UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA
PODIPLOMSKI ŠTUDIJ
MAGISTRSKI PROGRAM INFORMACIJSKO
UPRAVLJALSKE VEDE

Andrej Kampuš

PROJEKTNI MANAGEMENT PRI RAZVOJU PROGRAMSKIH REŠITEV

MAGISTRSKO DELO

Mentor: prof. dr. Andrej Kovačič
Somentor: prof. dr. Rudi Rozman

Ljubljana, september 2002

Kazalo

1 Uvod

1.1 Opis obravnavane vsebine

Projektni pristop je vedno bolj pogost pristop k vodenju in izvedbi del v organizacijah. Definicije pojma projekt se sicer nekoliko razlikujejo od avtorja do avtorja, lahko pa rečemo, da je projekt vsaka aktivnost, ki ima svoj začetek in konec in katere namen je ustvariti edinstven izdelek ali storitev. Projektni management obsega uporabo znanja, veščin, orodij in tehnik pri vodenju projektnih aktivnosti, z namenom da zadovoljimo naročnikove želje. Za uspešno vodenje projekta potrebujemo poleg pravil projektnega managmenta tudi znanja s področja splošnega managementa.

Projekti s področja informacijske tehnologije in razvoja programskih rešitev postavljajo projektnemu managementu še posebne izzive. Razvoj programske opreme poteka v dinamičnem, hitro spreminjajočem se okolju, brez dobro definiranih industrijskih standardov in brez zanesljivih kvalitativnih podatkov iz sorodnih projektov. Tehnologija se praktično spreminja iz dneva v dan, tako da panoga zahteva nenehno sledenje spremembam in izobraževanje zaposlenih.

McNeice Filler (2001) v članku podaja analizo svetovalne skupine Gartner, »da do leta 2002, 75 procentov projektov v informacijski tehnologiji ne bo doseglo zastavljenih ciljev, zaradi napak pri planiranju projektov«. V istem članku najdemo tudi mnenje svetovalnega podjetja KPMG, da tehnologija ni poglavitni razlog za ogromen odstotek neuspelih projektov. Raziskava v 256 podjetjih je ugotovila, da samo 14 procentov neuspešnih projektov lahko pripišemo nezmožnosti obvladovanja tehnologije. Ostalih 86 procentov projektov ne uspe zaradi tipičnih managerskih hib: neustrezno definiranih ciljev (17 procentov), neznanega obsega (17 procentov), pomanjkanja učinkovite komunikacije (20 procentov) in pomanjkanja managerskih znanj (32 procentov).V magistrskem delu želimo ugotoviti, kateri so najbolj pogosti vzroki za neuspeh projektov s področja informacijske tehnologije in kaj je potrebno pri vodenju projektov upoštevati, da se tem vzrokom izognemo.

Vedno bolj pomemben faktor pri vodenju projektov postaja uporaba ustrezne programske rešitve za vodenje projektov. Takšna programska rešitev podjetju omogoča optimizirati vire, izboljša komunikacijo med člani projektnega tima, poveča preglednost projektnih podatkov ter omogoča kontrolo nad tveganji in problemi.

Na trgu je prisotnih mnogo različnih ponudnikov programskih rešitev za vodenje projektov. Za podjetje je seveda ključno vprašanje, katero od ponujenih rešitev izbrati. Izbira je močno odvisno od kompleksnosti in obsega projektov, števila članov projektnih skupin in panoge, s katero se podjetje ukvarja.

1.2 Namen in cilji dela

Namen magistrskega dela je opredeliti projekti management pri razvoju programskih rešitev. Večina projektov razvoja programskih rešitev žal ne uspe zagotoviti glavnega cilja

projekta - razviti rešitev, ki je praktično brez hroščev, razvita v planiranih časovnih okvirjih, ustreza definiranim zahtevam in jo lahko vzdržujemo. Delo želi najprej ugotoviti glavne vzroke za neuspešnost projektov v praksi. Za ugotovljene vzroke je potrebno poiskati tudi ustrezna orodja in tehnike za njihovo odpravljanje.
Namen je tudi poudariti pomen učinkovitega programskega orodja za vodenje projektov. Orodij za podporo projektnemu vodenju je na trgu ogromno, zato se za podjetje postavlja vprašanje, katero od številnih orodij izbrati. Namen dela je razviti ocenjevalni model, ki bo podjetjem v pomoč pri izbiri različnih programskih orodij. Model mora biti dovolj dober, da bo opisal glavne lastnosti orodij in da bo izpostavil njihove prednosti in slabosti. Na osnovi modela, ocen in lastnih potreb se podjetje odloči katero od preizkušenih rešitev bo uvedlo v podjetje.
Cilji magistrskega dela so:

- preučiti osnovne pojme s področja projektnega managementa,
- predstaviti projektni način dela pri razvoju programskih rešitev,
- ugotoviti, v čem je programskih rešitev specifičen s stališča projektnega managementa,
- določiti najbolj pogoste razloge za neuspehe projektov s področja razvoja programske opreme,
- ugotoviti, kako se lahko ugotovljenim problemom učinkovito izognemo,
- opredeliti zakaj ustrezno orodje za vodenje projektov pomaga pri uspešnem projektnem managementu,
- določiti kriterije, ki opisujejo uspešnost programskega orodja za vodenje projektov,
- opredeliti način ocenjevanja posameznih kriterijev in določanja skupne ocene za programski orodje namenjeno vodenju projektov,
- preizkusiti tri programske rešitve: MS Project, e-Proj in Primavera TeamPlay v testnem večuporabniškem multiprojektnem okolju,
- oceniti preizkušene rešitve po predhodno definiranem ocenjevalnem modelu,
- analizirati dobljene rezultate ter predeliti prednosti in slabosti posameznih orodij,
- narediti primerjavo med preizkušenimi rešitvami,
- oceniti kako okolje vpliva na normiranje kriterijev za oceno programskih orodij,
- ugotoviti za katero okolje je posamezna rešitev najbolj primerna.

1.3 Uporabljene metode

Magistrsko delo se bo v uvodnih poglavjih opiralo na teoretične osnove s področja projektnega managementa. Uporabljena in primerjana bodo znanja različnih avtorjev (Lipovec, Rozman, Kovačič, Drucker). Delo pa se bo opiralo tudi na metodologijo projektnega instituta PMI (Project Management Institute).
Pri ugotavljanju specifičnosti projektnega managementa na področju razvoja programskih rešitev, bodo uporabljena spoznanja nekaterih najpomembnejših avtorjev s tega področja (Royce, Boehm, Gilb), viri na spletnih straneh in mnenja iz prakse.

Analiza razvoja programskih rešitev, bo osnova za opredelitev kriterijev uspešnega programska orodja za vodenje projektov. Vsak od glavnih kriterijev bo razčlenjen na več podkriterijev in ocenjen z oceno med 0 in 9. Glede na pomembnost bodo posamezni podkriteriji in kriteriji ustrezno oteženi. Uteži bodo normirane, tako da bo tudi končna ocena posameznega orodja v isti zalogi vrednosti.
V nalogi bo model za ocenjevanje programskih orodij za vodenje projektov preizkušen na treh orodjih: MS Project 2000, e-Proj in Primavera TeamPlay. Vse rešitve bodo preizkušene v testnem okolju. Testno okolje bo predstavljalo fiktivno, srednje veliko računalniško podjetje (za slovenske razmere), katerega primarna dejavnost je razvoj programskih rešitev za zunanjega naročnika. V testnem okolju bodo simulirani trije projekti s različno vsebino (razvojni, uvajalski in vzdrževalni projekt). Vsi projekti bodo časovno postavljeni v isto obdobje (prvo polletje leta 2002), tako da bo možno v testnem okolju simulirati povezavo med projekti. Vse orodja bodo preizkušena na IBM kompatibilnem osebnem računalniku z operacijskim sistemom Windows 2000, 700 MHz Intelovim procesorjem in 128 MB RAMa. Licence za preizkus rešitev MS Project 2000 in e-Proj (verzija 5.1.1.) je zagotovilo podjetje Genis d.o.o. Testno verzijo produkta Primavera TeamPlay (verzija 2.1.05) je zagotovilo hrvaško podjetje DelaComp d.o.o., ki je zastopnik za rešitve Primavera za področje Hrvaške in Slovenije.

1.4 Kratek opis poglavij

Poglavje 1: Uvod
Poglavje na kratko predstavi obravnavni problem magistrskega dela. Predstavljeni so namen in cilji dela, ter uporabljen metode.

Poglavje 2: Projektni management
Namen poglavja je predstaviti osnovna znanja s področja projektnega managementa. Poglavje definira pojem projekt, predstavi različne tipe projektne organizacije in definira vloge in odnose posameznikov znotraj projekta. Opredeli proces projekta od inicializacije pa do zaključka.

Poglavje 3: Projektni management pri razvoju programskih rešitev
Poglavje natančneje opredeli projektni management pri razvoju programski rešitev. Poudarjeni so najbolj pogosti vzroki za neuspešnost informacijskih projektov in tehnike za njihovo obvladovanje.

Poglavje 4: Programske rešitve za podporo projektnemu vodenju
V poglavju je razvit ocenjevalni model za orodja za podporo projektnemu managementu. Opisane so preizkušene rešitve, testno okolje in ocene posameznih orodij. Izpostavljene so prednosti in slabosti posameznih orodij in njihova medsebojna primerjava. Primerjava orodij je izvedena v treh primerih: enakomerno obteženih kriterijih ter kriterijih za majhno in veliko podjetje.

Poglavje 5: Zaključek

Poglavje predstavlja povzetek rezultatov ugotovljenih v magistrskem delu ter podaja primerjavo tez, iz katerih je delo izhajalo, in dejansko ugotovljenih rezultatov. Poglavje poudarja glavne rezultate dela in postavlja smernice za nadaljnje delo.

2 Projektni management

2.1 Osnovni pojmi

Enostavno opredeliti tako širok pojem kot je management vsekakor ni lahka naloga. Definicije managementa se od avtorja do avtorja seveda nekoliko razlikujejo, ampak vsaka izmed njih izhaja iz tehnične delitve dela. Tehnična delitev dela pomeni, da določene naloge ne opravi ena sama oseba, ampak jo razčlenimo na več manjših nalog, ki jih opravijo različne osebe ob različnih časih. Delitev dela ustvarja potrebo po usklajevanju. Rozman (1993, str.19) definira usklajevaje »kot proces, v katerem manager z vnaprej zamišljenim načrtom zavestno usmerja dejavnosti posameznika v skladu z zahtevami celotne naloge podjetja« in opredeli usklajevanje kot bistvo dela managerjev. Za lažje razumevanje pojma management, si oglejmo še definicije nekaterih drugih avtorjev.

Management je po Lipovcu (1987, str. 136-137) organizacijska funkcija ali proces, ki omogoča, da ločene operacije posameznih izvajalcev ostanejo člen enotnega procesa uresničevanja cilja gospodarjenje.

Drucker (1954, str. 341-342) primerja delo managerja z delom dirigenta, ko pravi: »Managerjeva naloga je ustvariti celoto, ki bo več kot vsota delov; celoto, ki bo dajala več kot vsota naporov, vloženih vanjo. Manager je podoben dirigentu simfoničnega orkestra. Z njegovim delom, vizijo in vodenjem zazvenijo posamezni inštrumenti, ki vsak zase povzročajo toliko hrupa, kot celota glasbe.«

Iz različnih definicij managementa lahko zaključimo, da je bistveno delo managerjev usklajevanje. Management izvaja usklajevanje z različnimi vlogami ali funkcijami. Po Rozmanu (1993) so funkcije managementa planiranje, kontrola, organiziranje in vodenje. Nekateri drugi avtorji med funkcije vključujejo še odločanje in delegiranje, Rozman pa zagovarja stališče, da sta odločanje in delegiranje prisotna pri vseh funkcijah managementa.

Delovna opravila in naloge v podjetju se izvajajo v procesu planiranja, izvedbe in kontrole. Planiranje je zamišljanje ciljev, rezultatov in poti za njihovo doseganje. V podjetju poteka planiranje na različnih organizacijskih nivojih, usklajuje cilje in poti posameznikov, kakor tudi celotnega podjetja. Planiranje predstavlja prvo funkcijo managementa. Planiranju sledi izvedba, pri kateri management neposredno ne sodeluje. Vzporedno s procesom izvedbe teče v podjetju opazovanje in spremljanje izvedbe, ugotavljanje odstopanj od plana, ugotavljanje vzrokov za odstopanje in ukrepanje, ki naj odpravi vzroke. Temu procesu pravimo kontrola in je druga funkcija managementa.

Planiranje, izvedba in kontrola delovnih opravil so neposredno vezani na delovna mesta in zaposlene v podjetju. Posamezna delovna opravila se združujejo v delovne naloge, te pa so vezane na večje enote, oddelke in posledično na celotno podjetje. Procesu določanja razmerij med zaposlenimi in vzpostavljanju strukture v podjetju pravimo organiziranje. Organiziranje je tretja funkcija managementa.

Neposredna posledica planiranja in organizacije je, da delavnih nalog ne opravlja tista oseba, ki si jih je tudi zamislila. Delovne naloge je torej treba prenesti na izvajalce, kar poteka v procesu komunikacije. Za uspešno izvedbo delovne naloge je poleg usposobljenosti izvajalca, potrebna še izvajalčeva želja, motiviranost za izvedbo naloge. Naloga managerja je, da v izvajalcih vzpodbudi motiviranost za opravljanje dodeljenih delovnih nalog. Skupno pravimo procesom komuniciranja, motiviranja in proženja akcij vodenje. Vodenje je četrta funkcija managementa.

Našteti pojmi so značilni za vsa področja managementa, v delu pa se bomo posvetili projektnemu managementu. Do definicije projekta pridemo preko poslovnih procesov. Poznavanje in obvladovanje poslovnih procesov v podjetju je ključno za poznavanje delovanja le tega. V vsakem podjetju obstajata pravzaprav dva glavna tipa poslovnih procesov:

- ponavljajoči se oziroma kontinuirani in
- ne ponavljajoči se oziroma enkratni poslovni procesi.

Ponavljajoči poslovni procesi so značilni za dele podjetja, kjer se srečujemo s serijsko, masovno proizvodnjo. Ti procesi ostajajo, vsebinsko in kakovostno gledano razmeroma nespremenjeni. Takšne procese lahko identificiramo tudi v okviru področij poslovanja, ki zagotavljajo infrastrukturo za potrebe opravljanja glavne dejavnosti podjetja.

Enkratni poslovni procesi so navzoči v vseh podjetjih, bodisi da gre za izvajanje glavne dejavnosti ali pa za realizacijo procesov razvoja in prilagajanja podjetja. Tem enkratnim poslovnim procesom pravimo projekti. Gre za neponavljajoče se procese, preko katerih realiziramo enkratne cilje.

Po definiciji projektnega instituta PMI (1996, str. 4) je projekt začasen proces, katerega namen je ustvariti edinstven produkt ali storitev. Začasen pomeni, da ima vsak projekt definiran začetek in konec. Edinstven pomeni, da sta dobljena produkt ali storitev do razpoznavnosti različna od vseh podobnih produktov ali storitev. Navedimo še definicije nekaterih drugih avtorjev.

Rozman (1993, str. 158) definira projekt kot enkratno dejavnost, sestavljeno iz vrste med seboj prepletajočih se aktivnosti. Dve enostavni in zato dobro uporabni definiciji ponuja tudi Rant et al. (1995, str.8): »Projekt je več časovno in strukturno med seboj povezanih opravil – dejavnosti.«. In še ena: »Projekt je način izvajanja enkratnih procesov.«

Navedimo še nekaj primerov projektov v podjetju:

- razvoj novega izdelka ali storitve,
- uvedba spremembe v načinu kadrovanja podjetja,
- načrtovanje novega transportnega vozila,
- razvoj ali uvedba novosti v obstoječem informacijskem sistemu,
- vodenje predvolilne politične kampanije.

Združitev obeh predstavljenih pojmov, managementa in projekta, nas pripelje do definicije projektnega managementa. Projektni management obsega uporabo znanja, veščin, orodij in tehnik pri vodenju projektnih aktivnosti, z namenom zadovoljiti naročnikove potrebe. Institut PMI (1996, str. 6) ugotavlja, da mora uspešen projektni management poiskati

ravnotežje med obsegom projekta, časovnimi roki, stroški in kvaliteto. Roki, stroški in kvaliteta tvorijo tako imenovan projektni trikotnik, z vsako od količin v ogliščih trikotnika. Sprememba ene od količin, nujno pogojuje tudi spremembe ostalih dveh. Po PMI-ju je naslednja ključna vloga projektnega managementa ugotovitev naročnikovih definiranih zahtev (potreb) in nedefiniranih zahtev (pričakovanj). Pregled in obvladovanje naročnikovih potreb in pričakovanj je ključno za uspešen projekt.

Projektni management vključuje tako znanja s področja klasičnega managementa, kakor tudi znanja specifična za projektni management (npr. analiza kritične poti in tehnike mrežnega planiranja). Po Rozmanu (1993, str. 158-161) štiri funkcije managementa uvrstimo v projektni management na sledeči način. Planiranje pomeni določitev vseh aktivnosti na projektu, njihovo povezanost, trajanje, stroške in vire. Planiranje temelji v glavnem na tehniki mrežnega planiranja. Tudi druga funkcija managementa, kontrola, se nanaša na projektne aktivnosti. Ugotavlja odstopanja izvedbe od plana in izvaja ukrepe, ki težijo k izvedbi plana. Pod organizacijo projekta razumemo določanje skupin in posameznikov odgovornih za izvedbo posameznih aktivnosti. Prav tako organizacija projekta opredeljuje odnose med sodelujočimi v projektu. V kontekstu projektnega managementa dobi novo podobo tudi četrta funkcija managementa, vodenje. Pri projektu vodenje postane značilno za projektne skupine, time. Ti združujejo strokovnjake z različnih področij, prav tako pa projektne skupine ne ustrezajo klasični hierarhični ali poslovno funkcijski organizacijski obliki podjetja. Pri projektnem managementu so torej zastopane vse funkcije splošnega managementa, le da dobijo v luči projektov nekoliko spremenjeno obliko. Seveda bodo posebnosti projektnega managementa podrobneje obdelane v nadaljevanju dela.

Po PMI-ju (1996, str. 9) se pri projektnem managementu prekrivajo znanja s treh področij: splošnega managementa, projektnega managementa in aplikativna znanja. Aplikativna znanja so znanja, značilna za področje, kjer projekt poteka. Primer aplikativnega področja so na primer tehnična področja kot razvoj programske opreme, farmacija ali gradbeništvo. Grafičn lahko sodelovanje znanj vseh treh področij prikažemo s sliko 2-1 (slika je zgolj shematična, prekrivanja posameznih področij niso proporcionalna):

Slika 2-1: Povezava različnih znanj, ki nastopajo v projektu.

2.2 Projektna organizacija

2.2.1 Projekti in oblike organizacijskih struktur

»Projektna struktura je organizacijska struktura enega samega projekta. Vzpostavi se za njegovo trajanje in se, ko je projekt končan, opusti.« (Rozman, 1993, str. 158). Projektna organizacija je torej prehodna organizacijska oblika, zato se postavlja vprašanje kako jo uvrstiti v organizacijsko strukturo podjetja. Izbira ustrezne organizacijske oblike je odvisna od mnogih situacijskih spremenljivk: velikosti podjetja, tehnologije, ciljev in strategije podjetja. S stališča intenzivnosti izvajanja projektov lahko podjetja v grobem razdelimo v dve kategoriji:

- podjetja, ki večino svojega dobička ustvarijo z izvajanjem projektov za zunanje naročnike. Tipične panoge za to skupino so arhitektura, svetovanje, gradbeništvo, razvoj programske opreme…,
- neprojektno usmerjena podjetja, kot so proizvodna in finančna podjetja.

Seveda bodo projektno usmerjena podjetja težila k organizaciji, ki bo podpirala organiziranje in izvedbo projektov, medtem ko je izvajanje projektov v neprojektno usmerjenih podjetjih težje izvedljivo.

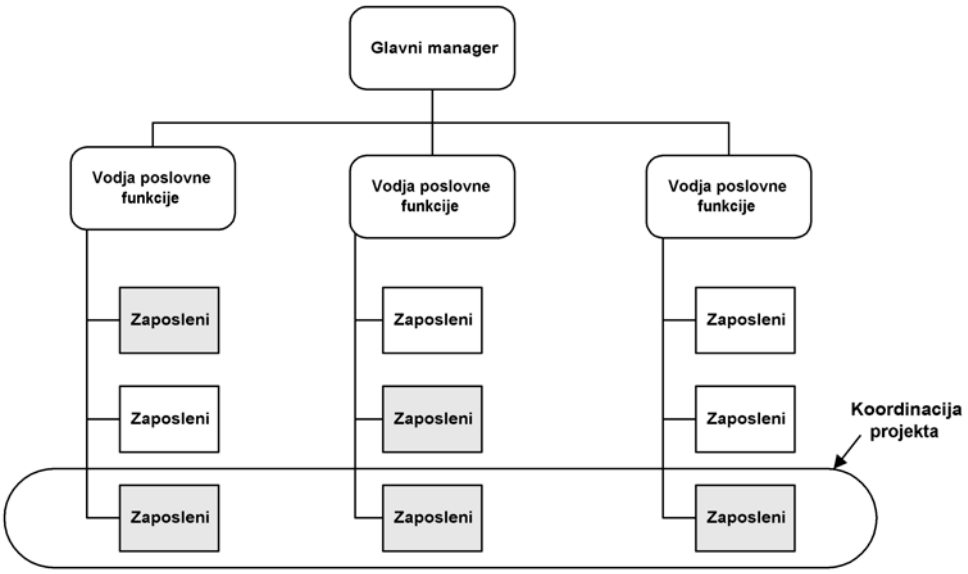
Najbolj tipične oblike organizacijskih struktur so: enostavna, poslovno-funkcijska, produktno-matrična, projektno-matrična in čista projektna. V nadaljevanju bomo na kratko predstavili primernost posameznih oblik za učinkovito izvajanje projektnega managementa.

Enostavna organizacijska struktura je značilna za majhna podjetja z nekaj zaposlenimi. V takšnih podjetjih je težko govoriti o projektnem managementu, saj je lastnik podjetja večinoma vodi vse naloge v podjetju. Glede na majhnost podjetja tudi ne moremo pričakovati velikega števila projektov, ki se izvajajo istočasno.

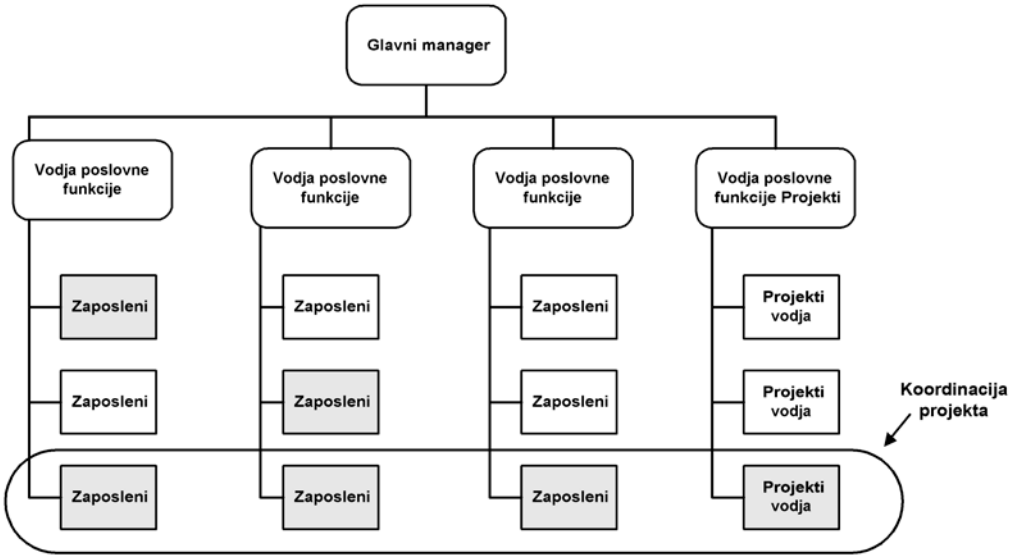
Osnovna značilnost poslovno-funkcijske oblike organizacije je delitev druge hierarhične ravni na poslovne funkcije (npr. nabava, proizvodnja, prodaja, finance, kadri). Za projektno vodenje je takšna struktura precej neprimerna. Projekti namreč združujejo zaposlene z različnimi znanji, ti pa so v poslovno-funkcijski obliki organizacije zaposleni po različnih poslovnih funkcijah. Povezava med posameznimi funkcijami je slaba, zato je za izvedbo projekta potrebno veliko medsebojnega usklajevanja. Interesi poslovne funkcije so večinoma pred interesi projekta, tako da je zelo verjetno odstopanje od časovnega plana projekta.

