan kôd se takođe obezbeđuje i kontinualnim testiranjem kôda u toku razvoja softvera. Ipak, postoji rizik, kao i u svim drugim pristupima razvoju softvera, jer veština i iskustvo članova tima određuje i nivo uspeha realizacije projekta. Ostati na ”pravom putu” postiže se čestim i kratkim sastancima koji kotrolišu i koriguju tok aktivnosti. Da bi projektni tim uspeo u primeni XP-a, njegovi članovi moraju biti odgovorni i motivisani, i da zaista žele da naprave sistem koji radi i na koji mogu biti ponosni. Ovo se odnosi kako na programere, tako i na ostale zainteresovane učesnike u projektu. Svako treba da obrati pažnju na aktivnosti koje direktno doprinose uspehu projekta i da izbegava aktivnosti koje se razlikuju od “pravog” posla.

XP timovi su uspešni zato što su njihovi članovi usresređeni na projekat. Praktične preporuke XP-a o kupcu na licu mesta, kolektivnom vlasništvu nad kôdom i programiranju u parovima promovišu aktivnu saradnju. XP timovi su uspešni i zato što su spremni da promene način svog rada. XP nalaže da se piše kôd za testiranje, pre nego što se napiše stvarni kôd programa. Na taj način se osigurava da je kôd uvek istestiran. XP u dizajnu koristi nove pristupe i modelira samo onoliko koliko je potrebno za početak, nakon čega se preporučuje da se stalno dorađuje kôd, da bi tokom rasta programa ostao kompaktan.

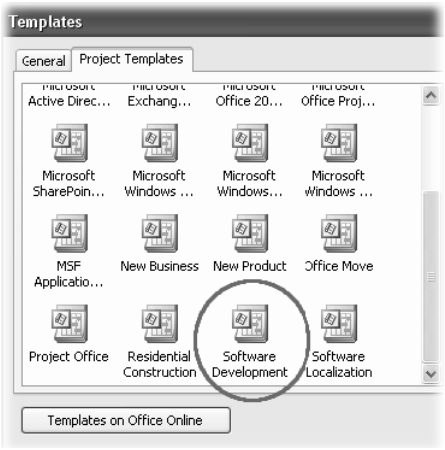
Lako je reći da je organizacija drugačija i da se metodologija upravljanja IT projektima i ekstremno programiranje jednostavno ne uklapaju u nju. Vrlo je teško, ali ne i nemoguće, uhvatiti se sa problemima u organizaciji i biti dovoljno hrabar za isprobavanje novih tehnika i alata. Vrlo je lako odustati, ali je teško izabrati uspeh. To najbolje znaju oni koji su uspešni u razvoju softvera. Metodologija upravljanja IT projektima i ekstremno programiranje su realnost, to svakako radi, i postoje da bi se koristili i bili od koristi.

DODATAK - Microsoft Office Project

MS Project je nastao po ideji *Microsoft*ovog direktora za razvoj, Alana Boyda, da se razvije alat koji će pomoći pri organizaciji velikog broja softverskih projekata koji su u fazi razvoja unutar *Microsoft*a [32]. Izrada bilo kog softvera predstavlja zaseban projekat, a većina velikih softverskih paketa su složeni projekti koji uključuju značajne finansijske resusrse, striktne vremenske rokove, angažovanje većeg broja timova programera, dizajnera, analitičara, ... Osnovno pitanje jeste da li je potrebna verzija za jednog korisnika koju će koristiti pojedinci ili serversko rešenje koje omogućava da više korisnika sarađuje koristeći zajedničke podatke.

Desktop aplikacija za upravljanje projektima, *MS Project Standard*, je zasnovana na operativnom sistemu *Microsoft Windows* i ujedno osnovni softverski paket iz porodice *Microsoft Office Project*-a. Ovo standardno izdanje prvenstveno je namenjeno jednom korisniku i samim tim ne omogućava rad u mreži. Osim osnovne, postoji i proširena verzija *MS Project Professional*, takođe desktop aplikacija zasnovana na *Windows* platformi, koja osim osnovnog paketa alata sadrži i dodatne funkcije za planiranje, zajednički rad i komunikaciju između članova projektnog tima. *MS Project Server* zasniva se na mrežnom rešenju putem Interneta i omogućava saradnju na projektima na nivou organizacije i koristi se zajedno sa *Professional* verzijom. U nastavku rada, pažnja će biti usmerena na dve osnovne verzije *Microsoft Office Project – Standard* i *Professional*.

Iako bi trebalo, mnogim projektima se ne upravlja pomoću pravog alata za planiranje. Često se u praksi koriste liste aktivnosti i resursa napravljene u programima za rad sa tabelama, poput *MS Excel*, ili gantogrami formatirani u programima za crtanje, kao što je to *MS Visio*. Prednost pravog alata za upravljanje projektima ogleda se u tome što on ima i softversku mašinu za planiranje. Na taj način je moguće: obraditi zahteve poput lančanih efekata kada jedan zadatak iz niza od više stotina zadataka promeni svoje trajanje, voditi računa o neradnim danima, specifičnim uslovima, ... Osnovne funkcije za rad u *MS Project*-u podrazumevaju izradu različitih projektnih planova, unošenje vrednosti vezanih za projektne zadatke i resurse, njihovo organizovanje, formatiranje različitih detalja plana projekta, praćenje stvarnih vrednosti u odnosu na planirane i preduzimanje koraka za korekciju plana u slučajevima kada se projekat ne realizuje na planiran način.