Produktno-matrična oblika organizacije je značilna za podjetja, kjer imajo posamezni produkti posebno pomembno vlogo. Značilno je notranje in zunanje zbiranje informacij vezanih na proizvodnjo in prodajo posameznega proizvoda, oblikovanje dolgoročne strategije razvoja proizvoda, sodelovanje pri oblikovanju in kontroli poslovnih funkcij vezanih na proizvod. S stališča projektnega managementa je produktna organiziranost že nekoliko bolj primerna, saj je produktni vodja neposredno odgovoren za aktivnosti povezane s produktom. Še vedno pa se produktni vodja srečuje s težavo, da sodelujoči pri

produktu pripadajo različnim poslovnim funkcijam. Prihaja do podvajanja odgovornosti in vprašanja prioritet med interesi produkta in poslovne funkcije.
Projektno-matrična organizacijska struktura je podobna produktno-matrični, z glavno razliko, da so produkte enote stalne, medtem ko so projekti začasni in neponovljivi. PMI (1996, str. 19) razdeli projektno-matrično obliko organizacije še nekoliko natančneje, in sicer na šibko in močno projektno-matrično organizacijo. Za lažjo predstavo sta ti dve obliki prikazani na slikah 2-2 in 2-3 (sodelujoči na projektu so obarvani sivo).



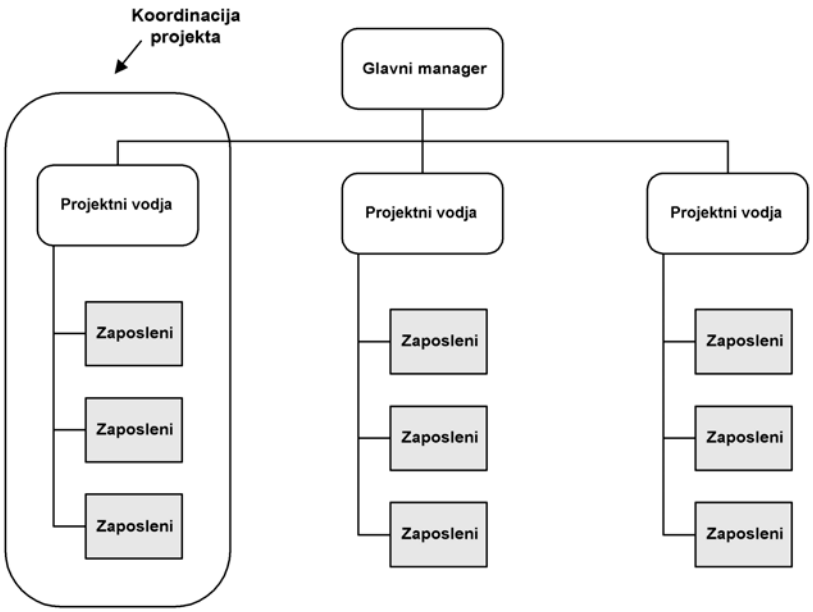
Slika 2-2: Šibka projektno-matrična oblika organizacije.



Slika 2-3: Močna projektno-matrična oblika organizacije.

Šibka projektno-matrična organizacija ohranja veliko lastnosti poslovno-funkcijske organizacije. Pristojnosti projektnega vodje so omejene, nastopa bolj v vlogi koordinatorja oziroma pospeševalca. Srečuje se s težavami pri usklajevanju zaposlenih v različnih poslovnih enotah, funkcijo projektnega vodje pa opravlja vzporedno z ostalimi nalogami.
V nasprotju s šibko projektno-matrično organizacijo je močna projektno-matrična organizacija že veliko bolj primerna za uspešen projektni management. Vodje projektov so

združeni v svoji poslovni funkciji, celoten delovni čas se posvečajo vodenju projektov in imajo povečana pooblastila. Prav tako imajo na voljo sodelavce, ki skrbijo za urejanje projektne dokumentacije.
Najbolj projektno usmerjena oblika organizacijske strukture je projektna oblika organizacijske strukture (slika 2-4).



Slika 2-4: Projektna oblika organizacije.

Projektna oblika organizacije je zelo dinamična struktura, v kateri se zaposleni velikokrat selijo iz projekta na projekt. Večina zaposlenih dela na projektih, odgovornosti in pooblastila projektnih vodij so zelo velika. Zaposleni so sicer še vedno združeni v oddelke, vendar poročajo neposredno vodji projekta.
Večina podjetij dejansko vključuje več od naštetih oblik organizacije, tako da v realnosti težko govorimo o čisto projektni ali čisti poslovno funkcijski obliki organizacije.
Naštete funkcionalnosti posameznih oblik organizacije podjetij lepo združuje tabela 2-1 (PMI, 1996, str. 18), ki prikazuje lastnosti projekta za različne oblike organizacije podjetij:

	Poslovno-funkcijska	**Šibka projektno-matrična**	**Močna projektno-matrična**	**Projektna**
Pristojnosti projektnega vodje	Majhne	Omejene	Visoke	Visoke, skoraj absolutne
Procent dela zaposlenih na projektih	Praktično ničen	0-25 %	50-95%	85-100%
Zasedenost projektnega vodje na projektu	Delna	Delna	Polna	Polna

Tabela 2-1: Primerjava različnih oblik organizacijskih struktur.

2.2.2 Projektna organizacija

V prejšnjem poglavju smo govorili o položaju projektne organizacije znotraj podjetja. V tem poglavju pa bo govora o organiziranosti znotraj projekta, oziroma o različnih vlogah, ki nastopajo pri projektu. Različni avtorji in metodologije pojmujejo organizacijo projekta na različne načine. V praksi se prav tako pojavljajo različni izrazi za praktično enake vloge na projektu (na primer izraza nadzornik in skrbnik).

Projektna metodologija, ki jo uvaja institut PMI, definira na projektu naslednje vloge (PM BOK, 1996, str. 15):

- **projektni manager:** oseba odgovorna za upravljanje projekta,
- **stranka:** posameznik ali podjetje, ki bo uporabljalo končni izdelek ali storitev projekta,
- **izvedbeno podjetje:** podjetje, ki je najbolj povezano z izvedbo projekta,
- **pokrovitelj, sponzor:** posameznik ali skupina znotraj izvedbenega podjetja, ki zagotavlja finančne vire, potrebne za izvedbo projekta.

Poleg te grobe delitve ostaja še mnogo pogledov na sodelujoče pri projektu. Lahko jih razdelimo na notranje ali zunanje sodelavce, stalne in pogodbene, lastnike in ustanovitelje… Zavedati se moramo, da imajo različni člani projekta ponavadi različne interese. Naročnik novega informacijskega sistema bo zahteval čim nižje stroške, sistemski analitik tehnično popolnost, izvajalec pa čim večji dobiček. Usklajevanje teh različnosti je ena ključnih funkcij projektnega managementa.

Zelo dobro definira vloge in odgovornosti udeleženih pri projektu Rant et al. (1995, str.37). »Projektno delo je treba organizirati tako, da se razdeli udeležence v:

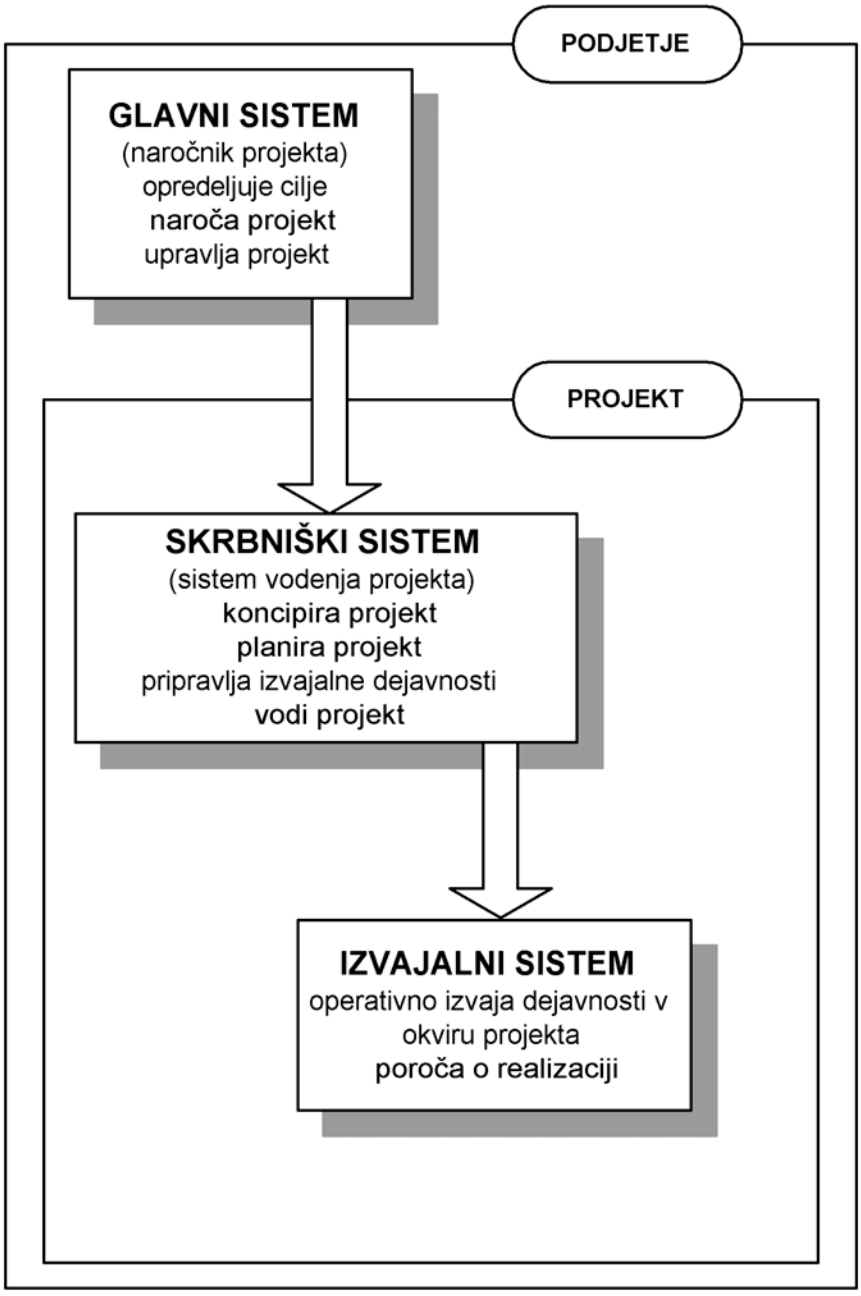
- **glavni sistem projekta** – to je naročnik projekta, ki projekt naroča, usmerja programiranje cilja projekta in upravlja projekt,
- **sistem vodenja in skrbništva projekta,** ki vodi izvajanje projekta in predstavlja projektno organizacijo.
- **sistem izvajanja projekta,** ki izvaja dejavnosti projekta.«

Glavni sistem projekta torej predstavlja naročnik, katerega najpomembnejša naloga je definiranje končnega cilja projekta. Naročnik določa vodjo projekta in kontrolira njegovo uspešnost. Ob zaključku projekta naročnik prevzame rezultat, produkt ali storitev projekta.

Vodja projekta je oseba, ki operativno vodi projekt. Odgovarja neposredno naročniku. V primeru velikih projektov lahko vodja projekta v soglasju z naročnikom formira projektno skupino. To ponavadi sestavljajo strokovnjaki s področij, pomembnih za izvedbo projekta. Vodja projekta lahko definira tudi skrbnika projekta, ki skrbi za operativno izvedbo projektnih dejavnosti. Za lažje izvajanje projekta lahko vodja določi tudi strokovnega tajnika projekta, ki opravlja administrativna dela povezana s projektom: razpisuje sestanke, sestavlja zapisnike, ureja projektno dokumentacijo.

Tretji nivo projekta je sistem izvajanja projekta. Za vsako dejavnost vodja določi vodjo dejavnosti. Pri bolj obsežnih dejavnostih govorimo tudi o delnih projektih ali podprojektih. Ti predstavljajo večjo skupino dejavnosti. Odgovorna oseba za izvedbo podprojekta v času

trajanja podprojekta tesno sodeluje s projektno skupino, lahko je tudi njen član. Na najnižjem nivoju operativno izvedbo prevzamejo izvajalci. Ti so člani projekta samo v času dela na dejavnosti. Izvajalne skupine so lahko notranje (samo zaposleni v izvedbenem podjetju), zunanje (zaposleni v drugih podjetjih) ali pa mešane. V primeru specifičnih del, ki zahtevajo angažiranje specialistov, govorimo o podizvajalcih. Podizvajalci tako kot izvajalci opravljajo točno določeno delo in po njegovi izvedbi izgubijo zvezo s projektom.
Rant et al. (1995, str.38) je projektne nivoje, vloge in pooblastila predstavil tudi shematično, kot je prikazano na sliki 2-5.



Slika 2-5: Organizacijski sistemi v projektu.

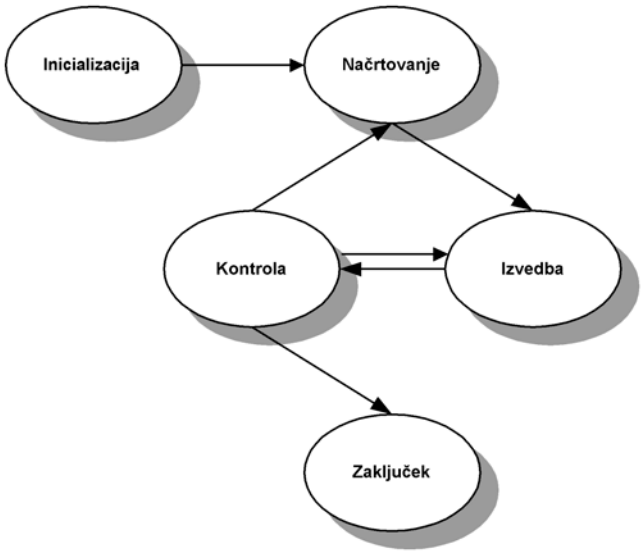
2.3 Proces projekta

Po definiciji je proces skupek aktivnosti, ki prinašajo končni rezultat. Projekti so sestavljeni s skupine procesov. Identifikacija procesov projekta in njihovih povezav se pri različnih projektnih metodologijah nekoliko razlikuje. Tako na primer PMI metodologija opredeli 5 procesov (PM BOK, 1996, str. 28): inicializacijo, načrtovanje, izvedbo, kontrolo in zaključek, medtem ko Rant et al. (1995, str.78) opredeli procese kot: inicializacijo, koncipiranje, definiranje in izvedbo. Prav tako Rant govori o fazah projekta, medtem ko PMI definira vseh 5 procesov znotraj vsake faze projekta, faze pa medsebojno zaporedno poveže.

V nalogi bom nekoliko podrobneje predstavil proces projekta kot ga opisuje metodologija PMI. PMI metodologija loči pri projektu 5 glavnih procesov (znotraj vsake faze):

- **inicializacija** – zazna potrebo po projektu in zagotovi, da je projekt začne,
- **načrtovanje** – zamišljanje izvedljive delovne sheme, po kateri naj projekt zagotovi poslovno potrebo, ki je sprožila projekt,
- **izvedba** – usklajevanje ljudi in virov v želji uspešno izvesti planirano delo,
- **kontrola** – spremljanje napredovanja projekta in zagotavljanje, da so izpolnjeni cilji projekta,
- **zaključek** – pregled rezultatov projekta in formalen zaključek projekta.

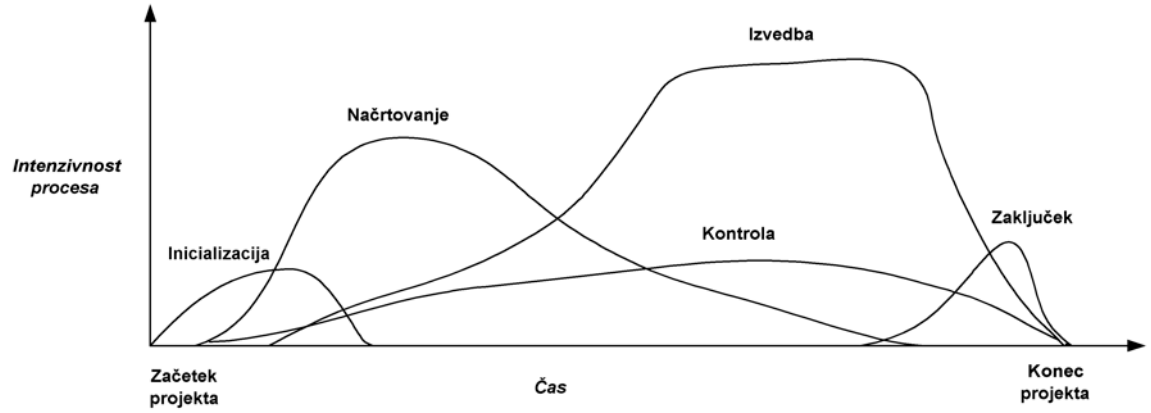
Seveda so procesi medsebojno povezani. Rezultati posameznih procesov so vhodni podatki procesov, ki sledijo. Shematično lahko prikažemo sodelovanje med posameznimi procesi s sliko 2-6 (puščice nakazujejo tok podatkov in dokumentov med procesi):



Slika 2-6: Povezava med procesi v projektu.

Vsak proces se sestoji iz vhodnih podatkov, orodij in tehnik ter izhodnih podatkov. Vhodni podatki so dokumenti na podlagi katerih se v procesu izvajajo ustrezne aktivnosti. Orodja in tehnike so mehanizmi kako vhodne podatke preoblikovati v izhodne, izhodni podatki pa predstavljajo rezultate procesa.

Projektni procesi se med seboj tudi časovno prepletajo, kar pomeni, da časovne meje med njimi niso ostro ločene. Na začetku prevladuje proces inicializacije, nato je prevladujoč proces načrtovanja, sledi izvedba in na koncu zaključek. Proces kontrole v bistvu spremlja vse ostale procese, najbolj intenziven pa je v procesu izvedbe. Ilustrativno lahko prekrivanje procesov prikažemo s sliko 2-7 (PM BOK, 1996, str. 29):



Slika 2-7: Prepletanje procesov v projektu.

Seveda so procesi in še posebej orodja uporabljena v procesih močno odvisni od vsebine, področja in obsega projekta, vseeno pa lahko govorimo o procesih skupnih večini projektov v večini aplikativnih področij. Metodologija PMI podrobno razgradi vsakega od projektnih procesov, v okviru dela pa bomo predstavili samo grobo razdelitev. Razdelitev v bistvu pomeni razčlenitev posameznih elips s slike 2-6.

2.3.1 Inicializacija

Proces inicializacije je pri grobi delitvi samostojen proces, katerega rezultat je začetek projekta oziroma naslednje faze projekta.

2.3.2 Načrtovanje

Načrtovanje projekta ima pri projektu veliko vlogo, saj je rezultat projekta edinstven produkt ali storitev. Obseg načrtovanja je sorazmeren z obsegom projekta in s številom informacij, ki so nam na voljo pri procesu planiranja. Načrtovanje je podvrženo pogostim iteracijam, saj posamezne spremembe na projektu praviloma povzročijo več potrebnih popravkov plana. Tako na primer sprememba končnega datuma projekta, vpliva na planiranje virov, stroškov, ali celo na zastavljene projektne cilje. Načrtovanje projekta ni do potankosti definirana znanost, tako da lahko dva projektna tima za isti projekt razvijeta precej različna plana.

Po PMI metodologiji (PM BOK, 1996, str.30) je proces načrtovanja razdeljen v glavni in podporni proces. Glavni proces vsebuje:

- **načrtovanje obsega** – definiranje dokumenta, ki določa obseg projekta in predstavlja osnovo za nadaljnje projektne odločitve
- **definiranje obsega** – podrobnejša opredelitev posamezni projektnih ciljev na manjše, bolj obvladljive komponente,

- **definiranje aktivnosti** – ugotavljanje posameznih aktivnosti, ki morajo biti izvedene za zagotavljanje posameznih projektnih rezultatov,
- **določitev zaporedja aktivnosti** – ugotavljanje povezanosti in odvisnosti med posameznimi aktivnostmi,
- **ocena trajanja aktivnosti** – ugotavljanje dela potrebnega za izvedbo posameznih aktivnosti,
- **časovna opredelitev aktivnosti** – časovna opredelitev aktivnosti glede na njihovo povezanost, trajanje in vire,
- **načrtovanje virov** – določanje virov (ljudje, oprema, material) potrebnih za izvedbo aktivnosti,
- **ocena stroškov** – ocena predvidenih stroškov potrebnih za izvedbo posameznih aktivnosti,
- **razvoj plana projekta** – združevanje rezultatov posameznih podprocesov v skupen dosleden in jasen dokument.

Glavni proces planiranja spremlja še vrsta podpornih procesov, katerih obseg je odvisen od narave projekta. Kljub temu, da se proces podpore ne izvaja neprekinjeno je obvezen del načrtovalnega procesa in vsebuje:

- **načrtovanje kakovosti** – ugotavljanje standardov kvalitete potrebnih pri projektu in ugotavljanje, kako jih zagotoviti,
- **načrtovanje projektne organizacije** – ugotavljanje in določanje projektnih vlog, odgovornosti in načina poročanja pri projektu,
- **pridobivanje osebja** – pridobivanje človeških virov potrebnih za izvedbo projekta,
- **načrtovanje komuniciranja** – določanje informacij in komunikacijskih kanalov v projektni organizaciji. Ugotavljanje kdo potrebuje informacije, kdaj jih potrebuje in kako mu bodo dostavljene.
- **ugotavljanje tveganja** – ugotavljanje potencialnih nevarnosti, ki bi lahko vplivale na uspešno izvajanje projekta,
- **merjenje tveganja** – kvalitativno ocenjevanje tveganja in možna povezava z rezultati projekta,
- **načrtovanje nabave** – določanje kaj je potrebno nabaviti in kdaj,
- **načrtovanje zahtev** – dokumentiranje zahtev končnega produkta ali storitve.

2.3.3 Izvedba

Proces izvedbe dobiva vhodne podatke iz načrtovanja in kontrole, predaja pa rezultate procesu kontrole. Sestoji se iz naslednjih procesov:

- **izvajanje projektnega plana** – sledenje projektnemu planu z izvajanjem projektnih aktivnosti,
- **preverjanje obsega** – formalizirano sprejetje obsega projekta.
- **zagotavljanje kakovosti** – periodično preverjanje izvajanja projekta, z namenom da projekt zadovolji predpisane standarde kakovosti,

- **razvoj projektnega tima** – razvoj posameznikovih in skupinskih znanj za bolj učinkovito izvajanje projekta,
- **distribucija informacij** – zagotavljanje, da so potrebne informacije dostopne udeleženim na projektu v sprejemljivem času,
- **izbira ponudnikov** – izbiranje ustreznih dobaviteljev za potrebe nabave projekta,
- **vzdrževanje pogodb** – nadzor nad pogodbami z zunanjimi dobavitelji udeleženimi pri projektu.

2.3.4 Kontrola

Za ugotavljanje odstopanj od projektnega plana, potrebuje projekt nenehno merjenje učinkovitosti izvedbe projekta. Spremljanje izvedbe in obravnavanje ugotovljenih odstopanj so del procesa kontrole. Ugotovljena odstopanja lahko sprožijo proces ponovnega planiranja. Del kontrole projekta so tudi preventivni ukrepi. Ti skušajo odpraviti probleme, ki bi lahko vplivali na učinkovito izvedbo projekta. Proces kontrole razdelimo na naslednje manjše procese:

- **kontrola celotnega projekta** – koordiniranje sprememb vezanih na celoten projekt,
- **kontrola obsega** – spremljanje sprememb obsega projekta,
- **kontrola časovnega plana** – koordiniranje sprememb povezanih s časovnih planom projekta,
- **kontrola stroškov** – spremljanje sprememb pri stroških projekta,
- **kontrola kvalitete** – pregled rezultatov projekta in ugotavljanje, če se skladajo s kakovostnimi standardi opredeljenimi za projekt. Odstranitev faktorjev, ki vplivajo na nezadostno učinkovitost projekta.
- **poročanje o stanju projekta** – zbiranje in analiza informacij o izvedbi projekta. Proces vključuje poročanje o statusu, napredovanju projekta in napovedovanje nadaljnjih aktivnosti.