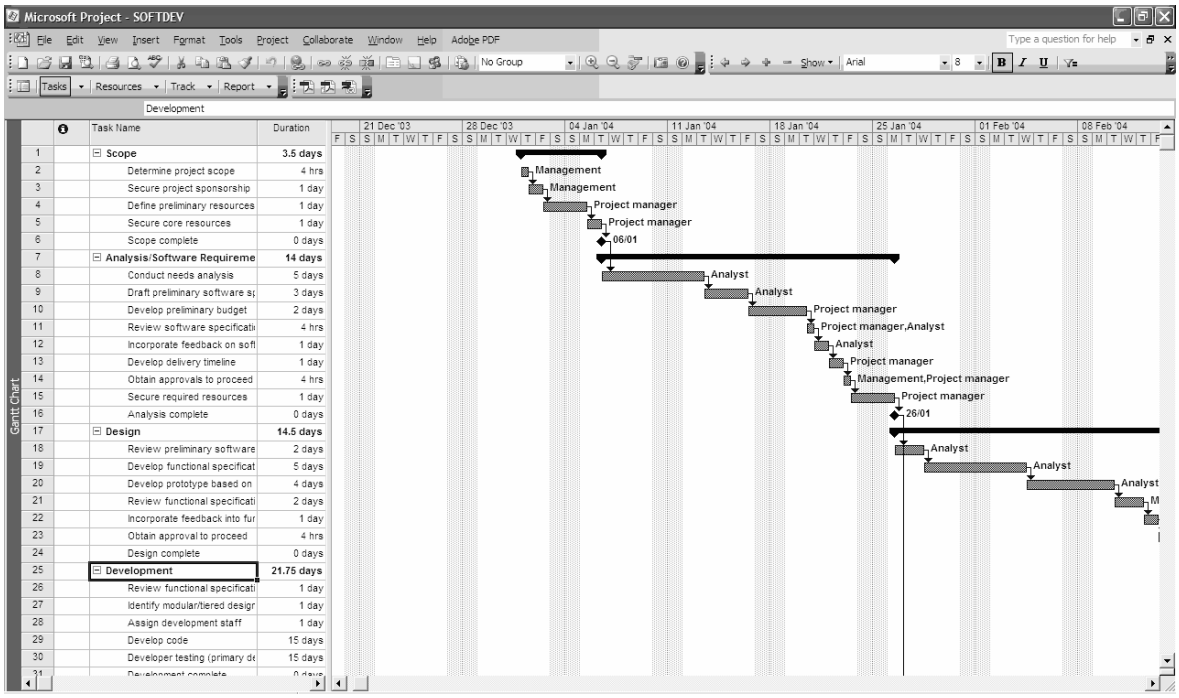


slika 23 - *Ugrađeni šabloni MS Project-a*

Često se umesto izrade poptpuno novog projektnog plana može koristiti određeni šablon (eng. template) koji sadrži većinu potrebnih informacija, kao što su nazivi aktivnosti, njihova trajanja i međuzavisnosti. Izvori ovih šablona mogu biti: šabloni instalirani sa programom, šabloni koji se već koriste u okviru organizacije ili šabloni koji su dostupni na Internetu.

Za kreiranje standardnog plana softverskog projekta može se izabrati *Software Development*, ugrađeni šablon *MS Projecta* (videti sliku 23), ili se može kreirati potpuno nov plan projekta.

Radna površina u *Microsoft Project*-u se naziva prikaz [32]. Na raspolaganju je više prikaza koji se koriste za unos informacija o projektu, za analiziranje, obrađivanje, prikazivanje i kreiranje izveštaja. Prikaz se fokusira na detaljima zadatka ili resursa. Pri pokretanju *Project*-a, uvek se prikazuje gantogram - *Gantt Chart* (videti sliku 24), koji prvo prikazuje listu zadataka, a potom i grafički svaki od njih. Na horizontalnoj osi je vreme, dok se na vertikalnoj osi nalaze zadaci koje je potrebno obaviti da bi se projekat završio. Samo trajanje zadatka predstavljeno je horizontalnim stupcima. Posebnim oznakama moguće je definisati, osim trajanja zadatka, procenat završenosti, međuzavisnost, značajne datume, ...



slika 24 – *Prikaz Gantt Chart*

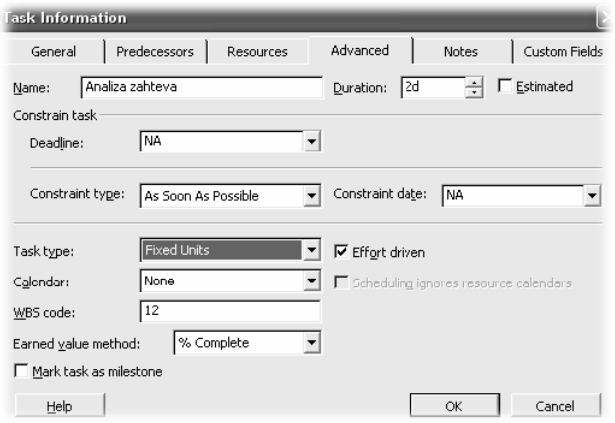
MS Project se primarno fokusira na vreme projekta [32]. Kod određenih projekata poznato je vreme početka, dok u drugim postoji samo konačni rok završetka projekta. U svakom slučaju u programu *MS Project* definiše se samo jedan datum, datum početka ili datum završetka projekta. Pri unosu početnog datuma projekta, uzimajući u obzir međuzavisnosti između zadataka, njihova trajanja i resursna ograničenja, *Project* računa najkraće moguće vreme realizacije. Slično tome, kada se definiše krajnji rok završetka projekta, izračunava se najkasniji mogući početak projekta. Iako je u praksi najčešće poznat krajnji rok završetka projekta, većina planova se izrađuje u odnosu na datum početka. To znači da svi zadaci treba da počnu što je ranije moguće, kako bi se obezbedila fleksibilnost u realizaciji [32].