2.3.5 Zaključek

Zadnji proces pri projektu oziroma fazi projekta je zaključek. Sledi procesu kontrole, ko ta ugotovi, da so zaključene vse projektne aktivnosti in da so izpolnjeni zastavljeni projektni cilji. Zaključek vsebuje zbiranje informacij zbranih med projektom, tako da so te dostopne in uporabne pri naslednjih projektih. Proces vključuje tudi formalno zaključitev projekta.

Opisani procesi so groba opredelitev procesov, ki nastopajo v večini projektov iz različnih področij. Naslednje poglavje govori o projektih s področja razvoja programskih rešitev. Skušali bomo ugotoviti v čem so projekti s področja razvoja programske opreme specifični glede na splošno znanje o vodenju projektov. Poudarili bomo, kateri procesi so za to panogo še posebej pomembni in katera orodja uporabljamo pri procesu razvoja programskih rešitev.

3 Projektni management pri razvoju programskih rešitev

Projekti s področja informacijske tehnologije in razvoja programskih rešitev postavljajo projektnemu managementu še posebne izzive. Razvoj programske opreme poteka v dinamičnem, hitro spreminjajočem se okolju, brez dobro definiranih industrijskih standardov in brez zanesljivih kvalitativnih podatkov iz sorodnih projektov. Tehnologija se praktično spreminja iz dneva v dan, tako da panoga zahteva nenehno sledenje spremembam in izobraževanje zaposlenih. Projektni pristop pri razvoju programskih rešitev pomeni delo v dinamičnem okolju, izraža potrebe po visoki učinkovitosti, zahteva optimalno izrabo kadrov in hiter pogled v tekoče stanje del na projektu.

McNeice Filler (2001) v članku podaja analizo svetovalne skupine Gartner, »da do leta 2002, 75 procentov projektov v informacijski tehnologiji ne bo doseglo zastavljenih ciljev, zaradi napak pri planiranju projektov«. V istem članku najdemo tudi mnenje svetovalnega podjetja KPMG, da tehnologija ni poglavitni razlog za ogromen odstotek neuspelih projektov. Raziskava v 256 podjetjih je ugotovila, da samo 14 procentov neuspešnih projektov lahko pripišemo nezmožnosti obvladovanja tehnologije. Ostalih 86 procentov projektov ne uspe zaradi tipičnih managerskih hib: neustrezno definiranih ciljev (17 procentov), neznanega obsega (17 procentov), pomanjkanja učinkovite komunikacije (20 procentov) in pomanjkanja managerskih znanj (32 procentov).

Glede na vse našteto lahko ugotovimo, da je upravljanje projektov s področja razvoja programske opreme težko delo. Tehnološka skupina Pointe v svoji študiji z leta 2001 izpostavlja 5 najbolj pogostih vzrokov, zakaj projekti s področja razvoja programskih rešitev tako pogosto ne zadovoljijo pričakovanj. Ti vzroki so:

- **nerealen plan** – če je preveč dela načrtovano za premajhno časovno obdobje, so težave neizogibne.
- **nedefiniran proces razvoja** – če razvojni proces ni jasen in dobro definiran, je zelo verjetno, da bo projekt zabredel v težave.
- **Slabo opredeljene uporabniške zahteve** – če so zahteve nejasne, nepopolne, preveč splošne ali neprimerne za testiranje, bodo nastale težave.
- **neustrezno testiranje** – brez temeljitega testa ne bo nihče vedel ali je program dober, dokler se naročnik ne bo pritožil ali se bo sistem postal nefunkcionalen.
- **slaba komunikacija** – če je komunikacija med člani projektne skupine slaba in so projektni podatki nedostopni in nepregledni, bodo nastopile težave.

Če je pri projektu navzoča katerakoli od naštetih težav, razvita programska ne bo dosegla želenih rezultatov oziroma projekt ne bo uspešen. Kvalitetna programska rešitev je rešitev, ki je praktično brez hroščev, razvita v planiranih časovnih okvirjih, ustreza definiranim zahtevam in jo lahko vzdržujemo.

V nadaljevanju bomo vsako od naštetih težav podrobneje analizirali in ponudili metode in orodja, ki so nam pri vodenju projektov na voljo, da se izognemo tem najbolj pogostim napakam. Težave in odgovori so zbrani v spodnji tabeli:

Težava	**Odgovor**

Nerealen plan	Obvladovanje časa in stroškov
Nedefiniran proces razvoja	Uporaba razvojnega modela
Slabo opredeljene uporabniške zahteve	Obvladovanje uporabniških zahtev
Neustrezno testiranje	Testiranje in zagotavljanje kvalitete
Slaba komunikacija	Programska rešitev za vodenje projektov.

Tabela 3-1: Najpogostejši vzroki težav pri informacijskih projektih in odgovori nanje.

3.1 Obvladovanje časa in stroškov

Eden osnovnih pogojev za uspešno izvedbo projekta je pravilna ocena in obvladovanje časovnih rokov in stroškov projekta (poglavje 2.3.2.). Pravilna ocena projektnih rokov in stroškov predstavlja osnovo za učinkovito izvedbo in kontrolo projekta.

Osnova za načrtovanje terminskega plana in stroškov pri projektih razvoja programske opreme je ocena dela in trajanja projektnih aktivnosti. Predvideno delo na aktivnostih se izraža v urah potrebnih za izvedbo aktivnosti, trajanje pa je število koledarskih dni potrebnih za končanje posamezne aktivnosti. Ocena potrebnega dela in trajanja aktivnosti pri projektih razvoja programskih rešitev je težavna iz več razlogov: nasprotujočih si ciljev projekta, pomanjkanja podrobnega opisa projekta, potrebe po uporabi novih tehnologij, različnem obsegu ponovno uporabljene kode, itd.

Terminsko planiranje projektnih aktivnosti temelji na tehniki mrežnega planiranja. Mrežni diagram grafično predstavlja zaporedje medsebojno povezanih aktivnosti, potrebnih za realizacijo projekta. Mrežni plan sestavljajo aktivnosti in dogodki. Rant et al. (1995, str.175) opredeli projektno aktivnost z naslednjimi pogoji:

- ima določeno trajanje,
- ima začetni in končni dogodek,
- povezuje natanko dva dogodka,
- začne se z dogodkom, ki pogojuje njen začetek,
- večinoma je vezana na vire.

Primeri projektnih aktivnosti so: koncipiranje pogodbe, izdelava programerskih specifikacij, izdelava načrta programske rešitve, izdelava uporabniških navodil, izobraževanje končnih uporabnikov.

Dogodek je stanje na projektu, ki nastopi, ko se določena aktivnost začne ali konča. Za dogodke velja:

- dogodek lahko sproži eno ali več aktivnosti,
- med dvema zaporednima dogodkoma je lahko samo ena aktivnost.

Primeri dogodkov pri projektu so: podpis pogodbe, začetek programiranja, konec izobraževanja uporabnikov, predaja programske rešitve v produkcijo.

Mrežno planiranje temelji na matematični teoriji grafov, ki ponuja več metod mrežnega planiranja. Osnovna delitev metod je na deterministične in stohastične. Pri determinističnih so cilji, aktivnosti in dogodki točno določeni, verjetnost njihove pojave v projektu je enaka 1. Stohastične metode za razliko od determinističnih operirajo z verjetnostjo: definiran cilji

ni nujno tudi dosegljiv, aktivnosti niso točno poznane, roki aktivnosti so določeni samo v določenih časovnih mejah.

Naslednja delitev metod je na metode, ki so orientirane na aktivnosti in metode, ki so orientirane na dogodke. Metode orientirane na aktivnosti, temeljijo na predpostavki, da se naslednja aktivnost lahko začne izvajati šele, ko je končana predhodna aktivnost. Metode orientirane na dogodke, pa trdijo, da se nek dogodek zgodi samo, če se je zgodil njegov predhodni dogodek. Oba vrsti metod pripeljeta do istih rezultatov, različen je način, kako pridemo do rezultata.

V praksi največkrat uporabljene metode mrežnega planiranja so:

- CMP – Critical Path Method, metoda kritične poti,
- PERT – Program Evaluation and Review Technique, tehnika ocene in preverjanja programa,
- MPM – METRA Potential Method, METRA metoda potencialov in
- GERT – Graphical Evaluation and Review Technique, grafična tehnika ocenjevanja in preverjanja programa.

V nadaljevanju bomo nekoliko podrobneje predstavili metodo CPM.

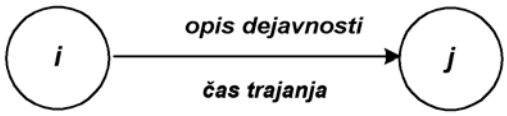
3.1.1 Metoda kritične poti

Metoda kritične poti je bila razvita v letih 1956 in 1957 v Združenih državah Amerike v sodelovanju podjetij Pont in Remington Rand Univac. Metoda se sestoji iz petih korakov:

1. **Analiza strukture projekta** – definira aktivnosti in opredeli njihovo medsebojno odvisnost. Rezultat so grafično prikazane aktivnosti, njihovo zaporedje in odvisnosti med njimi.
2. **Analiza časov** – določi čas trajanja posameznih aktivnosti, v odvisnosti od ostalih aktivnosti opredeli roke in določi čas potreben za izvedbo celotnega projekta in njegovih posameznih delov.
3. **Izbor virov za posamezne aktivnosti** – določi kateri ljudje bodo izvedli posamezne aktivnosti in posledično določi obremenjenost virov.
4. **Analiza in optimizacija stroškov** – določi stroške za posamezne aktivnosti in celoten projekt.
5. **Optimizacija zasedenosti virov** – uskladi potek aktivnosti z zasedenostjo posameznikov udeleženih na projektu.

Obvezni sta prvi dve fazi. Čeprav so ostale faze neobvezne, lahko njihovo opuščanje pripelje do pomanjkanja virov, prezasedenosti zaposlenih in povečanja stroškov.

Rezultat prve faze metode so grafično prikazane aktivnosti. Metoda kritične poti je metoda orientirana na aktivnosti, zato dogodke v mrežnem planu predstavimo z oštevilčenimi krogi na začetku in koncu aktivnosti. Aktivnosti so ponazorjene z usmerjenimi puščicami, ki kažejo od začetnega do končnega dogodka. Nad puščico lahko dodamo še opis aktivnosti, pod njo pa čas trajanja aktivnosti. Splošen primer aktivnosti med dogodkoma *i* in *j*, je prikazan na sliki 3-1.



Slika 3-1: Aktivnost pri metodi kritične poti.

Osnova za risanje mrežnega plana je spisek vseh aktivnosti in ugotovitev odvisnosti med njimi. Ko so v mrežni plan vrisane vse aktivnosti, sledi označevanje dogodkov. Dogodke označujemo s celimi števili, tako da ima končni dogodek aktivnosti večjo številko od začetnega.

Po dokončanem mrežnem planu nastopi analiza časov. Opredeli se osnovna časovna enota (na primer delovni dan, ura) in trajanje posameznih aktivnostih v predpisani časovni enoti. Predvideni časi trajanja aktivnosti se vnesejo v seznam aktivnosti. Pri metodi CPM je predvideno trajanje aktivnosti določeno z eno samo vrednostjo – točkovna ocena trajanja aktivnosti.

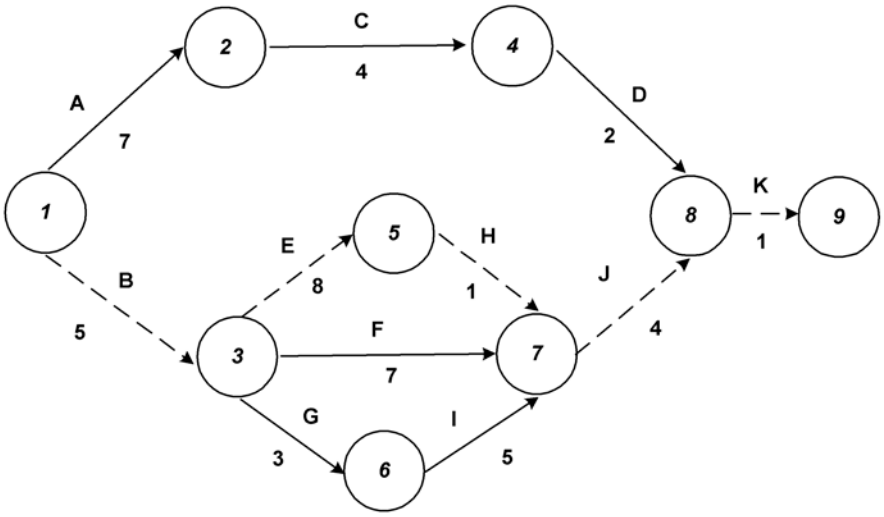
Vsaka aktivnost v mrežnem planu ima torej točno določeno predvideno trajanje, kdaj točno pa se bo izvedla, je odvisno od dogodkov, ki aktivnost omejujejo. Noben dogodek se ne more dogoditi pred terminom, ki ga imenujemo najzgodnejši rok dogodka, oziroma se končati po roku, ki ga imenujemo najkasnejši rok dogodka. Iz angleščine izhajata oznaki za najzgodnejši začetek E (earliest) in najpoznejši konec L (latest). Ko združimo trajanje aktivnosti s tipičnimi časi dogodkov, dobimo za dogodke in aktivnosti naslednje štiri tipične čase:

- najzgodnejši začetek ES (early start) – najzgodnejši rok za začetek dogodka,
- najkasnejši začetek LS (latest start) – najkasnejši rok, da se začne aktivnost in ne poruši časovnega zaporedja v mreži,
- najzgodnejši konec EF (early finish) – najzgodnejši možni rok zaključka aktivnosti in
- najkasnejši konec LF (latest finish) –najkasnejši možni rok končnega dogodka.

Maksimalni čas, ki je na voljo za izvedbo neke aktivnosti je torej razlika med najkasnejšim koncem in najzgodnejšim začetkom. Če je ta razlika večja od trajanja aktivnosti, ima aktivnost časovno rezervo (angleško float). Aktivnost se lahko izvede kadarkoli znotraj maksimalnega trajanja, ne da bi s tem ogrozila izvajanje ostalih aktivnosti na projektu. Če pa je maksimalna razlika enaka trajanju aktivnosti, se mora aktivnost izvesti točno v predpisanem času. Taka aktivnost nima časovne rezerve in jo imenujemo kritična aktivnost.

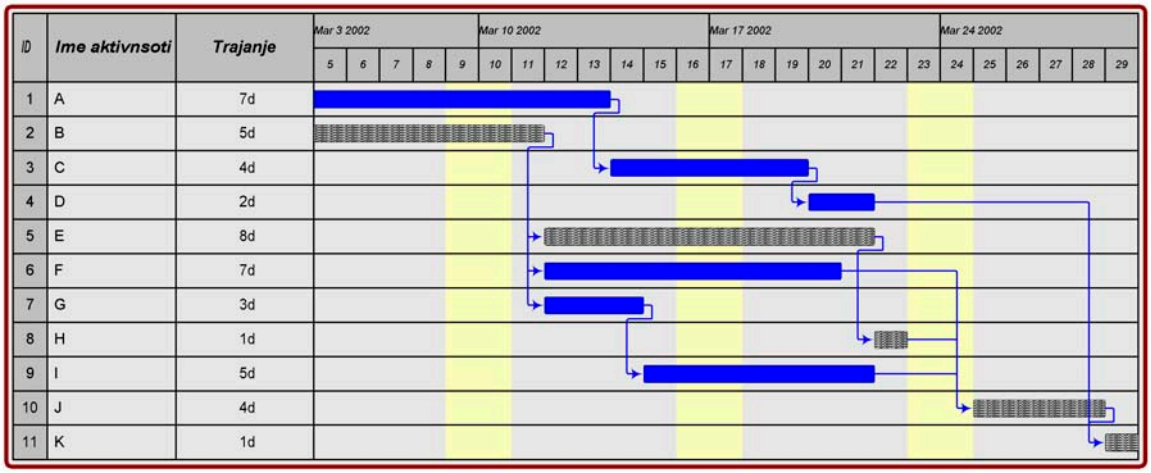
Kritične aktivnosti nas pripeljejo tudi do definicije kritične poti. Kritična pot je tista pot skozi mrežni diagram, na kateri ležijo samo kritične aktivnosti. Takšna pot skozi mrežni plan je časovno najdaljša in ne dopušča odstopanj od predpisanih rokov. Trajanje kritične poti določa tudi čas trajanja celotnega projekta. Kritičnih poti je lahko v terminskem planu več, seveda pa so vse enako dolge.

Izračun kritične poti temelji na enostavni algebri, ki pa je kar precej zamudna. Danes ponuja izračun kritične poti praktično vsak računalniški program namenjen časovnem planiranju. Primer enostavnega mrežnega plana z vrisano kritično potjo je prikazan na sliki 3-2 (kritična pot je označena črtkano, projekt pa traja 19 časovnih enot):



Slika 3-2: Primer mrežnega plana s kritično potjo.

V praksi smo bolj kot na mrežni diagram navajeni na predstavitev časovnega plana v gantogramu. Gantogram predstavi aktivnosti na časovni skali tako, da je dolžina stolpcev sorazmerna s trajanjem aktivnosti. Postopek pretvorbe mrežnega diagrama v gantogram je enostaven. Najprej na navpično os nanesemo dejavnosti, potem pa z risanjem črt opredelimo trajanje aktivnosti glede na določene roke in časovno skalo. Primer pretvorbe zgornjega mrežnega diagrama v gantogram je prikazan na sliki 3-3:

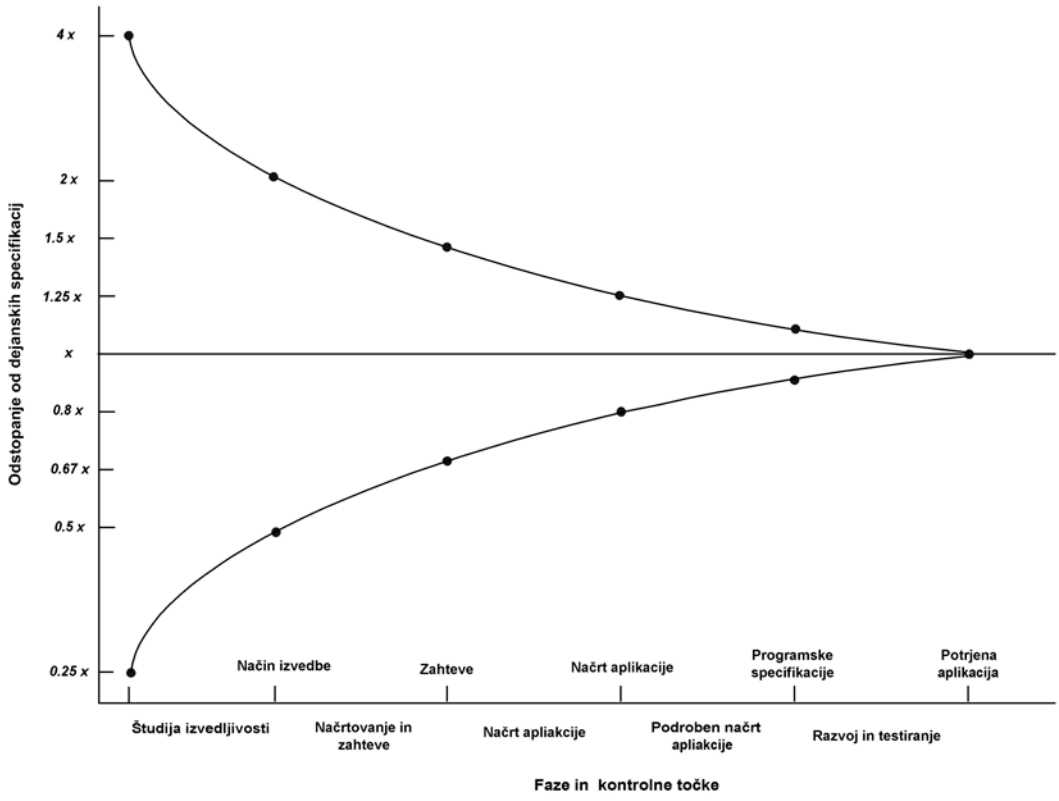


Slika 3-3: Primer gantograma s kritično potjo.

3.1.2 Ocena trajanja in stroškov projekta

Metoda kritične poti služi časovnemu planiranju projekta, neodvisno od področja s katerim se projekt ukvarja. Ocenjeno potrebno delo za posamezne aktivnosti je točno določeno, metoda pa ne ponuja nobene tehnike, kako smiselno oceniti potrebno delo za posamezne aktivnosti in s tem za cel projekt. Seveda se na začetku vsakega informacijskega projekta ponuja prav to vprašanje: Koliko dela in ostalih stroškov bo potrebnih za uspešno končanje projekta? Uspešna ocena je temelj za finančno opredelitev projekta in za planiranje virov na izvajalčevi strani.

Ključnih problem ocenjevanja pri razvoju programskih rešitev je, da je ocena potrebna preden je produkt dobro definiran. Podroben opis funkcionalnosti rešitve je zelo težaven, sploh v zgodnjih fazah projekta. V splošnem se točnost ocen izboljšuje tokom projekta, saj je na voljo vedno več informacij. Boehm (1984), eden pionirjev na področju ekonomike programskega inženirstva, je grafično predstavil kako je točnost ocen odvisna od faze projekta s sliko 3-4 (faze projekta so opredeljene po Waterfall modelu (glej 3.2)):



Slika 3-4: Točnost ocen trajanja in stroškov glede na fazo projekta.

Netočnost ocen je največja v začetku projekta. V študiji izvedljivosti so ta odstopanja do 400% in se nekako ustalijo šele, ko so izdelan načrt programske rešitve (odstopanja do 25%). Iz grafa je očitno, da je dobra ocena stroškov projekta zelo težavna. V izogib netočnostim in napakam že v zgodnji fazi projekta so se razvile različne tehnike ocenjevanja trajanja in stroškov projekta. Tehnike lahko združimo v 5 glavnih skupin:

1. **algoritmični modeli**: ocenjevanje stroškov projekta na podlagi spremenljivk, ki najbolj vplivajo na stroške projekta. Modeli zagotavljajo algoritme kako vhodne podatke, pretvoriti v oceno stroškov.
2. **ekspertne ocene**: temeljijo na mnenju enega ali več strokovnjakov s tehničnim znanjem in poznavanjem projektov s podobno vsebino.
3. **analogija**: ocena stroškov projekta glede na stroške sorodnih, že zaključenih projektov.
4. **pristop od vrha proti dnu**: skupni strošek je ocenjen glede na najbolj tipične lastnostni razvijajoče programske rešitve. Stroški so naknadno razdeljeni na manjše aktivnosti.
5. **pristop od dna proti vrhu**: najprej ocenimo stroške za posamezne programske module. S seštevanjem parcialnih stroškov določimo oceno stroška celotnega projekta.

Nobena od tehnik ni v vseh pogledih boljša od ostalih. Tako je na primer prednost algoritmičnih modelov objektivnost, ponovljivost in analitičnost, njihova težava pa je v točnosti vhodnih podatkov. Vsaka algoritmična napoved je lahko namreč le toliko natančna, kolikor so natančni vhodni podatki. Metode se torej medsebojno dopolnjujejo in ne izključujejo. V praksi je najbolje poiskati ocene po različnih metodah, dobljene rezultate medsebojno primerjati in uskladiti ugotovljena odstopanja.
V nadaljevanju bo nekoliko podrobneje predstavljen eden bolj znanih algoritmičnih modelov, model COCOMO.

3.1.3 COCOMO

Metodo COCOMO (COnstructive COst MOdel) je konec sedemdesetih let prejšnjega stoletja razvil Boehm (1981). Takoj je potrebo poudariti, da se v praksi model COCOMO ne uporablja več, saj ga je zamenjal COMOMO 2. Kljub temu, da se starejša verzija metode ne uporablja več, je dobra osnova za razumevanje principov algoritmičnega ocenjevanja stroškov projekta.
Ocena za osnovo stroškov projekta po metodi COMOMO je velikost programske rešitve in izbran razvojni način. Velikost programske rešitve se meri v številu ukazov izvorne kode v tisočih. Metoda določa tri razvoje načine: osnoven, sredinski in napreden. Osnovni način vključuje krajše programske rešitve, ki ne vsebujejo zapletenih algoritmov in ne zajemajo celotnega podjetja. V napredni način uvrščamo rešitve, ki zahtevajo popolno skladanje z zahtevami, vključujejo veliko uporabnikov in pokrivajo procese v celotnem podjetju. Velikost programske rešitve in razvojni način posredujeta prvo oceno potrebnih človek mesecev za izvedbo projekta. Tabela 3-2 prikazuje enačbe za izračun potrebnih človek mesecev (nominalnih) po metodi COCOMO (ČM – človek meseci, TUPK – tisoči ukazov programske kode, ČR – čas razvoja):

Razvojni način	**Ocena potrebnega dela (v človek mesecih)**	**Priporočen čas trajanja projekta (v mesecih)**
Osnovni	$ČM = 3.2\ (TUPK)^{1.05}$	$ČR = 2.5\ (ČM)^{0.38}$
Sredinski	$ČM = 3.0\ (TUPK)^{1.12}$	$ČR = 2.5\ (ČM)^{0.35}$
Napredni	$ČM = 2.8\ (TUPK)^{1.20}$	$ČR = 2.5\ (ČM)^{0.32}$

Tabela 3-2: Ocena nominalne vrednosti potrebnega dela in tajanja projekta.

Zveza med velikostjo programske rešitve in potrebnim delom torej ni linearna, temveč narašča kot potenčna funkcija. Potenčna osnova je odvisna od razvojnega načina. Pri enostavnih programskih rešitvah je zveza med velikostjo rešitve in potrebnim delom praktično linearna (1.05). Kompleksnejša kot je programska rešitev, bolj je linearnost porušena (pri zahtevnih rešitvah je faktor 1.20). Zanimivo je, da metoda za zahtevnejše projekte, ki zahtevajo isti obseg dela kot enostavnejši, priporoča krajši razvojni čas (potenčna osnova za zahtevne 0.32, pa enostavne pa 0.38).
Osnovno oceno nato pomnožimo s korekcijskim faktorjem, ki se določi z vrednostmi 15 parametrov. Če naštejemo samo nekatere izmed njih: zahtevana zanesljivost rešitve, velikost podatkovne baze, kompleksnost produkta, učinkovitost programerjev, izkušnje s

podobnimi rešitvami in uporabljenim programskim jezikom itd. Vsakega od parametrov ocenimo in določimo vrednosti, ki jih predpisuje metoda (od 0.7 do 1.6). Produkt posameznih faktorjev nam da število, s katerim moramo pomnožiti osnovno oceno. Nominalna ocena potrebnega dela na projektu se torej ustrezno poveča oziroma zmanjša glede na produkt ocenjenih faktorjev, kar ustrezno vpliva tudi na razvojni čas. Boehm je v članku (1984) preizkusil metodo COCOMO pri 63 projektih in v 68 procentih dosegel ocene točnosti v okviru 20 procentov.

Seveda je od časa, ko je bila razvita metoda COCOMO (1981), področje razvoja programske opreme doživelo silovite spremembe. Najbolj značilne so: uporaba novih razvojnih modelov, ponovno uporabljivi programski moduli, nastop novih tehnologij in programskih orodij, objektno orientirano programiranje… V odgovor naštetim spremembam je avtor metode Boehm et al (1995) razvil prenovljeno različico modela.

Glavna novost modela COCOMO 2.0 je spremenjen način ocenjevanja velikosti programske rešitve. Ta vsebuje objektne točke, funkcijske točke in število vrstic programske kode. V začetni fazi projekta, ko je netočnost ocen največja, uporablja metoda za oceno objektne točke. Objektne točke temeljijo na številu ekranov, poročil in ocenjeni stopnji ponovljivosti iz prejšnjih projektov. Na podlagi izkušenj in ekspertne ocene, določimo koliko objektnih točk je moč izvesti v določeni časovni enoti. Tako dobimo prvo oceno potrebnega dela na projektu.

V fazi zgodnjega načrtovanja rešitve poskusimo oceniti koliko tako imenovanih funkcijskih točk vsebuje projekt. Funkcijske točke merijo velikost programske rešitve z določanjem vhodnih in izhodnih podatkov, vnosnimi maskami in velikostjo obdelanih datotek. Šele v kasnejših fazah projekta, ko se točnost ocen dovolj poveča, uporabimo za oceno dejansko število programske kode.

Model COCOMO 2.0 upošteva še veliko drugih metod, kot so nelinearnost zaradi ponovne uporabe programskih modulov, popravki pri oceni korekcijskih faktorjev in druge, ki presegajo okvir tega dela.

V verziji iz leta 1998 je Boehm uporabil 161 različnih projektov za kalibracijo modela. V 75 procentih projektov je bila dosežena 30 procentna natančnost pri oceni potrebnega dela, pri 72 procentih projektov, pa ni bilo potrebno povečati časovnega obsega projekta. Glede na iste statistične točke, je bil model kalibriran še v letih 1999 in 2000.

Končno priporočilo je, da organizacija sama uravnoteži model glede na lastne izkušnje iz predhodnih projektov. Na ta način se lahko učinkovitost modela še drastično poveča, seveda pa je za ustrezno kalibracijo potrebno odlično poznavanje lastnega razvojnega procesa in obilica ekspertnega znanja.

Zaključimo lahko, da je za uspešno obvladovanje časa in stroškov projekta potrebna kopica znanj. V praksi ne obstaja splošno veljaven teoretični model, ki bi zadovoljeval potrebe najrazličnejših tipov razvoja programskih rešitev. Tako je ocena in obvladovanje časa in stroškov še vedno v veliki meri odvisna od ekspertnega znanja, ki pa mora biti podprto tudi z uporabo algoritmičnega modela za oceno trajanja in stroškov projekta.

3.2 Uporaba razvojnega modela

Razvojni model je skupek aktivnosti, ki omogočajo prehod od naročnikovih zahtev do delujoče programske rešitve. Primarno želi razvojni model odgovoriti na vprašanji kaj naj naredimo v nadaljevanju projekta in kako dolgo naj to počnemo. Tu je glavna razlika med razvojnim modelom in razvojno metodologijo. Metodologija se ukvarja z aktivnostmi znotraj posamezne faze projekta, kot so ugotavljanje podatkov in zagotavljanje rezultatov faze projekta. Razvojni model pa predstavlja vodilo za izvedbo celotnega projekta, prehode med fazami, zagotavljanje validacije, testiranja in drugih aktivnosti pomembnih za projekt v najširšem smislu.

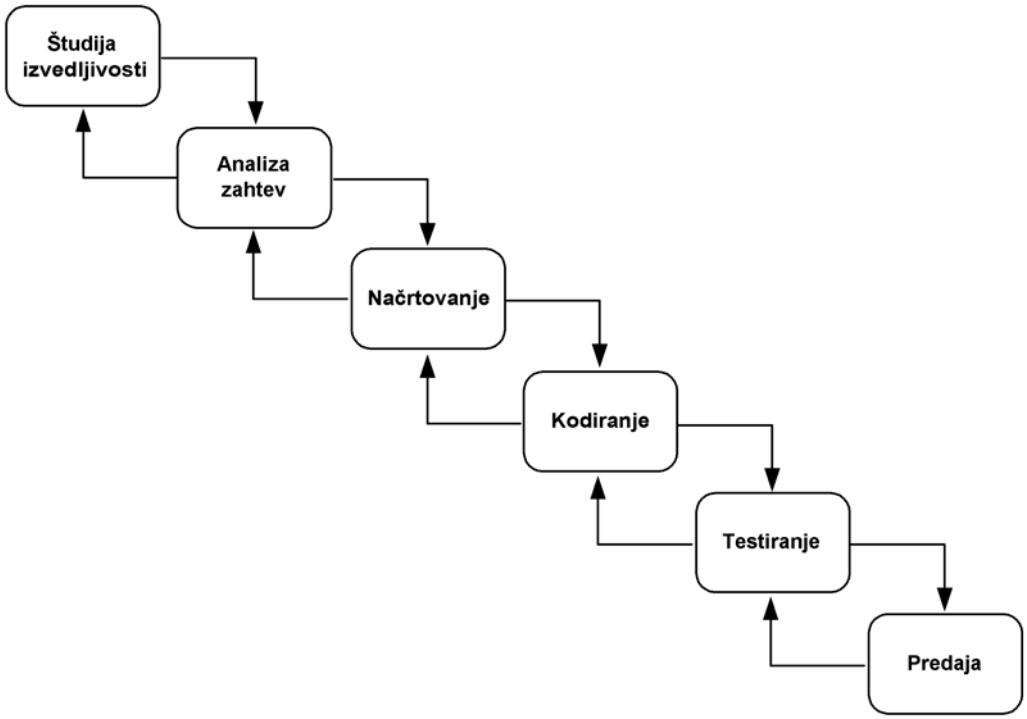
Seveda se najprej ponudi vprašanje, zakaj ne bi razvojni model vseboval samo dveh korakov: kodiranja in popravljanja programskih napak. Model morda celo deluje dokler je rešitev v domeni enega razvijalca, ki je hkrati tudi bodoči uporabnik programske rešitve. Pri nekoliko večjih projektih pa pri takem pristopu naletimo na naslednje probleme:

- Veliko število popravkov programske kode pomeni, da je koda slabo strukturirana, zato se poveča verjetnost napak in stroški popravljanja kode. Ugotovljeno poudarja potrebo po fazi načrtovanja pred kodiranjem.
- Tudi, če je koda dobro strukturirana in delujoča, še ne pomeni, da bo zadovoljila naročnikova pričakovanja. To poudarja potrebo po izdelavi specifikacij pred fazo načrtovanja.
- Brez izdelanega načrta testiranja in kontrole nad spremembami kode, so stoški spreminjanja programske kode zelo veliki. Ugotovljeno poudarja potrebo po načrtovanju, testiranju kode in obvladovanju sprememb.

Z naštetega je očitno, da potrebujemo pri razvoju programske rešitve ustrezen razvojni model. Žal v praksi ne obstaja edinstven razvojni model, ki bi bil primeren za vse različice razvoja programske opreme. Kateri model izbrati je odvisno od aplikativnega področja, tipa organizacije, zahtevnosti in velikosti rešitve ter mnogih drugih faktorjev. V praksi srečamo ogromno različnih razvojnih modelov, v nalogi pa bomo predstaviti štiri tipične: kaskadni, evolucijski, spiralni in razvojni model RUP (Rational Unified Process).

3.2.1 Kaskadni (waterfall) model

Avtor kaskadnega (waterfall) modela je Royce (1970). Kaskadni model je odigral nekakšno pionirsko vlogo med modeli razvoja programskih rešitev. Sestavljen je s 6 faz razvojnega procesa, ki se med seboj ne prekrivajo. Verifikacija in validacija predstavlja zaključek posamezne faze in hkrati vhodno točko naslednje faze. Grafično je model predstavljen na sliki 3-5:



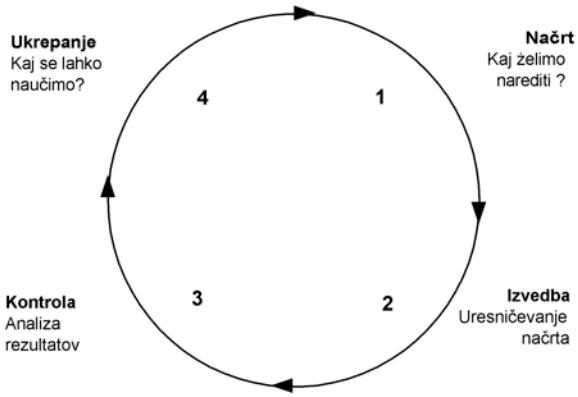
Slika 3-5: Faze kaskadnega modela.

Model se dobro obnese pri projektih, kjer so programske zahteve zelo stabilne in dobro znane že na začetku projekta. Koncept zmanjšuje količino odvečnega planiranja, saj je kompleten projektni plan narejen že ob začetku življenjskega cikla. Ker so faze med seboj ostro ločene, lahko v različnih fazah projekta sodelujejo različni posamezniki. Pogoj za prehod ene faze v drugo je dobro dokumentirana in zaključena prejšnja faza.
Waterfall model torej temelji na dobro znanih zahtevah, definiranih že ob začetku projekta. Pri večini projektov razvoja programskih rešitev je poznavanje zahtev v zgodnji fazi projekta majhno (glej 3.1.2). Veliko zahtev se ugotovi v fazi načrtovanja in celo v fazi kodirnanja. Pri takšnih projektih se kaskadni model težko obnese, saj so kršene njegove temeljne predpostavke.
Iz naštetega je razvidno, da je kaskadni model bolj primeren za produkcijo kot pa dejanski razvoj programskih rešitev. V praksi se uporablja predvsem pri vzdrževanju stabilnih programskih rešitev. Te so podvržene manjšim spremembam, ki so dobro definirane že pred načrtovanjem in implementacijo. Togost Waterfall modela skuša odpraviti naslednji predstavljeni model: evolucijski model.

3.2.2 Evolucijski model

Zgodnje začetke evolucijskega modela je postavil Mills (1971), pravo obliko pa je teorija dobila z Gilbom leta 1988. Bistvo evolucijskega modela je, da uporablja cikle. Vsak cikel sledi kaskadnemu modelu (3.2.1.) in se ukvarja samo z malim segmentom projekta, tako da cikli hitro sledijo eden drugemu.
V začetku razvoja je veliko neznank, ki jih moramo raziskati in spremeniti v znane zahteve. To počnemo iterativno, tako da se vsak cikel ukvarja s posameznim problemom. S povratnimi informacijami pridobljenimi v vsakem ciklu, spoznavamo obravnavano

problematiko in izboljšujemo učinkovitost izvedbe. Glede na ugotovljene delne rezultate lahko tokom projekta, popravljamo napačne predpostavke in povečamo možnost končnega uspeha projekta. Vplivanju delnih rezultatov na nadaljnjo izvedbo pravimo tudi Demingov krog, ki je prikazan na sliki 3-6:



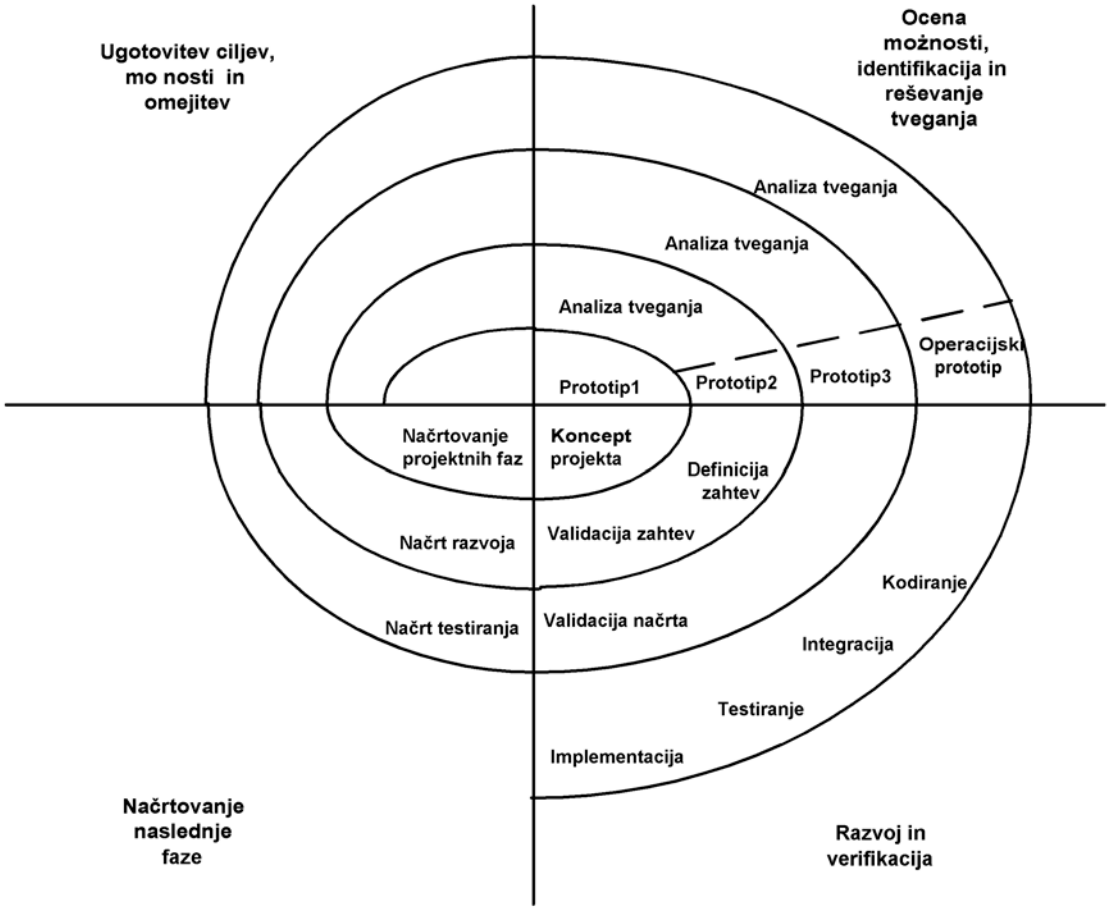
Slika 3-6: Demingov krog.

V evolucijskem razvojnem modelu v zgodnjih fazi projekta zadošča grob opis problema s strani naročnika, zato je možem zelo hiter start projekta. Prav tako se izognemo možnosti, da bi projekt preveč zašel iz želene smeri, saj naročniku nenehno predstavljamo delne rezultate.

Glavna pomanjkljivost modela je, da zaradi nenehnih popravkov, ne določa kako dolgo bo trajal projekt in kakšni bodo njegovi stroški. Zahteva veliko prisotnost naročnika med izvajanjem projekta, saj naročnik pogosto preverja vmesne rezultate in določa potrebne popravke. Subjektivne naročnikove pripombe lahko pripeljejo tudi do težav pri napredovanju projekta. Ponavadi je naročnikovo tehnično znanje precej omejeno, tako da je potrebno tudi precejšnje sodelovaje tehničnih strokovnjakov s strani izvajalca.

3.2.3 Spiralni razvojni model

Spiralni razvojni model temelji na identifikaciji in obvladovanju tveganja na projektu. Model, katerega avtor je Boehm (1988), združuje prednosti drugih razvojnih modelov in odpravlja njihove slabosti. Grafično je model predstavljen na sliki 3-7:



Slika 3-7: Spiralni razvojni model.

Slika 3-7 je ključna za razumevanje spiralnega razvojnega modela. Radialna komponenta spirale predstavlja kumulativne stroške projekta od začetka do trenutnega stanja na projektu. Kot spirale določa napredovanje posameznega cikla spirale. Model predpisuje za vsak cikel spirale iste korake (cikel se začne na abscisni osi in se konča, ko spirala zaobjame kot 360°).

Vsak cikel spirale se začne z ugotavljanjem:

- **ciljev**: izražajo kaj naj posamezna faza doseže (npr. funkcionalnost, zanesljivost rešitve),
- **različnih možnosti**: različne poti za doseganje zastavljenih ciljev (npr. uporaba različnih programskih jezikov, razvoj nove rešitve ali uporaba že razvite programske rešitve),
- **omejitev**: količine, ki omejujejo izvedbo faze (npr. roki, stroški, programski vmesniki).

Naslednji korak je ocena ugotovljenih možnosti, glede na zastavljene cilje in omejitve. Bistvo koraka je ugotavljanje negotovosti, ki predstavljajo tveganje pri projektu. Ugotovljena tveganja je potrebno formulirati in pripraviti stroškovni prerez strategij za njihovo odpravo. V tem koraku je tipična izdelava prototipov, simulacij, vprašalnikov za uporabnike rešitve in drugih tehnik, z namenom zmanjšanja tveganja na projektu.

Ugotovljena tveganja in njihova ocena določajo nadaljnje projektne aktivnosti. Dokler je ocena tveganja prevelika ponavljamo tehnike, ki nam omogočijo odpraviti neznanke in

tako zmanjšati tveganje. Zmanjševanje tveganja in sledenje spirali nas pripelje do validiranih zahtev, načrta razvoja in overjenega načrta testiranja. Zadnji verziji prototipa pravimo operacijski prototip. Operacijski prototip je prototip, ki je zadovoljil večino uporabniških zahtev in programske logike in naj ne bi bil deležen bistvenih sprememb. Operacijskemu prototipu sledijo koraki iz klasičnega kaskadnega modela: kodiranje, integracija, testiranje in implementacija. Koraki so možni, ker so predhodni spiralni cikli zagotovili ustrezne specifikacije. Seveda je namesto klasičnega waterfall modela, možno uporabiti tudi kakšen drug razvojni model, kar priča o fleksibilnosti spiralnega modela.
Bistvena prednost spiralnega modela je, da izkorišča najboljše lastnosti obstoječih razvojnih modelov, hkrati pa odpravlja njihove pomanjkljivosti z ugotavljanjem in obvladovanjem tveganja. V posebnih okoliščinah (npr. dobro definirane specifikacije že v začetni fazi projekta), se lahko spiralni model prelevi v katerega od klasičnih razvojnih modelov (npr. kaskadni).
Poleg naštetega ima spiralni model še naslednje dodatne prednosti:

- določa strategijo razvoja programske opreme v podjetju in omogoča ponovno uporabo programske kode,
- podpira razvoj produktnih programskih rešitev in obvladovanje sprememb za specifične naročnike,
- osredotoči se na zgodnje odpravljanje napak in nezanimivih možnosti,
- omogoča iteracije, vračanje v prejšnje faze in predhodno zaključevanje neuspešnih projektov,
- ponuja isti razvojni proces za razvoj novih rešitev in nadgradnjo obstoječih rešitev,
- nudi podporo razvoju strojne opreme.

Spiralni razvojni model se srečuje tudi z nekaterimi težavami:

- model se zelo dobro obnese pri internem razvoju, manj pa pri razvoju programskih rešitev za zunanjega naročnika. Interni razvoj ima več svobode pri prilagajanju projektnih faz , preoblikovanju rokov in stroškov, medtem ko pogodbeni odnos to fleksibilnost močno omejuje.
- premalo tehničnega vpliva v fazi izdelave specifikacij. Spiralni model je zelo učinkovit pri ugotavljanju in odpravi tveganj pri projektu. Obstaja nevarnost, da je v fazi odpravljanja tveganja prisotno premalo strokovnjakov s tehničnega področja. To lahko pripelje do odličnih specifikacij, ki pa so tehnično zelo težko izvedljive.
- potrebno je dodatno usklajevanje, da vsi udeleženci projekta delujejo v isti smeri. Ker se istočasno več ljudi ukvarja z identifikacijo in odpravo tveganja, je potrebno zagotoviti, da posamezniki medsebojno sodelujejo in vedno delujejo konsistentno.

Kljub naštetim težavam lahko zaključimo, da je spiralni razvojni model bolj prilagodljiv od večine drugih razvojnih modelov. V praksi se je spiralni model pokazal kot zelo uspešnega pri razvoju zahtevnih programskih rešitev v kratkem času, saj omogoča fleksibilno prilagajanje zahtev dinamičnega razvojnega okolja.

3.2.4 Razvojni model RUP

Za začetek je potrebno opredeliti razliko med pojmoma Unified Process in Rational Uinified Process, ki se v praksi vse prevečkrat spregleda. Poenoteni proces (Unified Process) je proces razvoja programske opreme, ki združuje najboljše primere z industrijske prakse. Model je zgrajen na podlagi izkušenj z različnih gospodarskih panog in okolij. Proces opisuje kako učinkovito uporabiti komercialno preizkušene metode razvoja programske opreme. Izraz poenoten v imenu procesa poudarja, da so v procesu združeni najboljši primeri in procesi več deset letnega dela. Pri oblikovanju procesa je sodeloval širok krog ekspertov, celotni proces pa je v knjigi združil Jacobson et. al (1999).