Projektni zadaci su osnovni elementi koji sačinjavaju jedan projekat. Oni predstavljaju različite poslove koje treba obaviti kako bi se ostvarili ciljevi određenog projekta. Zadacima se opisuje rad na projektu pomoću redosleda, trajanja i potrebnih resursa. U *MS Project*-u upotrebljava se i prikazuje nekoliko različitih tipova projektnih zadataka (videti sliku 24) [32]:

- sumarni zadaci - faze,
- pojedinačni zadaci,
- ključni zadaci - događaji.

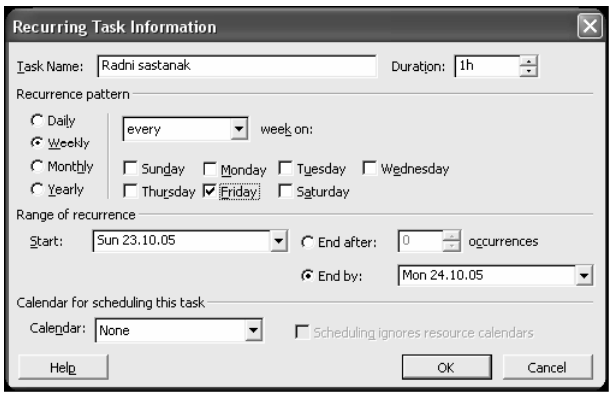
Trajanje zadatka predstavlja očekivano vreme potrebno za izvršavanje tog zadatka. *MS Project* može da radi sa vremenskim intervalima u rasponu od jednog minuta do nekoliko meseci [32]. U zavisnosti od obima samog projekta, u praksi su uglavnom trajanja zadataka izražena u časovima, danima ili nedeljama [34]. Automatsko unošenje uvek dodeljuje zadatku trajanje od jednog dana, dok ručno unošenje ne dodeljuje trajanje [32].

Često se javlja potreba za praćenjem značajnih događaja na projektu, poput završetka određene faze, kompletiranje rezultata projekta, ... S obzirom da ovakvi događaji ne podrazumevaju obavljanje nekog posla, oni u *MS Project*-u imaju trajanje nula [32]. Ključni zadatak se definiše kao događaj tako što se u *Advanced* u *Task Information* iz menija *Project* označi opcija *Mark task as milestone* (videti sliku 25) [34].



slika 25 – *Kartica Task Information/Advances*

Faze projekta su celine u koje su grupisani zadaci istog tipa. One se u programu *MS Project* predstavljaju u vidu sumarnih zadataka. Trajanje faze je zbir trajanja zadataka koji čine tu fazu. Kod automatskog unošenja trajanja, trajanje faze se ne može menjati bez promene trajanja zadataka koji je sačinjavaju. Kod ručnog unošenja, trajanje faze se može menjati, ali će *Project* pružiti uvid i u automatski izračunato trajanje. U okviru faza, na zadatke je moguće primeniti *Indent* i *Outdent*. Sa *Indent* zadatak postaje podzadatak zadatka koji se nalazi pre njega i na višem je nivou u hijerarhiji. Sa *Outdent* zadatak postaje nadzadatak sledećem zadatku u hijerarhiji, ako takav postoji, u suprotnom će postati zadatak najvišeg nivoa [32].



slika 26 – *Recurring Task Information*

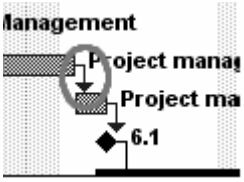
Raspoređivanje zadataka na vremenskoj osi postiže se, na jedan od sledećih načina, definisanjem:

- datuma početka i završetka zadatka,
- datuma početka i trajanja,
- trajanja,

a zatim se tipom ograničenja i povezivanjem vezama zavisnosti ostavlja *MS Project*-u da ga rasporedi.

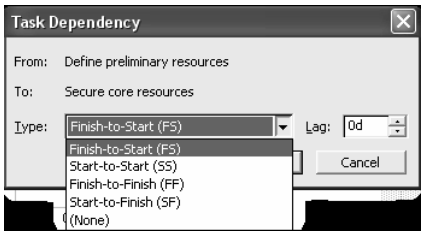
Ponavljajući zadaci se koriste u situacijama kada imamo aktivnost koju periodično obavljamo, npr. radni sastanci, kontrola kvaliteta, funkcionalni test, itd. Kreiraju se izborom opcije *Recurring Task Information* u meniju *Insert* (videti sliku 26), gde se unosi naziv, trajanje, obrazac ponavljanja (dnevno, nedeljno, mesečno, godišnje), dan ili dane u nedelji, mesecu, godini, kada počinje i kada se završava (posle određenog broja ponavljanja ili posle određenog datuma), radni kalendar.

Zadaci ne moraju imati unapred definisane datume početka i završetka već se to može automatski odrediti tako što se definišu veze zavisnosti između zadataka (videti sliku 27) [32].



slika 27 – *Veze zavisnosti između zadataka*

Tip veze u *Task Dependency* definiše zavisnost između početka odnosno kraja prethodnog zadatka i početka odnosno završetka narednog zadatka, dok *Lag* definiše odstupanje (videti sliku 28).