Kadar omenjamo pred imenom procesa še besedo Rational, s tem mislimo na programsko orodje, ki omogoča implementacijo poenotenega procesa. Orodje vključuje opise različnih delovnih tokov in faz procesa, cilje posameznih faz, odgovorne osebe za njihovo izvedbo, predloge razvoja in podobno. Rešitev podpira spletni dostop uporabnikov, ponuja bazo znanja in temelji na jeziku UML (Unified Modeling Language), ki definira standarde opisa modela programske rešitve. Orodje je produkt podjetja Rational Software Corporation in je tako tesno povezano s samim poenotenim procesom, da v praksi večinoma je razlikujemo med orodjem in modelom.

Osnova modela je enostavna in dobro definirana arhitektura procesa. Seveda predstavlja arhitektura procesa samo okvir, znotraj katerega je proces prilagodljiv glede na potrebe posameznega podjetja. Pri definiciji zahtev proces temelji na dejstvu, da želimo s programsko rešitvijo zadovoljiti potrebe končnih uporabnikov. Tako vsak del programskih zahtev opisuje interakcijo med uporabnikom in rešitvijo. Interakcijo definira z zaporedjem aktivnosti potrebnih, da rešitev vrne uporabniku želeni rezultat. Zbir vseh interakcij definira želeno funkcionalnost sistema. S takšnim zbiranjem zahtev povečamo verjetnost, da bo končni sistem res zadovoljil potrebe uporabnika. Interakcija s končnim uporabnikom je definirana z diagrami poteka, podatkovnimi modeli in scenariji. Ti dokumenti ne predstavljajo samo programskih specifikacij, temveč tudi osnovo za celotni proces razvoja programske rešitve. Diagrami se med razvojem tudi ustrezno dopolnjujejo, po predaji v uporabo pa so osnova za nadaljnje vzdrževanje rešitve.

Razvoj programske opreme po metodi RUP temelji na iteracijah. Ker je pri razvoju kompleksnih programskih rešitev praktično nemogoče zajeti vse cilje in izvesti ves razvoj v enem koraku, razbije model delo v tako imenovane "mini projekte" (podobno kot evolucijski model, glej 2.3.2). Vsak "mini projekt" se imenuje iteracija. Za uspešen projekt morajo biti iteracije nadzorovane, načrtovane in izvajane v pravilnem vrstnem redu. Vsebino in vrstni red interakcij določata dva faktorja. Vsebino posameznih iteracij določajo definirane zahteva, torej diagrami poteka, delovni tokovi in scenariji. Vrstni red izvajanja iteracij pa je odvisen od ocenjenega tveganja posameznih iteracij. Bolj tvegane iteracije so izvedene pred bolj rutinskimi.

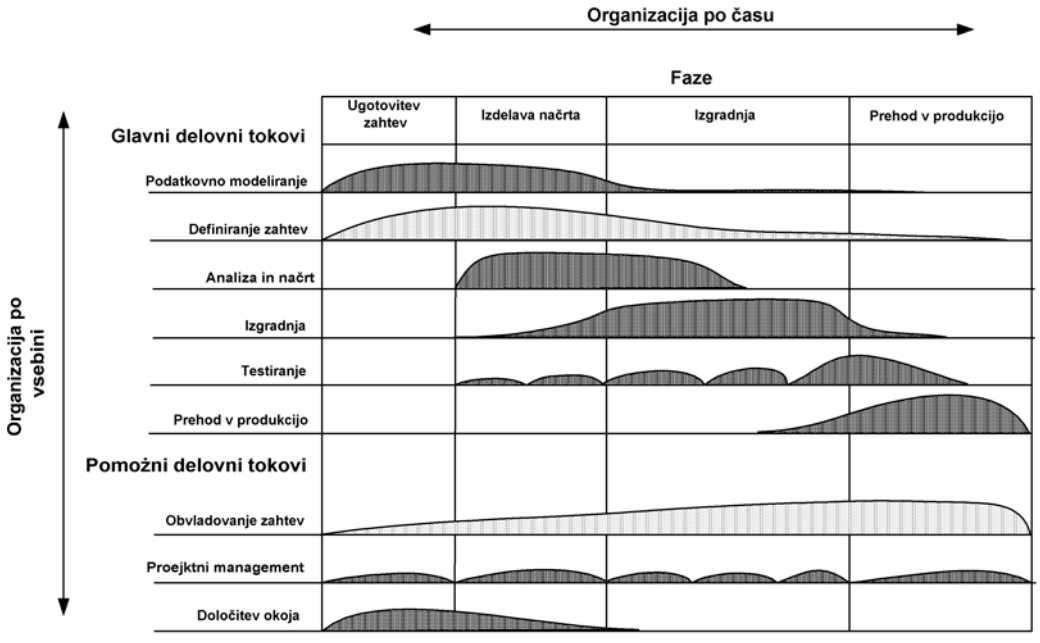
Pri RUP je proces razvoja programske rešitve razbit v cikle. Vsak razvojni cikel se ukvarja z novo generacijo produkta in je razbit v štiri zaporedne faze: ugotovitev zahtev, izdelava

načrta, izgradnja in prehod v produkcijo. Namen in rezultati posamezne faze v procesu RUP so prikazani v tabeli 3-3:

Faza	Namen	Rezultati
Ugotovitev zahtev	Prodaja projekta naročniku	Odločitev o izvedbi projekta Razvojni prototipi Ekonomika projekta
Izdelava načrta	Obvladovanje tveganja na projektu	Ugotovitev kritične poti Določitev orodij uporabljenih pri projektu Vzpostavitev razvojne arhitekture
Izgradnja	Razvoj programske rešitve	Programska rešitev z vsemi zahtevanimi specifikacijami
Prehod v produkcijo	Vpeljava rešitve v produkcijsko okolje	Rešitev predana v uporabo končnim uporabnikom

Tabela 3-3: Faze razvojnega modela RUP.

Vsaka faza se konča s kontrolno točko, s katero pregledamo ali so bili doseženi načrtovani rezultati faze in ali je možen zaključek trenutne faze in prehod v naslednjo fazo. Vsaka od faz je razdeljena na več iteracij, vsaka od iteracij pa gre skozi različne delovne tokove. Proces RUP predvideva šest različnih delovnih tokov: podatkovno modeliranje, definiranje zahtev, analiza in načrt, izgradnja, testiranje in prehod v produkcijo. Našteti delovni tokovi se pojavljajo v vseh štirih fazah. Proces RUP je torej razdeljen na dve dimenziji: faze in delovne tokove. Obe dimenziji sta prikazani na sliki 3-8, kjer vertikalna os predstavlja časovno komponento projekta, horizontalna pa aktivnosti glede vrsto dela in delovne tokove. Ploščina pod posamezno krivuljo predstavlja količino dela v posameznem delovnem toku znotraj posamezne faze projekta.



Slika 3-8: Razvojni model RUP.

Glavna prednost modela RUP je, da je razvit na osnovi izkušenj iz prakse in združuje znanje z mnogih uspešnih, že izvedenih informacijskih projektov. Model je razvit po meri principov razvoja programske opreme: upošteva iterativni razvoj, temelji na uporabniških zahtevah in postavlja v ospredje arhitekturo sistema. Model ima točno določene kontrolne točke, v katerih se jasno izražamo o nadaljevanju oziroma prekinitvi projekta. Velika prednost modela je tudi močna podpora orodja Rational, ki podpira jezil UML in omogoča dostopnost modelov vsem udeleženim na projektu.
Kljub vsemu ima model RUP tudi nekaj potencialnih slabosti. Model eksplicitno ne ponuja podpore multiprojektnemu okolju, modeliranju na nivoju podjetja in ponovni uporabi programskih modulov. Potencialno možnost neuspeha lahko predstavlja tudi iterativni razvojni pristop. Iteracije ne smejo biti zastavljene preveč ambiciozno, saj lahko izvajalci v tem primeru izgubijo pregled nad projektnimi prioritetami in izgubijo motiviranost zaradi nepreglednih rezultatov iteracije. Pogoj za uspešno implementacijo modela je tudi, da so zaposleni dobro seznanjeni z orodjem Rational.

3.3 Obvladovanje uporabniških zahtev

Ena najbolj zanesljivih "metod" zagotavljanja problemov pri razvoju kompleksnih programskih rešitev so slabo dokumentirane uporabniške zahteve. Zahteve predstavljajo podroben opis funkcionalnosti in lastnostni programske rešitve. Dobre specifikacije so jasne, natančne in zagotavljajo metode za njihovo vzdrževanje in testiranje. Pomembno je, da v proces definiranja programskih zahtev vključimo čim več posameznikov, ki igrajo pri projektu pomembno vlogo. pri definiranju programskih zahtev naj tako nastopajo končni uporabniki, preizkuševalci programske rešitve, prodajniki, skratka vsi, katerih želje želimo uresničiti z razvojem programske rešitve.
V prejšnjih poglavjih (glej 3.1.) smo že ugotovili, da se zahteve spreminjajo tokom življenjskega cikla projekta. Malotaux (2001) je problematiko obvladovanja programskih zahtev združil v dva paradoksa specifikacij. Prvi paradoks programskih zahtev pravi:

- Uspešna programska rešitev potrebuje stabilne programske zahteve.
- Programske zahteve se nenehno spreminjajo.

Kljub temu, da smo vložili vse moči za izdelavo popolnih in stabilnih zahtev, se bodo te tokom projekta spreminjale. Spremembe niso samo posledica tega, da se spremenijo naročnikove želje, ko vidijo delne rezultate razvojnega procesa. Do sprememb specifikacij pride tudi zaradi tega, ker dobijo med razvojnih procesom razvijalci programske opreme nov pogled na tematiko in na to kako naj bi izgledale specifikacije že na začetku. spremeljive specifikacije so torej znano tveganje, ki ga je potrebno obvladovati pri razvoju programskih rešitev.
Drugi paradoks programskih zahtev (Malotaux, 2001) pa pravi:

- Ne želimo si spreminjati programskih zahtev.
- Ker vemo, da predstavljajo spremembe specifikacij znano tveganje, jih poskušamo vzbuditi čim prej je to mogoče.

Drugi paradoks opozarja, da je pri spremembah zahtev potreben progresiven pristop. Kasneje, ko pri projektu pride do spremembe specifikacij, težje jih bo obvladovati in večji bodo stroški sprememb.

Obvladovanje sprememb pri projektih razvoja programskih rešitev je del širše problematike, ki jo obravnava konfiguracijski management (SCM - Software Configuration Managment). Poleg obvladovanja zahtev, se konfiguracijski management ukvarja še z obvladovanjem sprememb programske kode, dokumentiranjem procesov, zaznavanjem problemov in obvladovanjem sprememb načrta programske rešitve.

Osnovne funkcije konfiguracijskega managementa sta dobro definirala Wingerd in Seiwald (1998). V članku sta avtorja združila pogleda kot bivša razvijalca programskih rešitev in kot zdajšnja ponudnika orodij za podporo konfiguracijskemu managementu. Glavna področja delovanja konfiguracijskega managementa sta razdelila v šest skupin:

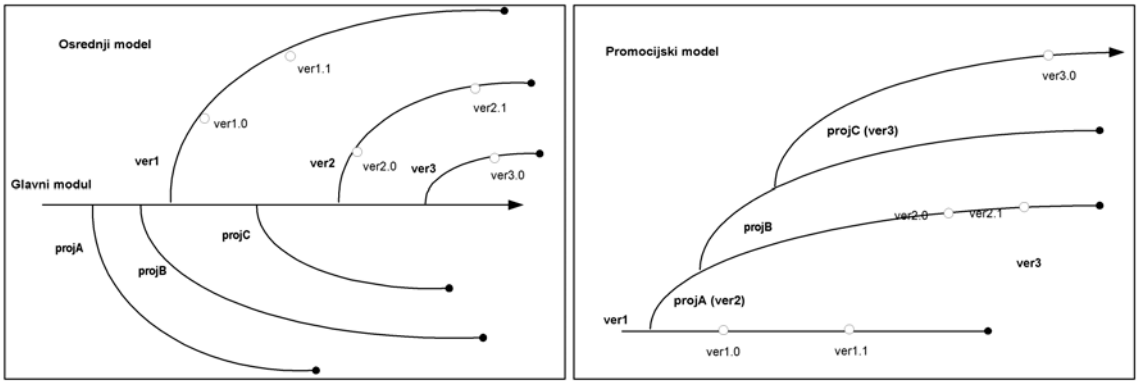
- razvojno okolje,
- programska koda,
- obvladovanje različic programske kode,
- verziranje,
- izgradnja rešitve (pretvorba izvorne kode v programsko rešitev),
- obvladovanje zgoraj naštetih področij.

Razvojno okolje je okolje, kjer razvijalci rešitev popravljajo izvorno kodo, gradijo komponente programske rešitve, testirajo in prevajajo izvorno kodo. Vsako razvojno okolje mora imeti definiran namen. Z drugimi besedami, razvojno okolje mora biti jasno ločeno od testnega, to pa od produkcijskega. Če isto okolje služi za razvoj, test in produkcijo to povzroča zmedo pri uporabnikih in povečuje možnost napak (npr. testiranje na napačni verziji programske rešitve). Delo razvijalcev in preizkuševalcev mora v celoti potekati znotraj definiranega okolja, saj je samo znotraj definiranega okolja možno slediti spremembam in spremljati razvoj programske rešitve. V razvojnih okoljih ponavadi razvija isto prograsmko rešitev več razvijalcev hkrati. V takšnih primerih morajo razvijalci paziti, da uskladijo spremembe, ki so jih naredili ostali. Sprememba posameznega modula pomeni, da ga morajo osvežiti tudi ostali koderji, tako da ostanejo v stiku s tekočo verzijo rešitve. Dobro je, da so spremembe modulov objavljajo takoj, ko so ti popravljeni, saj bo to pospešilo celotni razvojni proces.

Programska koda je skupek izvorne kode, ki se nahaja v različnih izvornih datotekah. Izvorna koda in uporabljeno programsko orodje skupaj omogočata izgradnjo programske rešitve. Vsak sklop izvorne kode mora imeti določen predpis za obvladovanje sprememb. Predpis določa kdo lahko popravlja posamezen sklop. Ob začetku popravljanja programer zaklene popravljani sklop (check-out), ki ga do nadaljnjega ostali ne morejo popravljati. To preprečuje, da bi dva različna programerja sočasno popravljala isti del kode. Ko razvijalec konča s spremembami kode, sledi pregled in testiranje sprememb. Po uspešnem testiranju sledi objava kode (check-in). Spremenjena koda je tako dostopna v uporabo ostalim in spet na voljo za popravljanje pooblaščenim programerjem. Predpis spreminjanja kode je ključna komponenta za dokumentiran in nadzorovan proces razvoja programske

rešitve. S stališča konfiguracijskega managementa, koda brez določenega predpisa za obvladovanje sprememb ni sprejemljiva in je izven nadzora. Predpis določa vsakemu sklopu izvorne kode tudi skrbnika, ki usmerja razvijalce. Skrbnik ima pregled nad celotno rešitvijo, tako da razume kako spremembe posameznih programskih modulov vplivajo na celotno rešitev.

Pri razvoju programske rešitve se programski moduli nenehno spreminjajo, zato je ključnega pomena obvladovanje različic programske kode in spremljaje verzij.Spremljanje verzij pomeni, da določen programski modul opremimo s številko verzije, ponavadi pa mu dodamo še številko izdaje. Nova verzija modula predstavlja bistvene spremembe v funkcionalnosti modula, številka izdaje pa sledi manjše spremembe in odpravljene programske hrošče. Pri evoluciji in spremljanju verzij programskih modulov ločimo dva glavna modela: osrednji (mainline) in promocijski (promotion) model. Grafično sta modela prikazana na sliki 3-9:



Slika 3-9: Osrednji in promocijski model

Pri osrednjem razvojnem modelu izhajajo spremembe programske kode iz osrednjega, glavnega modula. Tako odpravljanje napak, kot razvoj novih funkcionalnosti izvirata iz iste verzije programskega modula. Na sliki 3-9 so veje razvoja novih funkcionalnosti prikazane pod glavno vejo, veje razvoja novih verzij pa nad njo. Razvijalcem je namenjeno delo na spodnjih vejah, medtem ko so veje verzij namenjene testiranju in odpravljanju napak. Bistveno je, da se spremembe na obeh vejah stekajo nazaj v osnovni modul.

Promocijski model predstavlja drugačen pristop, ki ne ločuje med vejami razvoja novih funkcionalnosti in vejami verzij. S slike 3-9 je razvidno, da lahko postane razvojna veja hkrati tudi veja verzije in osnova za nadaljnji razvoj in naslednjo verzijo programskega modula. Vsaka nova verzija nastane iz razvojne verzije in ne iz osnovne, kot pri osrednjem modelu.

Promocijski model se srečuje z dvema hibama. Opredeliti je potrebno kdaj postane razvojna različica tudi nova verzija in kdaj se nova veja odcepi od trenutne. Merila za prehod so težko jasno določljiva. Druga težava nastopi, če se mora programer prestaviti iz ene razvojne veje na drugo. To zahteva usklajevanje obeh vej, kopiranje kode, je časovno

zahtevno in je lahko razlog za napake. Žal v praksi 90% vseh razvojnih procesov podpira promocijski model. Z uporabo osrednjega modela je razvojni proces usmerjen in poenostavljen, programska rešitev pa hitreje doseže želeno stabilnost.
Ne glede na to, kateri model uporabljamo, ostaja glavni problem pri spreminjanju programske kode ustvarjanje novih vej iz trenutno odklenjenih vej. Takšnemu vejenju kode se izognemo, če je le mogoče in ga uporabimo v res izjemnih primerih. Eden takšnih je, če imata nad isto kodo dve skupini različna predpisa. Testna skupina objavlja kodo (check-in) po temeljitem testu, medtem ko razvijalci objavljajo po manj strogem testiranju. Če skupini delata z isto kodo, to pomeni, da bodo razvijalci kodo že objavili, tesna skupina pa še ne. Vsaka takšna veja potrebuje posebno pozornost, saj lahko kasnejša verzija prepiše funkcionalnosti prejšnje.
Končni korak razvoja programske opreme je izgradnja programske rešitve. Edina vira, ki jih potrebujemo pri izgradnji končne verzije rešitve morata biti izvorna koda in razvojno orodje. Proces izgradnje rešitve mora biti enoličen, kar pomeni, da vedno vrne isto programsko rešitev. Pri izgradnji je potrebno dokumentirati katera orodja so bila uporabljena in določiti zaporedje korakov potrebnih pri izgradnji. Če potrebuje izgradnja tudi sodelovanje s končnim uporabnikom, posebej dokumentiramo različne možnosti, ki jih lahko uporabnik izbere.
Pri izgradnji je pametno ločiti področje, kjer imamo razvojno kodo, od področja, kjer shranjujemo zgrajene module. Za odpravo kasnejših napak je zelo pomembno, da vsi udeleženci projekta uporabljajo isto verzijo razvojnega orodja. Samo tako, bodo lahko razvijalci in preizkuševalci primerjali zgrajene končne verzije. Pri izgradnji posameznih modulov je pomembno beleženje zgodovine izgradnje. Ta zgodovina bo v veliko pomoč pri nadaljnjih izgradnjah modulov, saj bo opozorila na morebitne težave pri izgradnji in integraciji.
Proces konfiguracijskega managementa je kompleksen proces, ki vključuje načrtovanje, nadzor in implementacijo. Seveda je proces odvisen od posebnosti razvojnega okolja, kljub temu pa lahko omenimo nekaj generalnih usmeritev, ki veljajo za proces. Proces mora zajeti spremembe celotnih programskih paketov. Spremljanje sprememb za posamezne module je potrebno, a ne zadostno. Če želimo spremeniti verzijo programskega paketa, moramo vedeti, kateri moduli so doživeli spremembo. Spremljanje sprememb za celotni paket, nam tudi omogoča lažje odpravljanje napak in distribucijo novega programskega paketa.
Prav tako mora proces konfiguracijskega managementa zagotoviti, da ima vsak predpis, dokument, produkt, komponenta in izvorna koda svojega lastnika. Skrbniki zagotavljajo, da bo posamezni modul živel, se razvijal in dozorel. Moduli brez skrbnika, bodo naleteli na veliko preglavic, prepuščeni bodo sami sebi in zato obsojeni na neuspeh. Proces konfiguracijskega managementa mora zagotoviti, da je dokumentacija na procesu vedno ažurna in dostopna vsem udeležencev v procesu, tako na njihovem delovnem mestu, kot na domačem računalniku. Sistem dokumentiranja procesa mora zagotoviti ažurno spreminjanje dokumentov in zgodovino prejšnjih verzij dokumentov.

3.4 Testiranje in zagotavljanje kvalitete

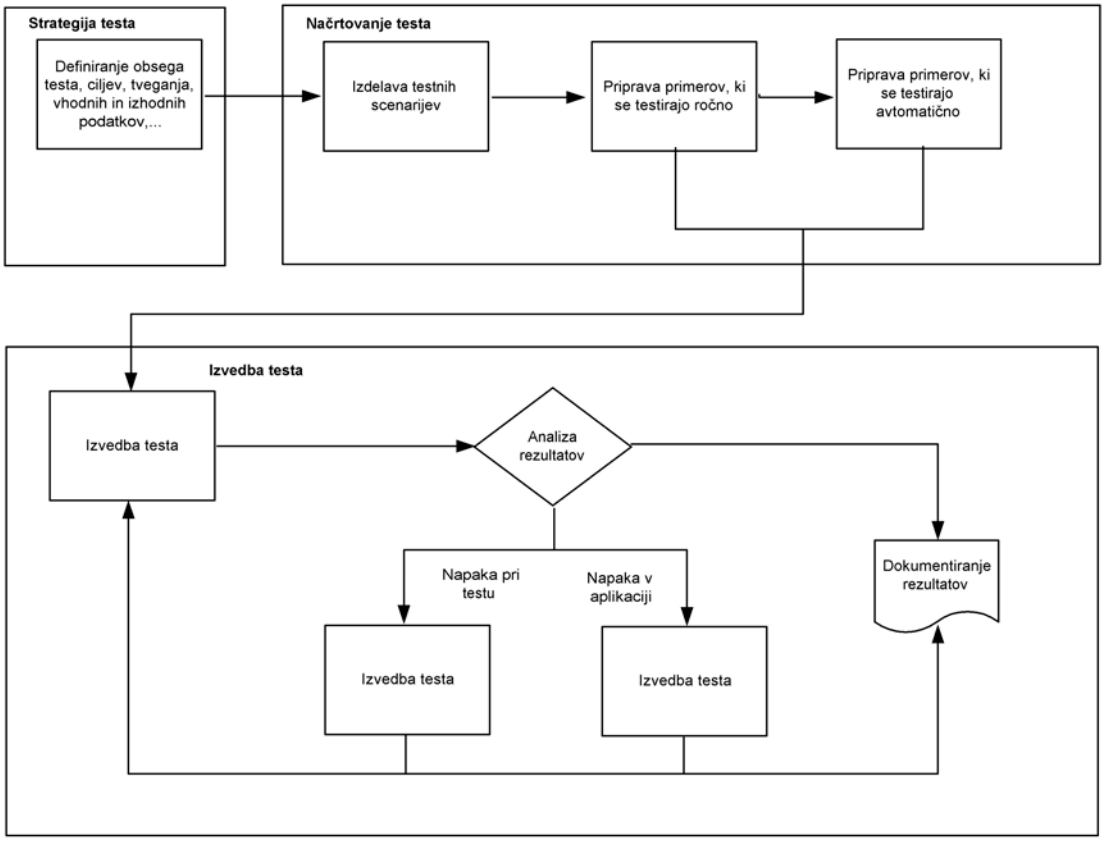
Testiranje programskih rešitev je namenjeno preventivi in odkrivanju napak. Preventiva pomeni spremljanje in izboljševanje procesa razvoja rešitve. Odkrivanje napak pa pomeni, da se rešitev razvija v kontroliranem okolju in da so rezultati razvoja ovrednoteni.
Pri procesu testiranja programskih rešitev nastopajo posamezniki z različnimi vlogami. Vodja testne skupine koordinira testne aktivnosti, obvešča vodjo projekta o trenutnem stanju testiranja in vodi testno skupino. Odgovorni za tehnično plat testa zagotovi za ustrezno strojno opremo in instalira rešitev na vse testne računalnike. Pomembno vlogo pri testu predstavlja vsebinski vodja testa, ki določa kateri podatki se testirajo in v kakšnih okoliščinah. Ne nazadnje nastopajo pri testu še sami preizkuševalci, ljudje ki dejansko opravijo test.
Testiranje programske opreme se izvaja na različnih nivojih. Po končanju testiranja na posameznem nivoju je potrebno rezultate dokumentirati in jih uskladiti s procesom konfiguracijskega managementa (glej 3.3). Testiranje na vsakem nivoju lahko opredelimo kot testiranje "črne škatle" ali testiranje "bele škatle". Testiranje črne škatle ne zahteva nobenega poznavanja programske logike in izvorne kode. Testirajo se samo zahteve in funkcionalnosti modula. Test "bele škatle" pa je test interne logike programa, izvorne kode, pogojev in sintakse.
V članku z leta 2000 opredeli Pointe Tehnology Group naslednje testne nivoje:

- **testiranje modula**: je prvi nivo testiranja in je večinoma v domeni razvijalcev, ne pa testne skupine. Testiranje modula se opravi, ko modul doseže pričakovane rezultate.
- **testiranje funkcionalnosti**: je testiranje "črne škatle", kjer se preverjajo funkcionalnosti posameznih modulov. Test mora opraviti testna skupina, ne pa razvijalci programske opreme.
- **test uporabnosti**: pri tem testu je poudarek na testiranju prijaznosti do končnega uporabnika. Seveda je to zelo subjektivna tema, odvisna od končnega uporabnika oziroma naročnika. Pri testiranju lahko uporabimo intervjuje, prototipe zaslonskih mask in druge tehnike, dobro pa je, če so pri testu prisotni končni uporabniki rešitve.
- **integracijski test**: se lahko začne, ko so končana testiranja vseh posameznih modulov. Pri integracijskem testu gre za preverjanje kako se posamezni moduli integrirajo v celoto, opravi pa ga testna skupina. Integracijski test se konča s primerjavo pričakovanih in dejanskih rezultatov, glede na različne vnosne podatke končnih uporabnikov.
- **sistemsko testiranje**: sledi končanemu integracijskemu testu. Med sistemskim testiranjem je celotni sistem nastavljen tako, da simulira dejansko produkcijsko okolje. Testirajo se vse funkcionalnosti sistema, ki bodo zahtevane v dejanskem uporabniškem okolju. Test vključuje povezovanje rešitve do relacijskih baz, uporabo mrežnih komunikacij ter povezovanje z ostalimi viri podatkov in strojno opremo. Pred začetkom testiranja se najprej pregledajo rezultati testiranja modulov in integracijskega testa. Za uspešno opravljen sistemski test je potrebno dobro poznavanje problemov, ki so bili ugotovljeni pri testiranju posameznih modulov. S stališča modularnosti je test na

najvišjem nivoju, saj vključuje vse komponente programske rešitve. Test opravi testna skupina, testiranje pa poteka kot testiranje "črne škatle".

- **performančno testiranje**: je ponavadi del sistemskega testiranja, lahko pa se opravi tudi kot samostojen test. Performančno testiranje obravnava odzivne čase, trajanje obdelav, velikost podatkov in jih primerja s programskimi zahtevami.
- **obremenitveni test**: preizkuša programsko rešitev v primeru velike obremenitve. Pri spletnih rešitvah to pomeni veliko število istočasnih dostopov do spletnega strežnika, pri transakcijskih obdelavah pa veliko število transakcij v majhni časovni enoti. Test meri kako velika obremenitev vpliva na odzivne čase končnih uporabnikov in ugotavlja ali lahko prevelika obremenitev pomeni tudi prenehanje delovanja sistema.
- **namestitveni test**: je test namestitve programske opreme. Potrebno je preveriti celotno namestitev, namestitev posameznih modulov oziroma nadgradnjo programske opreme. Podatki zbrani med testom so pomembni za izdelavo navodil za instalacijo programske opreme. Po namestitvi rešitve je potrebno opraviti še test ključih funkcionalnosti, da se prepričamo, če rešitev na uporabnikovi strani resnično deluje.
- **testiranje varnosti**: meri odziv rešitve na različne neavtorizirane poskuse pristopa. Test postaja še posebej pomemben v času spletnih programskih rešitev. Zaradi vedno novih metod vdora v informacijske sisteme je test zelo težaven in se mora izvajati periodično.
- **test sprejemljivosti**: je podoben sistemskemu testu s to razliko, da se izvaja s strani naročnika oziroma končnega uporabnika. S testom naročnik še enkrat preveri funkcionalnosti rešitve, preden gre ta v produkcijo. Kljub temu, da test izvaja naročnik rešitve, mora biti test voden s strani projektne skupine. Samo tako lahko dobimo merljive rezultate, primerljive s prejšnjimi testi.
- **alfa test**: je testiranje rešitve, ko je razvoj praktično zaključen. Testiranje izvajajo končni uporabniki, glede na njihove ugotovitve, pa so možne še manjše spremembe programske rešitve.
- **beta test:** je testiranje rešitve, ko je razvoj programske rešitve zaključen. Namen testa je odkriti še zadnje napake pred končno verzijo rešitve in preden gre programska rešitev v produkcijsko uporabo. Tako kot alfa test, tudi beta test izvajajo končni uporabniki.

Testiranje rešitve torej poteka na različnih nivojih, v različnem obsegu in s strani različnih uporabnikov. Za koordinacijo vseh naštetih vrst testov je potreben dobro definiran proces testiranja. Po analizi Pointe Tehnology Group (2000) se proces testiranja sestoji s treh korakov: strategija testa, načrtovanje testa in izvedba testa. Medsebojna odvisnost korakov je grafično prikazana na sliki 3-10:



Slika 3-10: Proces testiranja.

Proces testiranja se začne z določanjem strategije testa. Strategija testa je formalni opis kako bo programska rešitev preizkušena. S strategijo je potrebno opisati zahtevano programsko in strojno opremo za izvedo testa, testno metodologijo, vloge in odgovornosti članov testne skupine, funkcionalne in tehnične zahteve programske rešitve. Testna strategija določa okoliščine in pogoje testa, seznam testnih aktivnosti in kriterije, ki določajo uspešnost testa. V tej fazi je določen tudi časovni plan testiranja, ki opredeljuje roke za izvedbo testnih aktivnosti in posameznike, ki bodo testirali rešitev. Faza se konča s pregledom in odobritvijo strategije testiranja, ki predstavlja vhodni dokument za fazo načrtovanja testiranja.

V fazi načrtovanja testa je, poleg potrjene testne strategije, potrebno tudi razumevanje kompleksnosti programske rešitve in povezanosti posameznih programskih modulov. Bistvo faze je izgradnja testnih scenarijev. Scenarij je zaporedje korakov, ki sledi funkcionalnosti rešitve in določa testne pogoje, podatke uporabljene za test in določa želene rezultate (popravke v podatkovnih bazah, strukturo izhodnih datotek, poročila). Scenariji so načrtovani tako, da zaobjamejo tipične in netipične dogodke, ki se lahko dogodijo v rešitvi. Za lažje sledenje v času izvedbe testa, se scenariji zapišejo v obliki testnih skript. Testna skripta je opis zaporednih korakov potrebnih za izvedbo enega ali več scenarijev. Izdelane testne skripte je potrebno pred izvedbo testa še enkrat uskladiti z zahtevami. Tako zagotovimo, da test ostane v okviru predpisanih zahtev za programsko rešitev. Potrjeni scenariji, določeni testni pogoji, testni podatki in strategija testa so osnova za izvedbo testa.

Zadnja faza v procesu testiranja je izvedba testa. Testne aktivnosti se izvajajo po terminskem planu testiranja. Med izvajanjem testa se testna skupina sestaja na rednih sestankih, kjer pregleduje in obravnava potek testa, status posameznih aktivnosti in usklajuje nadaljnje aktivnosti. Testiranje modulov mora natančno slediti testnim scenarijem. Vsak preizkušeni postopek se zabeležiti v dnevnik testiranja, skupaj z morebitnimi odkritimi odstopanji od pričakovanih rezultatov. Ob končanem testiranju predstavlja dnevnik testiranja rezultat testa.
Rezultati testiranja so ovrednoteni s strani članov projektnega tima, ki ugotovijo ali so rezultati testa v sladu s pričakovanji. Vsa ugotovljena odstopanja in anomalije so dokumentirani za nadaljnjo obravnavo in reševanje. Glede na vrsto ugotovljeno napake, je možno napako odpraviti že med testom ali pa jo zabeležiti kot nerešen problem, ki je sporočen razvijalcem programske opreme. Za vsako ugotovljeno napako se dodatno opredeli ali je napaka tehnične ali logične narave.
Ko je testiranje tudi formalno zaključeno, so nadaljnji koraki opredeljeni s strani konfiguracijskega managementa. Ta določa kako je z migracijo testiranih programskih modulov v nove verzije. Pomembno je poudariti, da bo preizkušena programska rešitev prestavljena v produkcijsko okolje šele po uspešno opravljenem testu končnih uporabnikov oziroma naročnika.

3.5 Programska rešitev za vodenje projektov.

Ne glede na to, za katero od številnih projektnih metodologij se podjetje odloči, najbrž ni potrebno posebej poudarjati pomena informacijske podpore sistemu vodenja projektov. Projektni pristop pomeni delo v dinamičnem okolju, izraža potrebe po visoki učinkovitosti, zahteva optimalno izrabo kadrov in hiter pogled v tekoče stanje del na projektu. Delo na projektu je skupinsko in zahteva komunikacijo ter sodelovanje vseh članov skupine. Prav tako nastaja pri projektnem delu veliko dokumentov, ki jih moramo ustrezno obvladovati. Vse našteto so izzivi, ki jih dejansko okolje predpisuje uspešnemu informacijskemu sistemu za vodenje projektov.
Obstaja več kategorij informacijskih sistemov za podporo vodenju projektov. Glavne kategorije teh orodij so orodja za planiranje in oceno dela, za spremljanje projekta, komunikacijska orodja, orodja za obvladovanje tveganje in kontrolo na projektu. Seveda obstajajo na trgu orodja, ki podpirajo več oziroma vse od naštetih kategorij.
Za podjetje je seveda ključno vprašanje, katerega od mnogih orodij na trgu izbrati. Fabian in Robinson (1999) predlagata nekaj korakov, ki jim mora podjetje slediti pri oceni različnih programskih rešitev. Predlagani koraki si sledijo v naslednjem vrstnem redu:

- razumevanje lastnega procesa vodenja projektov,
- priprava pisnih specifikacij,
- priprava modela za ocenjevanje,
- izvedba ocene in izbira ustreznega orodja.

Prvi korak poudarja, da moramo pred uvajanjem programske rešitve dobro poznati lasten način vodenja projektov. Osnova za oceno projektnega vodenja v podjetju je spremljanje

procesov v organizaciji in učenje z že končanih projektov. Najprej se osredotočimo na najbolj pomembne segmente: analizo tokov podatkov, organizacijsko strukturo in odgovornosti na projektu. Bolj kot poznamo lastne procese, bolje bomo opremljeni za uvedbo informacijske podpore. Čeprav se zdi omenjen pogoj zelo logičen in samoumeven se v praksi izkaže, da veliko implementacij programskih rešitev za vodenje projektov ne uspe, ravno zaradi preslabega poznavanja lastnega načina dela in pričakovanj, da bo orodje odpravilo tudi vsebinska vprašanja. Pred začetkom uvedbe programskega orodja se moramo zavedati, da orodje samo ne bo spremenilo načina dela, razmišljanja udeleženih pri projektu in njihove projektne kulture. Od orodja pa lahko pričakujemo večjo transparentnost podatkov in večjo zmogljivost upravljanja s projektom.

Pri izbiri ustreznega orodja je ključno vprašanje tudi, kdaj vključiti informatike iz podjetja v proces izbire projektnega orodja. Seveda je sodelovanje z notranjo informatiko nujno, saj se mora nova programska rešitev prilagoditi obstoječemu informacijskemu sistemu. Kljub vsemu moramo biti pazljivi, da ostanejo v ospredju vsebinska vprašanja. Prevelik vpliv informatike lahko pomeni izbiro orodja na osnovi platforme oziroma tehničnih lastnosti nove rešitve. Takšna izbira pa ni nujno v skladu z vsebinskimi potrebami procesa vodenja projektov v organizaciji.

Naslednji korak pri izbiri sistema za vodenje projektov je priprava pisne specifikacije. Pisna specifikacija naj podrobno opisuje posamezne programske module (npr. modul za časovno planiranje projekta). V tem koraku je sodelovanje z informatiki v podjetju že bolj pomembno, saj moramo upoštevati tudi trenutni informacijski sistem. Vsak modul opišemo z vhodnimi podatki, procesi znotraj modula in izhodnimi podatki. Specifikacije naj določajo tudi uporabniški vmesnik, načine dostopa do podatkov in poročila, ki se pričakujejo od vpeljane informacijske rešitve. V tem koraku izbire rešitve tudi opišemo vgradnjo nove programske rešitve v obstoječi sistem. Če na primer že imamo informacijski sistem za spremljanje stroškov, je potrebno opredeliti na kakšen način bomo povezali oba sistema (na primer preko številke delovnega naloga).

Preden sestavimo model za oceno različnih rešitev, moramo definirati tudi obseg, ki ga bo pokrivalo novo orodje za vodenje projektov. Definirati je potrebno približno število uporabnikov, ki bo delalo z rešitvijo, lokacije teh uporabnikov, njihovo trenutno informacijsko pismenost in predvideno dodatno izobraževanje.Fazi pisne specifikacije novega sistema sledi priprava modela za ocenjevanje različnih ponujenih rešitev.

Model za ocenjevanje bo seveda odvisen od zbranih podatkov pri izdelavi pisnih specifikacij. Model mora odražati sliko celotnega projektnega dela v podjetju, ne samo dela za specifičen projekt iz preteklosti. Ocenitveni model naj predvidi ločene ocene za posamezne module, saj to pomeni, da lahko skupno rešitev sestavlja informacijski sitem različnih ponudnikov. Prav tako takšna ocena omogoča postopno implementacijo – ni nujno potrebno implementirati vseh modulov naenkrat. Z izdelavo modela ocene informacijskega sistema za vodenje se bo še posebej ukvarjalo naslednje poglavje naloge (4.1).

Po definiranju modela za oceno sledi še konkretna ocena možnih rešitev in pregled ugotovljenih rezultatov. Pri izbiri ocenjevalne skupine je pomembno, da so zastopani predstavniki različnih organizacijskih enot, ki bodo uporabljale sistem za vodenje projektov. Ocenjevala skupina se mora odločiti:

- Ali bo vse module uvedel isti ponudnik, ali pa bodo moduli porazdeljeni na več izvajalcev?
- Ali so ponujeni module ustrezni za podjetje, ali pa bodo potrebna dodatna prilagajanja modulov?
- Kdo bo povezal različne module v primeru odločitve za več modulov različnih ponudnikov?
- Koliko časa in kakšen oseb dela bo potrebovala uvedba, glede na različne možne rešitve?

Seveda sledi izbiri najbolj ugodnega ponudnika, še implementacija izbranega informacijskega sistema. Implementacija je lahko zelo enostavna, npr. nabava programske rešitve in instalacija na en računalnik, ali pa zelo zahteven, več mesec trajajoč projekt. Takšen projekt seveda zahteva že isto obravnavo, kot ga bodo zahteva projekti, ki jih bomo spremljali z uvajano programsko rešitvijo.

Za konec poglavja o procesu razvoja programskih rešitev bi težko našli bolj primeren zaključek, kot je seznam desetih ugotovitev o metriki informacijskih projektov, ki jih je podal Boehm (1987). Ugotovitve so sicer stare že dobrih 10 let, ampak se njihova točnost in aktualnost ohranila do danes. Razmera in števila, ki jih podajajo trditve se sicer v praksi nekoliko spreminjajo, bistvene pa so ideje, ki jih je podal avtor. V nadaljevanju so ugotovitve navedene v poševni pisavi, zraven pa sledi še kratek komentar.

1. *Ugotovitev in odprava napake v programski kodi po implementaciji rešitve, stane 100 krat več kot odprava iste napake v zgodnji fazi projekta.*
   Razmerje med stroški odprave napake v času produkcije in1 v zgodnjih fazah, je tako veliko, da je osnova za vse različne modela razvoja programskih rešitev. Ugotovitev ne velja samo na informacijske projekte, temveč tudi za projekte v ostalih panogah.
2. *Čas izvdebe razvojne aktivnosti lahko skrčimo za največ 25% od načrtovanega.*
   Čas izvedbe določene projektne aktivnosti lahko zmanjšamo z dodajanjem človeških virov na to aktivnost. Pri dodajanju virov pa se seveda podaja vprašanje, kako dodajanje vpliva na čas izvedbe aktivnosti. Logično je, da se trajanje aktivnosti ne more v nedogled zmanjševati sorazmerno z dodajanjem novih ljudi. Praktični primeri so celo privedli do trditve, da lahko čas trajanja aktivnosti zmanjšamo za največ 25%. Aktivnosti, ki ima načrtovano trajanje 10 mesecev, torej ne bo končana prej kot v 7 mesecih in pol.
3. *Za vsak porabljen tolar v fazi razvoja, bomo porabili dva tolarja za vzdrževanje.*
   Boehm imenuje trditev tudi "železni zakon razvoja programskih rešitev". Trditev drži tako za razvoj in vzdrževanje produktov, kakor tudi pri razvoju programskih rešitev na ključ.

4. *Stroški razvoja in vzdrževanja programske opreme so najbolj odvisni od števila vrstic programske kode.*

   Ugotovitev je rezultat tega, da se praktično vsaka programska rešitev prilagaja naročnikovim željam, pomanjkanju univerzalnih programerskih standardov in premajhni ponovni uporabljivosti programske kode.
5. *Učinkovitost razvoja programskih rešitev je v veliki meri odvisna od zaposlenih ljudi.*

   Ugotovitev se še posebej izkaže v primerih, ko ne znate točno ugotoviti zakaj je projekt uspel ali propadel. V takšnih primerih je odgovor ponavadi v ljudeh, ki so sodelovali pri razvoju rešitve.
6. *Stroški za programsko so večji od stroškov strojne opreme. Razmerje lahko ocenimo na 85:15, se pa še vedno povečuje v prid stroškov razvoja rešitev.*

   Razlog za takšno razmerje je poleg nenehnega padanja cen strojne opreme, tudi vedno večja kompleksnost rešitev. Iz tega dejstva sledi, da je potrebno zelo smotrno sestavljati programske specifikacije, saj prevelika kompleksnost obvezni prinese večje stroške, ki pa niso nujno sorazmerno pridobljenim funkcionalnostim.
7. *Samo okrog 15% časa razvoja programske opreme naj bo posvečeno programiranju oz. kodiranju.*

   Žal je v praksi odstotek praviloma veliko večji, kar pomeni zanemarjanje vsaj enega od potrebnih spremljevalnih procesov (obvladovanja zahtev, načrtovanja, kontrole, obvladovanja sprememb testiranja).
8. *Stroški s kompleksnostjo programske rešitve ne naraščajo linearno, temveč s kvadratom kompleksnosti rešitve.*

   Posledica trditve je, da z večanjem števila programskih vrstic rastejo tudi stroški za posamezno programsko vrstico. Ugotovitev velja za razvoj rešitev, se pa ne sklada s projekti v ostalih panogah, kjer večji obseg praviloma prinese zmanjšanje stroškov na posamezno enoto.
9. *Pregledi programske rešitve odkrijejo okrog 60% napak.*

   V splošnem se pri pregledu rešitve končni uporabniki posvečajo predvsem izgledu rešitve in napakam povezanim z vnosom podatkov. Veliko manj odkritih napak je logičnih, povezanih s pravilnim delovanjem in funkcionalnostmi rešitve. Posledica ugotovitve je, da mora biti proces testiranja rešitve dobro načrtovan in mora imeti jasno definirane cilje.
10. *80% prispevka prihaja od 20% udeleženih virov (Paretov zakon).*

    Ugotovitev je nekakšno splošno pravilo, ki velja za večino tehničnih procesov. v primeru razvoja programskih rešitev, bi lahko trditev opravičili z naslednjimi primeri:
    80% načrtovanje je posledica 20% specifikacij,
    80% stroškov projekta je posledica 20% programskih modulov,
    80% napak je povezano z 20% programskih modulov,
    80% virov je zasedeno z 20% funkcionalnosti rešitve,
    80% načrtovanja je realizirano z 20% orodij,
    80% napredka je posledica dela 20% udeleženih na projektu.

4 Programske rešitve za podporo projektnemu vodenju

Kot je bilo prikazano v prejšnjem poglavju je eden ključnih dejavnikov za uspešno vodenje projektov v podjetju tudi učinkovita programska rešitev. Tak sistem omogoča članom projektne skupine enostaven in hiter dostop do projektnih podatkov, določa status projekta, nudi pregled stroškov in problemov na projektu ter povečuje komunikacijo znotraj projektnega tima. Glede na ogromno količino različnih programskih rešitev (http://www.infogoal.com/pmc/pmcswr.htm, http://www.startwright.com/project1.htm) se logično pojavlja vprašanje, katero programsko rešitev izbrati. Namen poglavja je opredeliti kriterije, ki določajo kvaliteto informacijskega sistema za vodenje projektov. Po opredelitvi kriterijev, bodo v poglavju opisane tudi tri dejanske programske rešitve: Microsoft Project 2000, e-Proj in Primavera TeamPlay. Po opisu bomo posamezne rešitve tudi ocenil s kriteriji določenimi na začetku poglavja. Primerjava vseh treh opisanih produktov bo pokazala prednosti in slabosti posameznih rešitev in posledično njihovo primernost za različna okolja.

4.1 Kriteriji za ocenjevanje

Ocenjevanje uspešnosti programskih rešitev za podporo vodenju projektov izhaja iz kriterijev, ki jih določajo znanja s področja projektnega managementa in tehnološka izvedba rešitve. Kot je bilo opisano že v prejšnjem poglavju (3.5) je glavni poudarek pri določanju kriterijev na vsebinskih vprašanjih, manjši (a ne zanemarljiv) del pa odpade na tehnično izvedbo rešitve. Kriterije za oceno programskih rešitev bomo razdelili na naslednje sklope:

- prilagodljivost procesu vodenja projektov,
- podpora časovnemu planiranju,
- spremljanje izvedbe,
- komunikacija med člani projektne skupine,
- projektna poročila in baza znanja,
- tehnična ustreznost rešitve,
- cena.

Preden se lotimo bolj podrobnega opisa posameznih kriterijev, je potrebno opisati še kako se posamezni kriteriji vrednotijo in kako se izračunana končna ocena za preizkušeno programsko orodje.

Vsak od naštetih kriterijev je razčlenjen na več podkriterijev (od 3 do 5). Vsak od podkriterjev se ocenjuje z vrednostjo med 0 in 9. Pri tem večje število pomeni boljšo oceno, temeljna pravila ocenjevanja pa so predstavljena v tabeli 4.1:

Ocena	Opis ocene
0	Ocena 0 pomeni, da ocenjevani kriterij v orodju ni implementiran.
1	Ocena 1 pomeni, da je ocenjevana funkcija prisotna v programski rešitvi, ampak je njena implementacija zelo slaba oziroma neustrezna.
5	Ocena 5 pomeni, da je ocenjevani kriterij sprejemljiv in da bi se ga dalo še

	izboljšati.
9	Ocena 9 pomeni, da je ocenjevana funkcija v programski rešitvi implementirana zelo dobro in da predstavlja najboljšo praktično implementacijo na tem področju.

Tabela 4-1: Temeljna pravila za ocenjevanje kriterijev.

Glede na pomembnost so posamezni kriteriji še ustrezno uteženi. Utež je število med 0 in 1, ki predstavlja pomembnost podkriterija glede na ostale. Zaradi normalizacije imajo uteži takšne vrednosti, da je njihova vsota enaka 1. Vsota produktov ocenjenih podkriterijev in uteži pomeni oceno kriterija. Zaradi normalizacije uteži, je tudi končna ocena kriterija med 0 in 9.

Pri računanju končne ocene programske rešitve se postopek izračuna ponovi (tokrat na enem nivoju višje). Vsak od kriterijev je utežen z utežjo, končni rezultat je vsota uteženih ocen kriterijev. Uteži kriterijev so prav tako normalizirane (njihova vsota je 1), kar pomeni, da je tudi končna ocena produkta nekje med 0 in 9.

Namen dela je podati objektivne ocene za posamezne kriterije. V delu bomo najprej primerjali posamezne rešitve glede na ocene (brez uteži) in tako dobili dejansko oceno, kateri od produktov je po izbranih kriterijih najboljši (poglavje 4.3.3.).