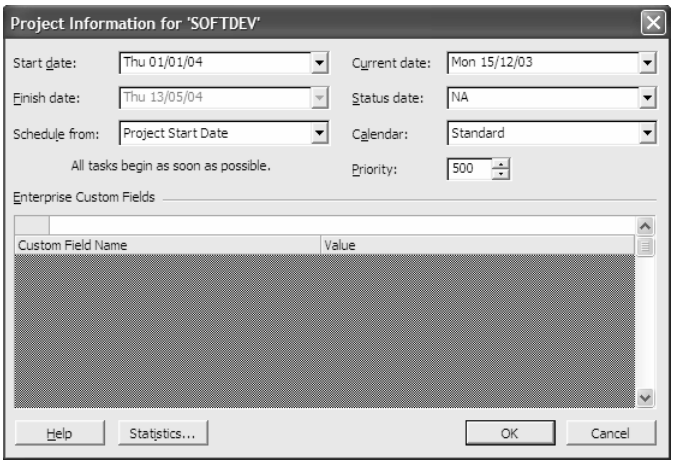


slika 28 – *Task Dependency*

Osnovni tipovi međuzavisnosti zadataka [34]:

- „završi da bi započeo" – FS (akronim, eng. *Finish-to-Start*)
- „započni da bi započeo" – SS (akronim, eng. *Start-to-Start*)
- „završi da bi završio" – FF (akronim, eng. *Finish-to-Finish*)
- „započni da bi završio" – SF (akronim, eng. *Start-to-Finish*)

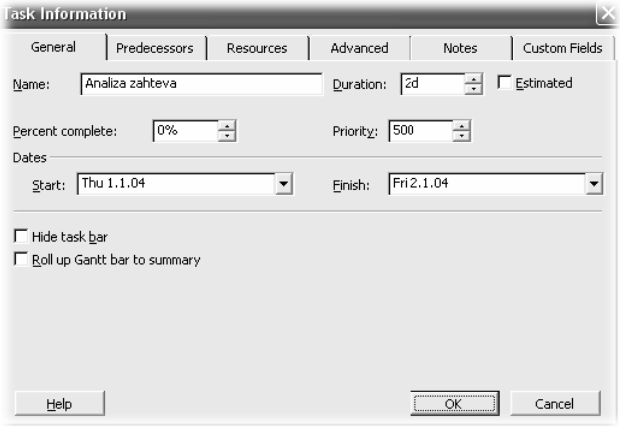
Nakon uspostavljanja relacija između zadataka, moguće je menjati datume početka i završetka, odnosno ubacivati dodatne aktivnosti. *MS Project* će uzeti u obzir svaku promenu i iznova izračunati sve neophodne parametre kao što su ukupno vreme i troškovi projekta.



slika 29 – *Prozor Project Information*

Prilikom izrade novog projektnog plana, najpre je potrebno definisati datum početka projekta, a zatim uneti projektne zadatke, njihova trajanja i međuzavisnosti. Nakon togra, zadatke treba organizovati po fazama projekta, podesiti radno vreme resursa i proveriti ukupno trajanje projekta. Kada se izrađuje novi plan projekta, program automatski podešava trenutni datum, kao datum početka projekta [32]. Planirani datum početka projekta može se promeniti biranjem *Project Information* u meniju *Project* (videti sliku 29).

Novi zadatak se definiše u T*ask Information* (videti sliku 30), gde se u kartici *General* unosi naziv zadatka, trajanje, procenat završenosti, prioritet, početni i krajnji datum, da li se stubac vidi na dijagramu i da li se vidi u zbirnom pregledu.



slika 30 – *Kartica Task Information/General*

U kartici A*dvanced* (videti sliku 25) može se definisati krajnji datum, tip ograničenja, tip zadatka, kalendar (za svaki zadatak moguće je vezati poseban radni kalendar), *Earned value method* - način određivanja količine završenog posla, definisanje zadatka kao važnog datuma (eng. *milestone*).

Iako vođe projekata gantogram (videti sliku 24) koriste kao standardni način vizuelizacije plana projekta, njime se često prezentuje plan projekta i osobama koje nisu direktno uključene u njegovu realizaciju. Pored toga, gantogrami su pogodni i za unošenje i fino podešavanje detalja o zadacima, kao i za analiziranje samog projekta i toka njegove realizacije [32].

Pored gantograma postoje i druge mogućnosti za praćenje aktivnog projekta. U meniju *View* izborom *Resource Sheet* umesto gantograma prikazuju se informacije o resursima projekta (videti sliku 31).

	Resource Name	Type	Material Label	Initials	Group	Max. Units	Std. Rate	Ovt. Rate	Cost/Use	Accrue At	Base Calendar	Code
1	Management	Work		M		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	
2	Project manager	Work		P		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	
3	Analyst	Work		A		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	
4	Developer	Work		D		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	
5	Testers	Work		T		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	
6	Trainers	Work		T		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	
7	Technical commun	Work		T		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	
8	Deployment team	Work		D		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard	

slika 31 – *Prikaz Resource Sheet*

MS Project podržava rad sa tri tipa resursa [32]:

- radni
- materijalni
- troškovni

Pod radnim resursima se podrazumevaju ljudi i oprema neophodni za izvršavanje projektnih zadataka. *MS Project* se fokusira na dva aspekta radnih resursa – njihovu raspoloživost i troškove. Raspoloživost resursa se odnosi na vreme kada određeni resusrsi mogu biti angažovani na projektnim zadacima i

količinu posla koju mogu da urade. Kod ljudi radni dan traje između 8 i 12 sati, dok kod opreme ovaj interval može trajati neprestano, odnosno onoliko koliko redovno održavanje opreme dozvoljava. Osim toga, oprema je usko specijalizovana za neke operacije, dok kod ljudi ipak postoji određena fleksibilnost. Troškovi predstavljaju količinu novca koju treba izdvojiti za angažovanje navedenih resursa. Radni resursi se u *Project*-u mogu definisati na nekoliko načina [32]:

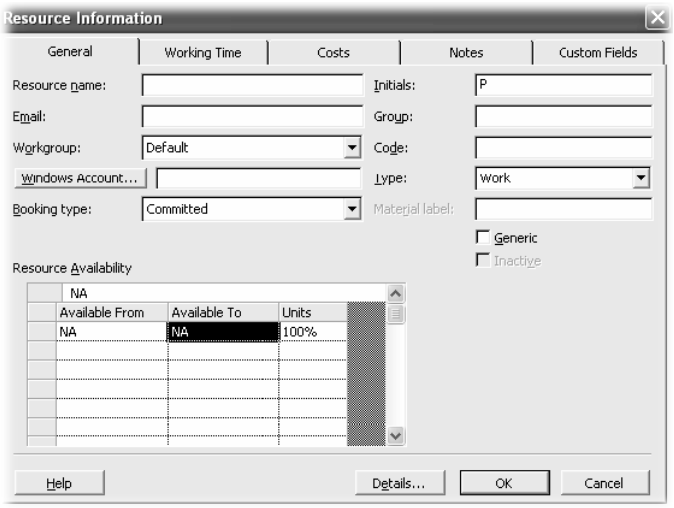
- pojedinačni ljudski resursi identifikovani prema imenu (na primer, Janko Janković)
- pojedinačni ljudski resursi identifikovani prema radnoj kvalifikaciji ili funkciji (na primer, programer, master informatičar, ...)
- grupa ljudi koja poseduje zajedničke veštine (na primer, dizajneri, ...)
- radna oprema (na primer, štampači, ...)

Materijalni resursi su potrošni materijali neophodan za realizaciju projekta. Pod troškovnim resursima se podrazumevaju finansije potrebne za realizaciju projektnih zadataka.

U programu *MS Project* ne moraju se dodeljivati resursi zadacima da bi se upravljalo vremenskim planom projekta. Međutim, postoji više dobrih razloga zbog kojih je poželjno da se vremenskom planu projekta pridruže resursi [34]:

- identifikovanje ljudi koji će izvršavati određene projektne aktivnosti,
- određivanje optimalnog broja ljudi koji će realizovati projekat,
- postizanje optimalne angažovanosti svakog pojedinačnog resursa, ...

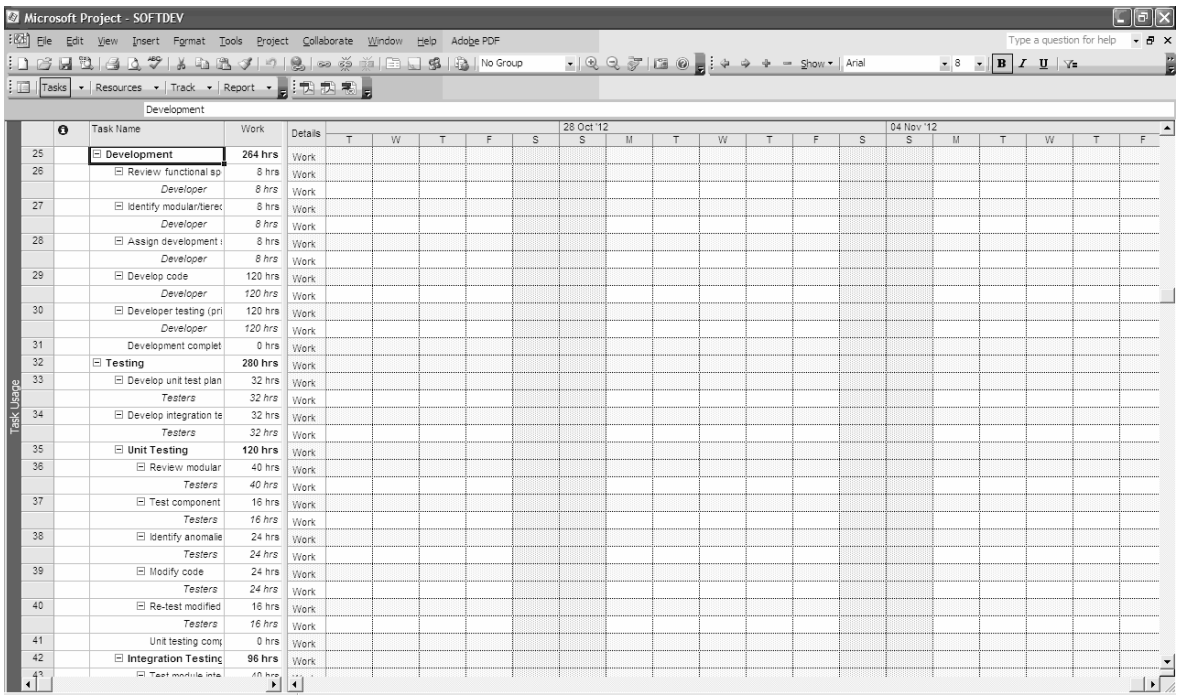
Prilikom definisanja resursa potrebno je preći na stranicu za unos resursa. Oni se definišu analogno zadacima, odnosno izborom *Resource Information* iz menija *Project* (videti sliku 32). Za definisanje materijalnih resursa potrebno je dodati kolonu za tip resursa - *Type* i kolonu za oznaku količine – *Materija Label*. Na dijalogu za podešavanje resursa može se podesiti naziv, e-mail (ukoliko je u pitanju osoba), radna grupa, tip (rad ili materijal), dostupnost (*Resource Availability*), radni kalendar, cena resursa, itd.



slika 32 – *Resource Information*

Podaci o resursima su prikazani u tabeli u kojoj svaki od resursa zauzima po jedan red (videti sliku 31). Ovaj prikaz se naziva prikaz liste. Pored resursa postoji još jedan izuzetno značajan prikaz liste, pod nazivom *Task Sheet*, u kome su dati detalji o zadacima (videti sliku 33).