Seveda pri dejanski izbiri programske rešitve ključno vlogo odigrajo postavljene uteži. Uteži namreč odražajo dejanske potrebe podjetja. Če je na primer cena ključni faktor za izbiro, bo ta ustrezno obtežena in bo odločilno vplivala na izbiro produkta. Ob koncu dela je podan tudi primer kakšne so tipične uteži za majhno in veliko podjetje.

4.1.1 Prilagodljivost procesu vodenja projektov

Kriterij ocenjuje kako je možno orodje prilagoditi procesu vodenja projektov v podjetju. Kot smo opisali v prejšnjih poglavjih, ima projekt več faz. Zaradi preglednosti programske rešitve je zelo ugodno, če so te faze nastavljive in sledijo dejanskim fazam projektnega procesa. Podobno pomembno je, če lahko v projektnem informacijskem sistemu ločimo med različnimi tipi projektov, npr. investicijski, razvojni, marketinški. Projekti različnih tipov imajo ponavadi različne življenjske cikle, zato je dobro, če vpeljana rešitev omogoča diferenciacijo po tipih projektov. Različni tipi projektov ponavadi uporabljajo tudi različno projektno dokumentacijo, imajo različne časovne plane, različne cilje in zahtevajo različne rezultate. Tako na primer programska rešitev, ki že ob izbiri tipa projekta ponudi privzet časovni plan, olajša delo projektnim vodjem, podjetju pa omogoča, da s popravljanjem privzetega časovnega plana gradi na izkušnjah iz prejšnjih projektov. Nastavljivost privzetih projektnih parametrov zelo približa rešitev naročnikovim željam.

Pomemben vidik vodenja projektov je tudi definicija organizacijske sheme projekta, definiranje vlog udeleženih na projektu in posameznikov, ki imajo v projektnem vodenju posebne vloge, npr. nadzornik projektov, projektna pisarna. Uspešna programska rešitev za vodenje projektov mora omogočiti fleksibilno nastavljivost vlog udeleženih na projektu in vlog, ki se odražajo na celotno projektno okolje. Vloge na projektu avtomatično določajo

tudi pristojnosti pri delu z rešitvijo, npr. kdo lahko popravlja posamezne segmente projekta, kdo ima vpogled v projektne stroške, probleme in podobno.
Največ kar lahko v tem kriteriju pričakujemo od programske rešitev je, da nudi podporo različnim projektnim metodologijam. Ob začetku projekta izberemo po kateri metodologiji želimo projekt voditi, potem pa nas orodje vodi skozi izbrani proces.
Kriteriji, ki določajo prilagodljivost orodij procesu vodenja projektov so zbrani v tabeli 4.2.:

Kriterij	Opis kriterija
Nastavljive faze projekta	Ali programska rešitev podpira različne projektne faze? Je možno te faze nastaviti tako, da sledijo dejanskemu življenjskemu ciklu projektov?
Podpora različnim tipom projektov	Ali orodje loči med različnimi tipi projektov? So tipi projekta nastavljivi? Ali izbran tip določa tudi nadaljnje »obnašanje« projekta?
Nastavljivi tipi projektne dokumentacije	Ali je možna je nastavitev tipov projektne dokumentacije? Podpira nastavitev enega (npr. projektna, tehnična…) ali več nivojev (npr. projektna\ uporabniška, projektna\ zapisniki)? Je mogoč strukturiran pregled dokumentacije?
Podpora privzetim parametrom	Ali je možno je nastaviti privzet časovni plan (npr. glede na tip)? Ali je možno definirati tipične rezultate posameznih faz?
Podpora projektnim metodologijam	Ali je možno nastaviti različne projektne metodologije in projekt voditi po eni izmed njih?

Tabela 4-2: Kriteriji za oceno prilagoditve programske rešitve procesu vodenja projektov.

4.1.2 Podpora časovnemu planiranju

Kriterij ocenjuje kako orodje podpira časovno planiranje projektnih aktivnosti. Gre torej za aplikativno podporo procesu opisanem v poglavju 3.1. Uspešna programska rešitev za vodenje projektov mora omogočati definiranje in pregled seznama aktivnosti, določanje odvisnosti med aktivnostmi, planiranega in dejanskega trajanja aktivnosti, določanje izvajalcev aktivnosti, podpirati grafične tehnike (npr. gantogram), omogočati izračun kritične poti in podobno. Kriteriji so zbrani v spodnji tabeli:

Kriterij	Opis kriterija
Strukturirano definiranje aktivnosti	Ali lahko v rešitvi definiramo projektne aktivnosti? Je možna delitev na podaktivnosti in združevanje v sumarne aktivnosti? Ali lahko definiramo odvisnosti med aktivnostmi (konec ene pomeni začetek druge)?
Načrtovanje trajanja in dela na aktivnostih	Ali lahko za aktivnosti določimo obseg dela, odgovorno osebo in izvajalce? Ali lahko določimo različne kriterije kot osnovo za planiranje (čas, obseg dela)? Ali rešitev upošteva zasedenost virov?
Podpora grafičnim tehnikam	Podpira rešitev prikaz aktivnosti v gantogramu? Ali podpira mrežni diagram prikaza aktivnosti? Ali podpira rešitev kakšne dodatne grafične tipe prikaza?
Podpora tehnikam časovnega planiranja	Ali rešitev podpira metodo kritične poti? Ali podpira analizo v primeru zamujanja aktivnosti oziroma spremembe končnega roka na aktivnosti?

Tabela 4-3: Kriteriji za oceno časovnega planiranja v programski rešitvi.

4.1.3 Spremljanje izvedbe

Merilo uspešnosti programske rešitve za vodenje projektov je tudi podpora spremljanju izvedbe. Pod spremljanje izvedbe uvrščamo beleženje opravljenega dela na aktivnostih, spremljanje planiranih rezultatov, pregled nad problemskimi stanji, možnost določanja preventivnih ukrepov, spremljanje kontrolnih točk projekta, podpora projektni dokumentaciji ipd. V fazi izvedbe projekta nas vedno zanima primerjava med načrtovanih in izvedenim delom, preostali finančni obseg in preostalo delo na projektu. V fazi izvedbe morajo biti omogočeni različni pregledi projektnih podatkov, glede na vlogo posameznega člana projektne skupine. Tako izvajalca zanimajo aktivnosti, na katerih trenutno dela in dokumentacija, ki je povezana z njim. Odgovorni za posamezne aktivnosti želijo spremljati predvsem njim dodeljene aktivnosti, medtem ko se vodja projekta ne želi spuščati v vsako podrobnost, zanima pa ga celotna slika projekta. Podobno kot pri predhodnih kriterijih, so posamezna merila zbrana v tabeli:

Kriterij	Opis kriterija
Spremljanje statusa aktivnosti	Ali imajo aktivnosti različne statuse? Ali lahko na aktivnosti dodajamo poročila o opravljenem delu, stroških, porabljenih urah ?
Podpora projektni dokumentaciji	Ali lahko z rešitvijo obvladujemo projektno dokumentacijo? Lahko dokumentacijo ustrezno strukturiramo? Je mogoč pregleden prikaz projektne dokumentacije?
Pregled problemskih stanj	Lahko s programsko rešitvijo definiramo problemska stanja? Jih spremljamo in obvladujemo rešitve? Lahko na osnovi problema ustvarimo korektivne ukrepe?
Preglednost podatkov	Imajo posamezniki z različnimi vlogami možnost različnih pregledov nad projektnimi podatki (npr. izvajalci aktivnosti, na katerih delajo, projektna pisarna pregled nad statusi projektov)?

Tabela 4-4: Kriteriji za oceno spremljanja dela na projektu.

4.1.4 Komunikacija med člani projektne skupine

Pomemben element za uspešno izvedbo projekta je tudi pretok informacij med člani projektnega tima. Osnova za komuniciranje članov s pomočjo elektronskega medija je vsekakor elektronska pošta. Osnovno je torej, da rešitev omogoča za vse udeležene na projektu določiti njihovo elektronsko pošto. Ta je lahko določena že v sistemu v katerem teče rešitev, sicer pa nam mora rešitev ponujati povezavo med udeleženimi in njihovimi elektronskimi naslovi. Prednost orodja je tudi možnost nastavitve avtomatičnega obveščanja ob določenih dogodkih na projektu (npr. spremembi faze, preteku roka za izvedbo aktivnosti, definiranju problemskega stanja…). Rešitev mora prav tako omogočati ročno obveščanje posameznikov o dogodkih na projektu in beleženje zgodovine obveščanja.

Kriterij	Opis kriterija
Določene elektronske pošte uporabnikov	Ali rešitev omogoča definiranje povezave med posamezniki in njihovimi elektronskimi poštami?
Ročno obveščanje o dogodkih	Ali lahko posamezniki o določenih dogodkih obvestijo ostale člane po elektronski pošti? Ali se beleži zgodovina obveščanja? Imajo obveščeni

	možnost replike? Obstaja diskusijski forum znotraj rešitve?
Avtomatično obveščanje	Ali orodje obvešča izvajalce o aktiviranju aktivnosti? Omogoča rešitev nastavljivost »alarmov«, dogodkov ob katerih naj se avtomatično obvesti izbrane člane tima?

Tabela 4-5: Kriteriji za oceno komunikacije med člani projektnega tima.

4.1.5 Projektna poročila in baza znanja

Kriterij določa ustreznost projektnih poročil in pregled nad zaključenimi projekti, ki predstavljajo bazo znanja o projektnem vodenju v podjetju. Projektna poročila lahko razdelimo na več kategorij: periodična, ob zaključku posameznih faz, zgrajena ročno – na zahtevo uporabnika. Rešitev mora omogočati prilagodljivost poročil glede na zahteve uporabnika. Poročila lahko razdelimo tudi na poročila o določenem projektu, ali na poročila multiprojektnega okolja (poročila na nivoju podjetja). Prva zanimajo predvsem projektnega vodjo in člane projektne skupine, druga pa vodstveni kader in projektno pisarno. Zaradi preglednosti je pomembna tudi podpora različnim grafičnim prikazom, kot so stolpčni, črtni in tortni diagrami. Namen vsakega orodja je prav tako zgraditev baze znanja, na kateri podjetje gradi svoje nadaljnje delo, se uči iz predhodnih napak in izboljšuje poslovanje na osnovi zaključenih projektov. Programska rešitev za vodenje projektov mora nuditi pregleden arhiv zaključenih projektov, omogočati iskanje po podatkih zaključenih projektov in izpostaviti pridobljene izkušnje.

Kriterij	**Opis kriterija**
Prednastavljena poročila	Podpira rešitev vnaprej nastavljena periodična in zaključna poročila? So poročila pogoj za prehod v naslednjo fazo, zaključek aktivnosti?
Poročila na zahtevo uporabnika	Ali lahko uporabnik sam ustvari projektna poročila? Ali jih lahko sam oblikuje, prilagodi svojim potrebam?
Poročila multiprojektnega okolja	Podpira rešitev poročila, ki zajemajo več projektov? Nudi pregled nad statusi projektov, obremenitvijo zaposlenih?
Baza znanja	Ali obstaja arhiv zaključenih projektov? Je struktura arhiviranih podatkov ustrezna? So možne analize in poizvedbe po arhiviranih projektih?

Tabela 4-6: Kriteriji za oceno projektnih poročil in oceno zgrajene baze znanja

4.1.6 Tehnična ustreznost rešitve

Pod tehnično ustreznost rešitve prištevamo merila, ki se vežejo neposredno na tehnične lastnosti rešitve. Ena od takšnih so platforme, ki jih programska rešitev podpira. Možnost uporabe na različnih platformah oziroma operacijskih sistemih je seveda prednost. Ker so pri projektu večinoma udeleženi tudi zunanji sodelavci, je zelo pomembno, če jim lahko omogočimo dostop do naše programske rešitve za vodenje projektov. To z drugimi besedami pomeni, da mora rešitev omogočati tudi spletni dostop uporabnikom. Rešitve, ki omogočajo spletni dostop so v prednosti tudi zaradi dejstva, da ni potrebno nastavljati posebnih odjemalcev na strani uporabnika. To zniža stroške implementacije in vzdrževanja sistema. Pri spletnih rešitvah je administracija sistema vezana samo na strežniško plat, kar pomeni večjo obvladljivost in manjše stroške. Večina rešitev za vodenje projektov pri

svojem delu shranjuje podatke v relacijski bazi podatkov. Pomembno je da rešitev za vodenje projektov omogoča povezovanje na čim večje število na trgu prisotnih relacijskih baz.

Kriterij	Opis kriterija
Podpora različnim platformam	Deluje rešitev na različnih platformah oziroma operacijskih sistemih? Kakšne so potrebe po strojni opremi?
Način dostopa končnih uporabnikov	Potrebujejo končni uporabniki posebno programsko opremo za delo z rešitvijo? Ali rešitev podpira spletni dostop končnih uporabnikov? Ali moramo za delovanje rešitve pri končnem uporabniku namestiti kakšno dodatno programsko opremo?
Podpora različnim relacijskim bazam	Ali shranjuje rešitev podatke v relacijski bazi? Koliko različnih relacijskih baz podpira rešitev ?

Tabela 4-7: Kriteriji za oceno tehnične ustreznosti programske rešitve

4.1.7 Cena

Po vrstnem redu v nalogi cena produkta sicer nastopa kot zadnji kriterij, v praksi pa je večinoma ključni oziroma izločitveni kriterij. Cene programskih rešitev za vodenje projektov se gibljejo v velikem razponu, odvisno od kompleksnosti in nastavljivosti rešitve ter proizvajalca. Najcenejše so seveda rešitve namenjene zgolj časovnemu planiranju in enemu uporabniku. Veliko takšnih rešitev je celo zastonj, cena pa ne bi smela presegati $25. Nekoliko obsežnejši produkti, ki podpirajo spremljanje projekta in delo nekaj uporabnikov, se gibljejo v cenovnem razredu od $300 do $500. Ceno za produkte, ki omogočajo delo velikemu številu uporabnikov v multiprojektem okolje je zelo težko opredeliti, lahko pa doseže tudi nekaj deset tisoč ameriških dolarjev. Pri ceni uvajanja rešitve pa ni pomembna samo cena strežniških oziroma uporabniških licenc. Pomembno je oceniti tudi koliko bodo stale morebitne prilagoditve rešitve na naše želje in kakšne so cene povezane z izobraževanjem končnih uporabnikov.

Kriterij	Opis kriterija
Cena osnovnega paketa	Kolikšna je cena strežniške in uporabniških licenc za osnovno verzijo programske rešitve?
Cena nadgradnje posameznih modulov	Koliko bodo stale prilagoditve glede na želene nadgradnje in prilagoditve programske rešitve?
Cena izobraževanja končnih uporabnikov	Koliko uporabnikov želimo usposobiti za delo z rešitvijo ? Kakšen je obseg izobraževanja? Ali potrebujejo zaradi nove rešitve še kakšno dodatno izobraževanje?

Tabela 4-8: Kriteriji za oceno cenovne ustreznosti programske rešitve

Pred izvedbo ocene posameznih programskih rešitev sledi še kratek opis preizkušenih rešitev. V delu bodo testirane tri programske rešitve, in sicer MS Project 2000, e-Proj in Primavera TeamPlay. Opis rešitev temelji na gradivu, ki ga poleg rešitev ponujajo proizvajalci in na testiranju rešitev.

4.2 Opis preizkušenih programskih rešitev

Opisi posameznih programskih rešitev so v delu razdeljeni na 4 podpoglavja:

- proizvajalec in namen rešitve,
- glavne lastnosti rešitve,
- tehnične zahteve,
- implementacija v praksi.

Namen prvega podpoglavja je opredeliti proizvajalca in prodajni moto s katerim programska rešitev nastopa na trgu. V naslednjem poglavju sledi opis glavnih lastnosti rešitve z nekaj tipičnimi zaslonskimi maskami rešitve. V poglavju o tehničnih zahtevah je opredeljena trenutna razvojna verzija programske rešitve, okolje v kateri rešitev deluje in zahteve strojne opreme za delovanje rešitve. Zadnje poglavje definira obseg že implementiranih verzij rešitve in skuša oceniti prisotnost na trgu.

4.2.1 MS Project 2000

4.2.1.1 Proizvajalec in namen rešitve

Programska rešitev MS Project 2000 je produkt podjetja Microsoft Corp. (spletni naslov http://www.microsoft.com). Orodje Microsoft Project 2000 je namenjeno načrtovanju, časovnemu planiranju ter spremljanju stroškov in virov projekta. Delo z orodjem poteka na samostojnem osebnem računalniku, torej ne podpira tehnologije odjemalec strežnik. Del celotnega programskega paketa je tudi rešitev Microsoft Project Central, ki podpira tudi večuporabniško okolje odjemalec-strežnik. V delu se bomo posvetili izključno verziji Microsoft Project 2000.

Produkt je namenjen uporabi v različnih gospodarskih panogah od gradbeništva, farmacije pa do informacijskih podjetij. Programska rešitev želi pridobiti tako uporabnike, ki so vešči v poznavanju teorije projektnega managementa, kakor tudi začetnike na tem področju. Orodje ponuja odlično podporo prav začetnikom, saj jim je v pomoč veliko orodij, čarovnikov, rešitev pa ponuja zelo podoben uporabniški vmesnik kakor ostale rešitve paketa MS Office.

4.2.1.2 Glavne lastnosti rešitve

Glavna lastnost programske rešitve MS Project 2000 je predvsem podpora časovnemu planiranju in spremljanju projekta. V tem sklopu nam orodje ponuja naslednje glavne funkcionalnosti:

- definiranje seznama aktivnosti, kjer naštejemo vse projektne aktivnosti.
- določanje odvisnosti med aktivnostmi, s čimer za vsako aktivnost določimo njene predhodne in naslednje aktivnosti in tipe povezav med aktivnostmi (npr. dve aktivnosti se začneta sočasno).
- definiranje obsega dela, izvajalcev in trajanja aktivnosti. Programska rešitev vedno upošteva razmerje, da je trajanje aktivnosti produkt med delom in dodeljenimi izvajalci

(človek dnevi). Na aktivnosti lahko fiksiramo eno od treh spremenljiv (trajanje, delo oziroma čl./dni), ostali dve pa se prilagata vnosu in fiksirani vrednosti.

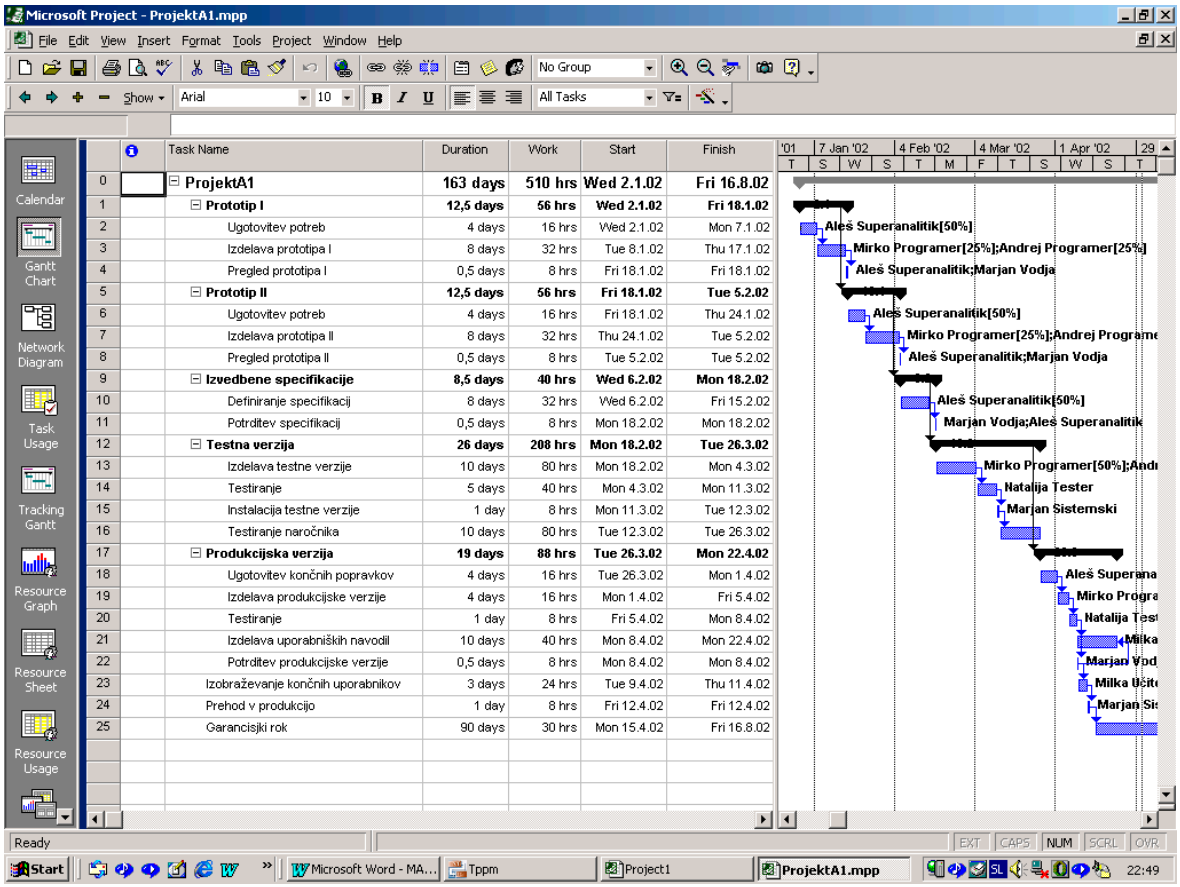
- združevanje aktivnosti nam omogoča, da drevesno strukturo aktivnosti tako, da so določene aktivnosti drugim nadrejene oziroma podrejene. Trajane in delo za nadrejene aktivnosti se avtomatično izračuna iz podrejenih aktivnosti.
- določanje kontrolnih točk, ki nimajo opredeljenega trajanja, ampak določajo točno določene dogodke v projektu.
- za vsako aktivnost lahko določimo tudi posebne omejitve. Takšne omejitve so recimo zadnji sprejemljivi končni rok aktivnosti, datum pred katerim se aktivnost ne sme začeti ipd.
- na vsaki aktivnosti lahko določimo procent izvedbe. Procent neposredno določa tudi status aktivnosti: planirane aktivnosti imajo procent realizacije 0,aktivnosti v izvajanju med 0 in 100, zaključene pa 100.
- aktivnosti lahko določimo tudi prioriteto, ki določa pomembnost aktivnosti. Prioriteta je število v obsegu med 0 in 1000.

Definiran časovni plan je podprt tudi z dobrim uporabniškim vmesnikom. Tako lahko aktivnosti filtriramo po različnih kriterijih, npr. pregledujemo samo aktivnosti v izvajanju, samo kritične aktivnosti ipd. V uporabniškem vmesniku lahko aktivnosti tudi grupiramo po različnih kriterijih, npr. po trajanju, omejitvah, virih, kontrolnih točkah ipd.

Programska rešitev nam nudi tudi nekaj različnih pregledov nad projektnimi aktivnostmi. Osnovni pregled aktivnosti je gantogram, kjer je na absciso nanesen čas, trajanje aktivnosti pa je prikazano s stolpci. Naslednji pregled je diagram, ki predstavlja aktivnosti kot vozlišča mrežnega diagrama. Različni tipi aktivnosti so prikazani z različnimi liki, aktivnosti pa so medsebojno povezane s puščicami. Programska rešitev nam omogoča tudi izračun kritične poti, ki določa katere aktivnosti ne smejo zamujati, če ne želimo prekoračiti načrtovanega roka zaključka projekta. V rešitvi imamo tudi koledarski in grafični pregled obremenitve virov. Možen je pregled obremenitve po posameznih virih ali pa skupen pregled zasedenosti virov na projektu.

Če želimo primerjati podatke v fazi planiranja in izvedbe projekta, lahko projekt shranimo z osnovnimi, planiranimi podatki. V tem načinu lahko kasneje primerjamo planirane in dejansko izvedene roke in ure na aktivnostih.

Programska rešitev MS Project 2000 nudi tudi možnost prilagoditev oziroma programiranja specifičnih zahtev naročnika. Rešitev podpira standard COM operacijskega sistema Windows, kar pomeni, da lahko z različnimi programskimi jeziki dostopamo do praktično vseh delov rešitve. Tako lahko recimo povežemo nabor virov projekta s seznamom zaposlenih v podjetju.