Treba imati na umu da u podacima o resursima nije prikazan nijedan podatak o zadacima kojima ovi resursi mogu biti dodeljeni [32]. Prikazom *Resource Usage* vide se zadaci kojima je svaki od resursa dodeljen. Izborom *Task Usage* opcije iz menija *View* prikazuju se svi resursi koji su dodeljeni svakom od zadataka. Pored ovoga, mogu se prikazati i zadaci koji su dodeljeni resursima, na vremenskoj skali, uz različite vremenske periode prikaza (npr. dnevni, nedeljni...).

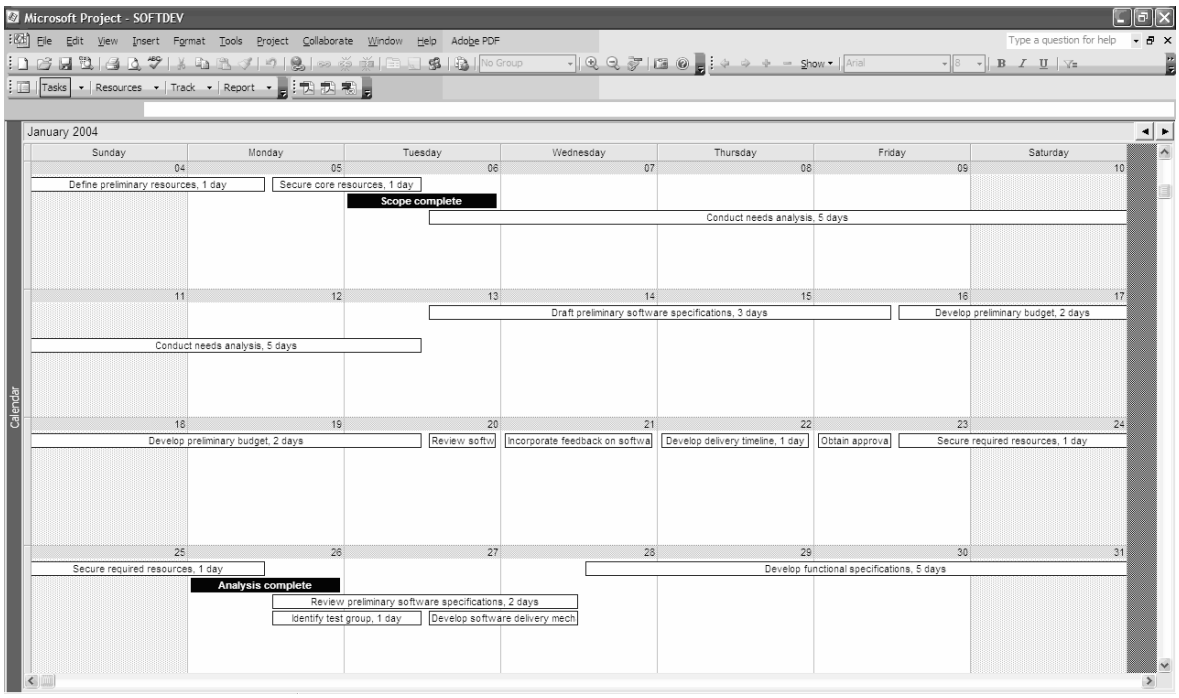


slika 33 – *Prikaz Task Sheet*

Svakom zadatku potrebno je dodeliti resurse koji se koriste za njegovo izvršavanje. Ukoliko se desi da paralelni zadaci koriste iste resurse rada (na primer, isti ljudi rade na njima) može se desiti da ukupna količina rada prelazi 100%. U tom slučaju je resurs preopterećen i potrebno ga je rasteretiti automatski ili ručno, pri čemu je ručna preraspodela resursa po zadacima ili pomeranjem zadataka po vremenskoj osi [32].

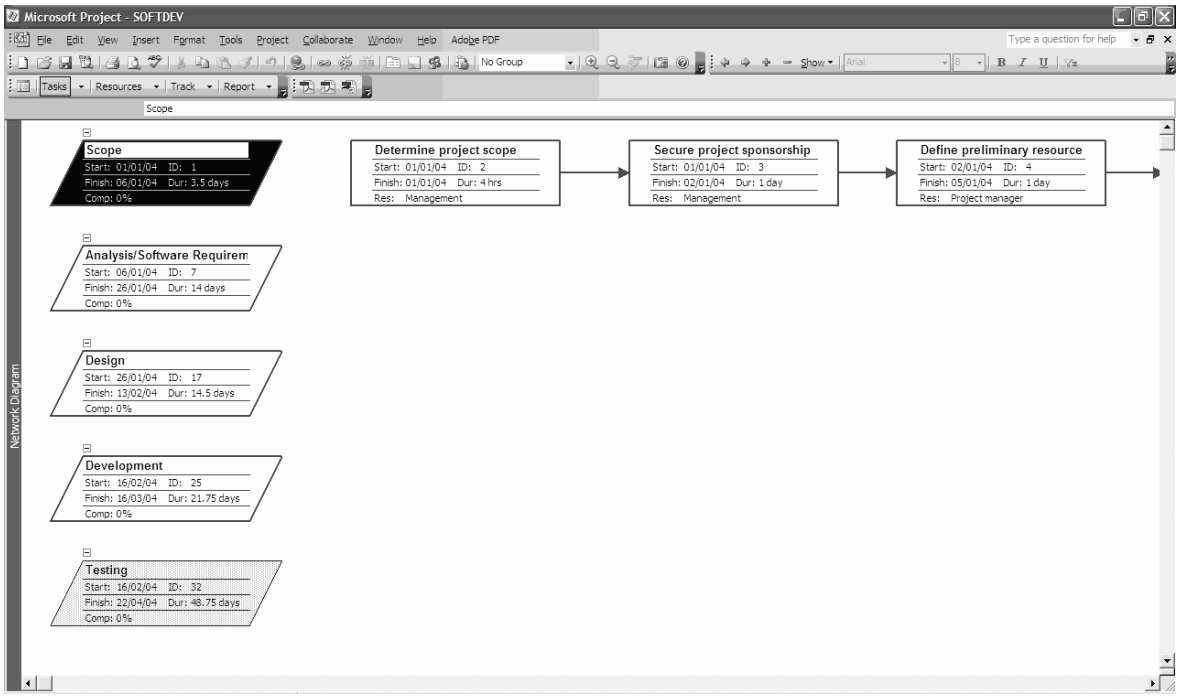
MS Project proračunava na osnovu unetog rada i količine resursa. Kada su resursi dodeljeni zadacima i ustanovljeno je koliko rada je potrebno uložiti, automatski se računaju troškovi. To se prenosi i na nivo sa zadacima i na kraju na nivo celog projekta. Svaki resurs može imati svoj kalendar što znači da se na taj način može definisati koji dan i u kojoj smeni je određeni resurs dostupan. Cene resursa se koriste da bi se izračunala potrošnja resursa i ukupna cena. Svaki resurs može imati više zadataka u više planova i svakom zadatku može biti dodeljeno više resursa. Prema tome, *Project* planira kada će se koji zadatak odvijati, s obzirom na trenutno stanje u kalendaru. Svi resursi mogu biti definisani bez ograničenja [32].

Prikaz *Calendar* iz menija *View* (videti sliku 34) predstavlja izuzetno korisnu opciju, jer se na jednostavan i brz način prikazuju planirane aktivnosti tokom određene nedelje ili meseca. Nazivi zadataka se, u okviru kalendara, pojavljuju na prikazima dana tokom kojih je planiran njihov početak, a ukoliko je planirano trajanje nekog zadatka duže od jednog dana, on će biti prikazan na više dana.



slika 34 – *Prikaz Calendar*

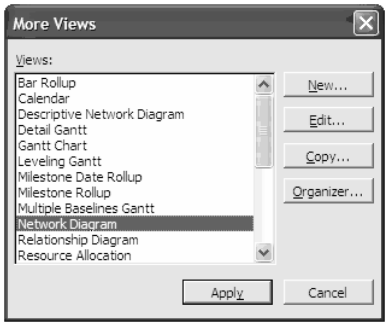
Mrežni dijagram (videti sliku 35) predstavlja još jedan od prikaza koji se često koriste u *MS Project*-u. Biranjem *Network Diagram* iz menija *View* prikazuju se međusobne relacije između pojedinih zadataka. Svako polje ili čvorište u ovom prikazu pokazuje detalje o zadatku, dok linije između njih predstavljaju odnose među zadacima.



slika 35 – *Prikaz Network Diagram*

Pored pregleda pojedinačnih prikaza koji su navedeni, postoji i mogućnost njihovog kombinovanog pregleda. Ovo se postiže deljenjem prozora plana projekta na dva dela, pri čemu je u svakom od njih različit prikaz. Ovi prikazi su sinhronizovani tako da izbor određenog zadatka ili resursa u jednom prozoru dovodi do toga da se u drugom prozoru prikažu detaljni podaci o izabranom zadatku ili resursu [32]. Odabirom opcije *More Views* u okviru menija *View* otvoriće se prozor *More Views*

(videti sliku 36). U njemu je spisak svih unapred definisanih prikaza koji su raspoloživi u *Microsoft Project*-u.



slika 36 – *Prozor More Views*

Definisanje radnog vremena ili radnog kalendara podrazumeva definisanje intervala rada na projektu. Ponuđeni kalendari su *Standard* (od ponedeljka do petka, od 8 do17h sa pauzom od 1h), *Night Shift* (noćni rad, od ponedeljka do subote, od 23 do 8h), *24 hours* (rad non-stop). Moguće je definisati nove kalendare ili modifikovati postojeće, kako bi se prilagodili određenim klasama resursa, na primer, poseban radni kalendar za analitičare, dizajnere, ... [32] Dijalogu za definisanje kalendara *Change working time* pristupa se iz menija *Tools*.

MS Project prepoznaje različite nivoe korisnika. Svaki od tih nivoa ima drugačiji pristup projektima i podacima, dok su kalendari, filteri i tabela dostupni svim korisnicima, bez obzira koja prava su im prethodno dodeljena [34].

Sve do početka realizacije projekta, projektni tim je fokusiran na planiranje. Sa početkom realizacije projekta, počinje i nova faza upravljanja projektom – praćenje projekta. Pod praćenjem se podrazumeva dokumentovanje detaljnih podataka o projektu, koji se odnose na urađeni posao, utrošeno vreme i ostvarene troškove. Detalji koji se prate nazivaju se stvarni podaci, a njihovo praćenje se sprovodi u cilju upoređivanja sa planiranim, i smanjivanja eventualnih odstupanja na minimum [34].

MS Project podržava nekoliko načina za praćenje napretka na projektu, koji se razlikuju u količini specifičnih detalja i nivou kontrole koji je potreban vođi projekta i projektnom timu. Ti načini praćenja napretka su [32]:

- praćenje prema rasporedu,
- praćenje prema procentu izvršenja,
- praćenje stvarnih vrednosti,
- praćenje pojedinačnih aktivnosti.

Kod većine zahtevnijih projekata moguće je upotrebiti kombinaciju navedenih načina, kako bi se sagledalo što više detalja [32].

Literatura

[1] Beck K.: ***Extreme Programming – Explained***, Addison Wesley, 2000.

[2] Fowler M., Kent B., Brant J., Opdyke W., Don R.: ***Refaktorisanje - poboljšanje dizajna postojećeg kôda***, Addison Wesley, CET – srpsko izdanje 2003.

[3] Baird S.: ***Extreme programming***, Sams Publishing, 2003.

[4] IPMA: ***IPMA Competence Baseline***, Monmouth, UK: International Project Management Association, 1999.

[5] http://www.atlassian.com (konsultovano februara 2012.)

[6] Thomsett R.: ***Extreme Project Management***, Executive Report, Cutter Consortium, Vol.2, No. 2, 2001.

[7] PMI: ***A Guide to the Project Management Body of Knowledge, Third Edition (PMBOK Guide)***, Newtown Square, PE: Project Management Institute, 2004.

[8] Wysocki K. R., McGary R.: ***Effective Project Management***, Third Edition, Indianapolis, IN: John Wiley & Sons, Inc, 2003.

[9] Kleim L. R., Ludin S. I.: ***Project Management Practitioner's Handbook***, AMACOM Books, 1998.

[10] Kerzner H.: ***Project Management: A Systems Approach to Planning, Scheduling, and Controlling***, Eighth Edition, Hoboken, NJ: John Wiley & Sons, Inc, 2003.

[11] Heerkens R. G.: ***Project Management***, New York, NY: McGraw-Hill, 2002.

[12] Hughes B., Cotterell M.: ***Software Project Management***, Second Edition, London: McGraw-Hill, 1999.

[13] Charvat J.: ***Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects***, Hoboken, NJ: John Wiley & Sons, Inc, 2003.

[14] Cockburn A.: ***People and Methodologies in Software Development***, University of Oslo, 2003.

[15] Cockburn A., Williams L.: ***The Costs and Benefits of Pair Programming***, eXtreme Programming and Flexible Processes in Software Engineering XP2000.

[16] Highsmith J.: ***Agile Project Management***, Boston, MA: Addison-Wesley, 2004.

[17] ***Manifesto for Agile Software Development***, dostupno na http://www.agilemanifesto.org (konsultovano maja 2012.)

[18] Coram M., Bohner S.: **The Impact of Agile Methods on Software Project Management**, *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05*), IEEE, 2005.

[19] Turnquist A. M., Nozick K. L.: **Allocating Time and Resources in Project Management Under Uncertainty**, *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03*), IEEE, 2002.

[20] **IEEE Standard Glossary of Software Engineering Terminology**, IEEE Std 610.12-1990 (R2002), 1990.

[21] **Guide to the Software Engineering Body of Knowledge (SWEBOK)**, IEEE Computer Society, 2004.

[22] http://www.extremeprogramming.org (konsultovano marta 2012.)

[23] Laird L., Brennan C.: **Software Measurement and Estimation: A Practical Approach**, John Wiley & Sons, Inc., Hoboken, New Jersey, Jun. 2006.

[24] Boehm B.: **Software Engineering Economics**, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[25] Roetzheim W.: **Estimating and Managing Project Scope for New Development**, Crosstalk - The Journal of Defense Software Engineering, April 2005.

[26] http://www.comptia.org (konsultovano marta 2012.)

[27] Sandhu S. P., Bassi P., Brar S. A.: **Software Effort Estimation Using Soft Computing Techniques**, World Academy of Science, Engineering and Technology 46, 2008.

[28] Williams L.: ***Debunking the Nerd Stereotype with Pair Programming*** (Broadening Participation in Computing Series), IEEE Computer, Vol. 31, No. 5, pp. 83-85, May 2006.

[29] Williams L.: ***Pair Programming***, Making Software: What Really Works, and Why We Believe It, O'Reilly Media, Inc., 2011.

[30] Williams L.: ***The XP Programmer: The Few Minutes Programmer***, IEEE Software, Vol. 20, No. 3, pp. 16-20, May/June 2003.

[31] Williams L., Kessler R. R., Cunningham W., Jeffrise R.: ***Strengthening the Case for Pair-Programming***, IEEE Software, Vol. 17, No. 4, pp. 19-25, July/Aug 2000.

[32] Chatfield C., Johnson T.: ***Microsoft Office Project 2007 Korak po Korak***, CET, Beograd, 2007.

[33] Maljković M., Perić N.: ***XP (eXtreme Programming),*** Matematički fakultet, Univerzitet u Beogradu, skripta, 2008.

[34] Perić N.: ***MS Project [alat za upravljanje projektima],*** Matematički fakultet, Univerzitet u Beogradu, skripta, 2008.

[35] Baird S.: ***Extreme Programming Practices in Action***, Pearson education, Informit, dostupno na http://www.informit.com (konsultovano jula 2012.)

[36] Baird S.: ***Leading Your Extreme Programming Project***, Pearson education, Informit, dostupno na http://www.informit.com (konsultovano jula 2012.)

[37] Cole A.: ***Runaway Projects – Causes and Effects***, Software World, UK, Vol 26. No. 3, 1995.

[38] http://www.oracle.com (konsultovano februara 2012.)