Slika 4-1: MS Project 2000: Seznam aktivnosti in gantogram

4.2.1.3 Tehnične zahteve

Tip	Opis zahtev
Odjemalec	• osebni računalnik z Windows operacijskim sistemom z od 30MB (tipična instalacija) do 240 MB (celotna instalacija) prostora na trdem disku • minimalno 32 MB RAMa

Tabela 4-9: Tehnične zahteve rešitve e-Proj

4.2.1.4 Implementacija v praksi

Kljub temu, da je prišla rešitev MS Project na trg orodij za podporo projektnemu managementu precej pozno (1990), ji je uspelo v svoji kategoriji orodij zavzeti levji delež na trgu. K temu je najbolj prispeval agresiven prodajni pristop, odlična integriranost z okoljem Windows in ugodna cenovna politika. Rešitev tako zavzema vodilno mesto v kategoriji projektnih orodij namenjenih uporabi na enem osebnem računalniku, ki so dostopna za ceno manj kot $1000. Z rešitvijo MS Project Central pa podjetje Microsoft očitno kaže zanimanje tudi za trg orodij, ki podpirajo večuporabniška multiprojektna okolja.

4.2.2 e-Proj

4.2.2.1 Proizvajalec in namen rešitve

Programska rešitev e-Proj je produkt slovenskega podjetja Genis d.o.o (spletni naslov podjetja http://www.genis.si). V gradivu, ki ga podjetje Genis d.o.o. prilaga k programski rešitvi, najdemo moto programske rešitve (2002):
"Programska rešitev e-Proj omogoča boljše obvladovanje posameznega projekta ter vodenje in usklajevanje dejavnosti z drugimi projekti. Rešitev poudarja mehanizme pospeševanja in zasleduje princip prijaznosti do uporabnika - pomoč pri obvladovanju projekta za vodjo projekta, izvajalce in nadzorni svet.V osnovi rešitev podpira vse temeljne metodološke pristope, izhaja pa iz standarda PMI. Ta predvsem izpostavlja praktičen pristop in procese pospeševanja projektov. Zato poskuša vodenje projektov vključiti v poslovno okolje, ki pa se od podjetja do podjetja razlikuje."
Iz naštetega je razvidno, da rešitev sledi enemu temelnjih motov podjetja Genis d.o.o.: "Iz prakse za prakso". Cilji uporabe rešitve e-Proj so naslednji:

- zagotoviti preglednost postopkov in projektne dokumentacije,
- omogočiti organizirano vodenje in spremljanje nalog,
- vzpostaviti skupno bazo podatkov o projektih,
- zagotoviti postopno gradnjo baze znanja o projektih,
- povečati motiviranost udeležencev v projektu.

4.2.2.2 Glavne lastnosti rešitve

Programska rešitev e-Proj predstavlja integrirano projektno programsko rešitev, ki podpira multiprojektno okolje, spremljanje različnih tipov projektov in definiranje različnih vlog za udeležene pri projektu.
Prva pomembna lastnost rešitve je parametrična zasnova. Parametrična zasnova pomeni nastavljivost projektnih parametrov. V rešitvi e-Proj je možno nastaviti:

- različne tipe projektov in njim prirejene nadzornike,
- faze projekta in prehode med fazami (elektronska odobritev zaključka faze),
- različne tipe projektne dokumentacije (poljubno število nivojev, npr. »Projektna dokumentacija\ Navodila\ Uporabniška navodila …),
- predpisano obliko posameznih tipov dokumentacije (npr. dopis ima vedno isto obliko),
- referenčni terminski plan za posamezen tip projekta,
- način obvladovanja sprememb projektnega elaborata,
- pravice dostopa do podatkov (kakšne podatke lahko vidijo posamezniki z določenimi vlogami na projektu).

Z naštetimi nastavitvami je možno zadovoljiti osnovne naročnikove zahteve po prilagoditvi rešitve na način dela v podjetju. Za vsako multiprojektno okolje je še posebej pomembna razdelitev na različne tipe projektov, ki povečuje preglednost rešitve in jasno razdeli različna projektna področja.

Naslednja pomembna lastnost orodja e-Proj je modularnost. Modularnost pomeni, da je rešitev razdeljena v različne vsebinske module.Moduli programske rešitve e-Proj so prikazani v spodnji tabeli.

Modul	Opis modula
Projektni elaborat	Projektni elaborat opredeljuje namen in cilje projekta, definira organizacijsko strukturo projekta, skratka določa vse podatke povezane s planiranjem projekta.
Časovni plan	Časovni plan je v bistvu sestavni del projektnega elaborata, določa pa časovni potek projektnih aktivnosti, opredeljuje kontrolne točke in določa izvajalce aktivnosti.
Dokumentacija	Dokumentacijski modul je namenjen spremljanju in obvladovanju projektne dokumentacije.
Poročila	Modul poročil je namenjen spremljanju opravljenega dela izvajalcev na posameznih projektnih aktivnosti. Poročila izvajalcev so osnova za periodična in mesečna poročila za aktivnosti in projekt.
Stroški	S stroški spremljamo planirana in že porabljena sredstva na projektu. Modul se lahko poveže tudi na že obstoječ informacijski sistem namenjen spremljanju finančnega poslovanja podjetja.
Kartica projekta	Kartica projekta nosi osnovne podatke o projektu in je namenjena hitremu pregledu nad stanjem projekta. Nosi informacijo o projektni strukturi, statusu, že opravljenem delu in porabljenih sredstvih. Namenjena je predvsem nadzornikom, kot trenutni posnetek stanja na projektu.

Tabela 4-10: Moduli programske rešitve e-Proj

Časovni modul rešitve nudi tudi integracijo z rešitvijo MS Project. Tako je možen prenos časovnega plana (aktivnosti, rokov, statusa, virov) z MS Projecta v e-Proj in obratno. Rešitev omogoča popolno integracijo med obema rešitvama in večkratno sinhronizacijo podatkov v obeh smereh.

Pri uvajanju rešitve e-Proj ni nujna hkratna uvedba vseh modulov. Modularnost omogoča naročniku, da se odloči spremljati samo tiste podatke (module), ki ga dejansko zanimajo pri projektu. Seveda je po prehodu v produkcijo mogoče brez težav dodajati oziroma odvzemati posamezne module.

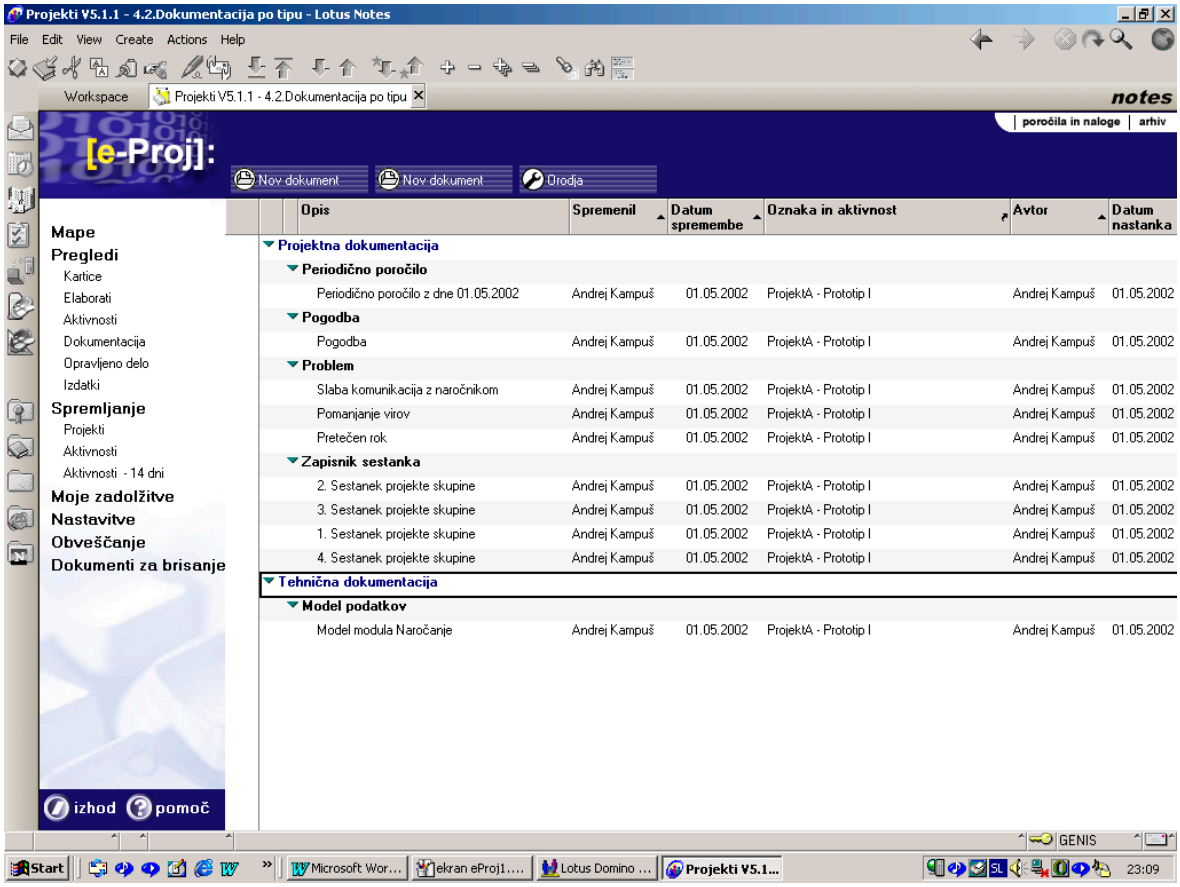
Naslednja lastnost, ki jo ponuja programska rešitev e-Proj je odprtost in varnost. Kot odprtost se smatra zelo široka možnost uporabe programske rešitve. Rešitev omogoča dostop končnih uporabnikov preko spletnega brskalnika, kar pomeni, da lahko pri spremljanju in izvedbi projekta sodelujejo tudi zunanji sodelavci. Pri projektnih podatkih se vedno ponuja tudi vprašanje varovanja podatkov. Rešitev e-Proj ponuja več nivojev varovanja nad vsakim sklopom podatkov. Za vsakega uporabnika posebej je določeno ali lahko pregleda določene podatke, ali jih lahko popravlja oziroma ali lahko dodaja nove podatke.

Rešitev e-Proj omogoča dobro podporo vodjem projekta in nadzornikom. Omogoča nastavitev elektronskega potrjevanja posameznih faz projekta, kar pomeni, da so nadzorniki vedno obveščeni o vsebini in stanju projektov. Z rešitvijo se beleži tudi zgodovina potrditev projektnih faz, obveščanja posameznikov o dogodkih in podobno.

Programska rešitev e-Proj nudi tudi različne preglede nad projektnimi podatki. Vsi podatki o določenem projektu se zbirajo v projektni mapi. V multiprojektnem okolju pa so možni pregledi po tipih projekta, po datumih, po avtorju dokumentov, pregledi pa se lahko prilagodijo tudi željam naročnika. Izvajalci aktivnosti imajo pregled nad aktivnostmi na katerih trenutno delajo.

S časom uporabe rešitve e-Proj se gradi tudi baza znanja. Z baze znanja lahko ugotovimo povprečno trajanje projektov, razmerje med načrtovanim in realiziranim delom, težave, ki so nastajale pri projektu ipd.

Programska rešitev e-Proj torej omogoča podjetju implementacijo multiprojektnega, večuporabniškega informacijskega sistema. Rešitev izhaja iz procesa vodenja projektov v organizaciji in temelji na prijaznosti do končnih uporabnikov in preglednsoti podatkov.



Slika 4-2: e-Proj: Pregled projektne dokumentacije.

4.2.2.3 Tehnične zahteve

Tip	Opis zahtev
Strežnik	• Domino strežnik verzija 5.0.2b ali višji, ki teče na kateremkoli Windows operacijskem sistemu, AIX ali UNIX platformi • minimalno 128 MB RAMa
Odjemalec	Osebni računalnik s programsko opremo: • Lotus Notes verzija 5.0.2b ali višja, • MS Project 98 ali višji (za delo s časovni plani programa MS Project), • spletni brskalnik MS Explorer 5.5 ali višji (za spletni dostop),

• minimalno 32 MB RAMa.

Tabela 4-11: Tehnične zahteve rešitve e-Proj

4.2.2.4 Implementacija v praksi

Trenutno je programska rešitev uspešno uvedena v 10 slovenskih podjetjih. Podjetje Genis d.o.o. načrtuje z programsko rešitvijo prodreti tudi na tuje trge. Rešitev je že prevedena v hrvaški jezik, trenutno pa nastaja tudi angleška verzija rešitve.

4.2.3 Primavera TeamPlay

4.2.3.1 Proizvajalec in namen rešitve

Primavera Teamplay je programska rešitev podjetja Primavera Systems, Inc. (spletni naslov podjetja je http://www.primavera.com). Podjetje Primavera Systems, Inc ponuja na trgu celo paleto programskih rešitev za vodenje projektov. Najbolj znano orodje je Primavera Project Planner, ki je orientirano predvsem na področja posameznih inženirskih gospodarskih panog. Za razliko od rešitve Project Planner, je orodje Primavera TeamPlay namenjeno posebej podpori projektom s področja informacijske tehnologije.
Glavni cilji na katerih temelji programska rešitev so:

- najbolje izkoristiti vložena sredstva,
- povečati preglednost projektnih podatkov,
- povečati hitrost in učinkovitost izvedbe projektov,
- zagotoviti ponovljivost.

Namen programske rešitve Primavera TeamPlay je, da podjetje poveča izkoristek vloženih sredstev (ROI). Rešitev omogoča, da so cilji izvajanih projektov v skladu s strategijo podjetja in da je dodana vrednost pri projektu največja. Rešitev nudi ažurne podatke vsem udeleženim pri projektu, s čemer se poveča tudi učinkovitost in hitrost izvedbe projektov. Z uporabo rešitve dobi podjetje arhivsko bazo projektov, s pomočjo katere dobi univerzalen in skladen proces za vodenje projektov.

4.2.3.2 Glavne lastnosti rešitve

Programsko rešitev Primevera TeamPlay tvorijo 4 glavni moduli, ki so našteti v tabeli 4-12:

Modul	Namen modula
Project Manager	Glavni modul, ki omogoča načrtovanje in kontrolo projektov v multiprojektnem okolju.
TeamPlayer	Modul, ki omogoča spletni dostop do projektnih podatkov
Portfolio Analyst	Modul omogoča spremljanje projektnih portfeljev
Methodology Manager	Podpora različnim projektnih metodologijam za vodenje projektov

Tabela 4-12: Moduli programske rešitve Primavera TeamPlay

Glavni modul programske rešitve je Project Manager, integriran programski modul za vodenje projektov. Namen modula je zagotoviti izčrpno zbirko podatkov o vseh projektih v

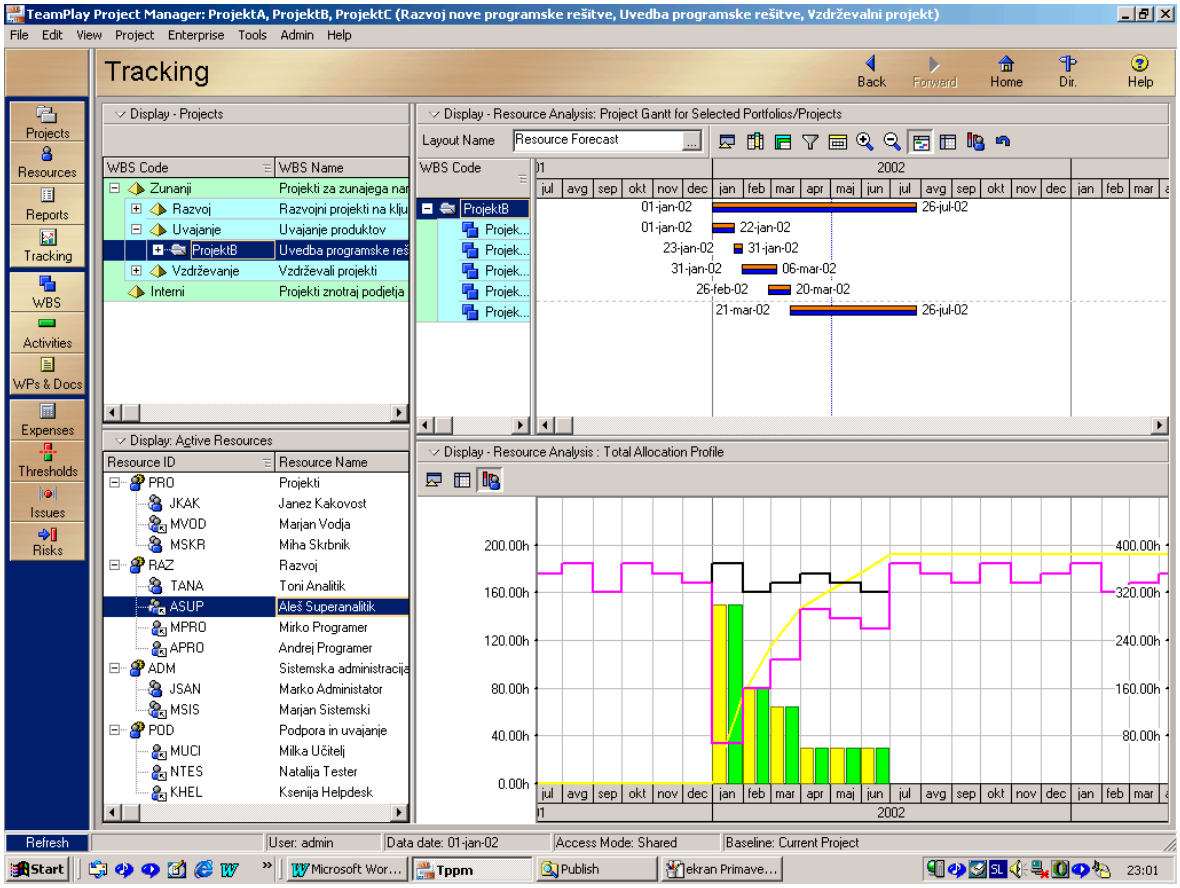
podjetju, tako da so podatki pregledni tako na vodstvenem kakor tudi na izvedbenem nivoju. Komponente modula zagotavljajo vsestranski nadzor nad projektom in zagotavljajo načrtovanje projekta, časovno planiranje in poročanje. Modul lahko ločimo na 12 vsebinskih sklopov, ki so med sabo povezani in se dopolnjujejo. Sklopi in njihovi opisi so našteti v nadaljevanju:

1. **razčlemba dela** predstavlja hierarhično organizirane produkte in storitve, ki jih želimo ustvariti s projektom. Razčlemba dela je tipično prva stvar, ki jo naredi vodja projekta pri načrtovanju projekta. Na ustvarjeno strukturo (angleško WBS – Work Breakdown Structure) kasneje povežemo produkte in dokumente in definiramo aktivnosti potrebne za dosego načrtovanih rezultatov. V programski rešitvi Primavera Teamplay predstavlja razčlenjena struktura dela tudi osnovo za spremljanje tveganja, problemov in stroškov projekta.
2. **dodana vrednost** je tehnika merjenja uspešnosti izvedbe projekta glede na planirane stroške in roke. Temelji na primerjavi načrtovanega in dejansko opravljenega dela. Tipično se veže na predhodno definirano razčlembo dela, čeprav povezava ni nujna.
3. **razčlemba organizacijske strukture** je strukturirana delitev organizacije projekta. Struktura ni vezana na samo en projekt, temveč jo lahko uporabimo v multiprojektnem okolju. Organizacijska struktura določa tudi pravice dostopa do posameznih projektnih podatkov.
4. **projektni viri in vloge** definirajo posameznike in opremo potrebne za izvedbo projektnih aktivnosti. Viri so časovno omejeni in deljeni med aktivnostim vseh projektov multiprojektnega okolja. Seveda so viri povezani z definirano organizacijsko strukturo. Vsak vir ima definiran koledar, ki določa kdaj je vir dosegljiv, posameznikom pa lahko določimo tudi dodatne podatke, kot so vloge, kontaktne informacije in vrednost ure. Z dodajanjem virov na posamezne aktivnosti, avtomatično dobimo tudi pregled zasedenosti zaposlenih in ostalih virov v podjetju. Programska rešitev nam omogoča tudi določanje največje obremenjenosti virov in določanje tolerance, do katere mere še lahko obremenimo posamezni vir. Rešitev nam omogoča tudi definiranje standardnih vlog, vezanih na tip dela (npr. razvijalec programske opreme, administrator baze podatkov itd.). Te vloge lahko vežemo na aktivnosti in na posameznike. Na ta način je mogoče tudi spremljanje znanja zaposlenih in načrtovanje njihovega izobraževanja.
5. **stroški** v kontekstu rešitve Primavera TeamPlay predstavljajo vse izdatke povezane s projektom razen dela zaposlenih. Tipično so to enkratni in nepovratni stroški, na primer stroški nabave materiala, potovanj, izobraževanja. Stroške lahko povežemo tudi na stroškovna mesta in spremljamo njihovo porabo tokom izvedbe projekta.
6. **koledar** definira število delovnih dni in delovnih ur za vsak delovni dan v koledarju. Različni koledarji se povežejo na posamezne aktivnosti in vire, tako da lahko pri planiranju upoštevamo praznike, dopuste posameznikov in druge predvidene odsotnosti. Programska rešitev Primavera TeamPlay loči tri vrste koledarjev: koledar za vse projekte, koledarje za posamezne projekte in koledarje za posamezne vire.

7. **aktivnosti** v programski rešitvi so najnižji nivo projekta, katerega še neposredno spremlja vodja projekta. Za vsako aktivnost definiramo tipične podatke, ki jih opredeljuje projektni management: oznaka oziroma koda aktivnosti, začetni in končni datum, tip, procent realizacije, stroške, predhodne in naslednje aktivnosti, vire ipd. Na aktivnost lahko povežemo tudi projektno dokumentacijo in rezultate. Seveda lahko v programski rešitvi določimo tipične odvisnosti med aktivnostmi, kot so konec prve aktivnosti pomeni začetek naslednje in podobno. Po definiranju odvisnosti med aktivnosti, dobimo za vsako aktivnost tudi pregled morebitnih časovnih rezerv oziroma pregled kdaj lahko najkasneje začnemo z aktivnostjo, ne da bi to vplivalo na celoten projektni plan.
8. **problemi** so znane težave na projektih, ki zahtevajo pozornost oziroma korektivne ukrepe. S programsko rešitvijo lahko probleme definiramo ročno, lahko pa tudi nastavimo dogodke, ki avtomatično kreirajo problemska stanja. Takšni avtomatični dogodki se lahko vežejo na razčlembo dela, aktivnosti ali na projektne vire. Tipična problemska stanja, ki jih ponuja programska rešitev so prekoračitev stroškov, prekoračitev planiranega roka zaključka aktivnosti, preobremenjenost virov ipd. Vsakemu problemskemu stanju lahko določimo prioriteto, način spremljanja in odgovorno osebo za odpravo problema.
9. **obvladovanje tveganja** omogoča ugotovitev, opredelitev in določanje prioritete potencialnemu tveganju na projektu. Uporabniki lahko načrtujejo tveganje na projektu, tako da opredelijo verjetnost pojave tveganja, to verjetnost pa povežejo na posamezne projektne aktivnosti. Za vsako možno tveganje lahko določimo tudi najverjetnejši datum nastanka, obseg in število virov potrebnih za odpravo nastalega problema. Z naštetimi podatki programska oprema izračuna, kako bi uresničitev tveganja vplivala na celotni projekt in za koliko bi se povečali stroški projekta.
10. **izdelki in dokumentacija** so vsi načrti, zapisniki, preglednice, kontrolni listi in vsa ostala dokumentacija, ki spremlja načrtovanje in planiranje projekta. Programska rešitev omogoča uporabnikom povezavo dokumentacije na posamezne aktivnosti in spremljanje njihovega življenjskega cikla. Isto kot za projektno dokumentacijo velja tudi za rezultate projekta, le da te ponavadi definiramo že ob načrtovanju projekta, med izvajanjem pa v glavnem spremljamo njihovo realizacijo.
11. **spremljanje projekta** omogoča uporabnikom spremljanje izvedbe na osnovi palčnih diagramov, gantograma, profila aktivnosti, profila virov in profila stroškov. Načrt spremljanja je lahko dostopen vsem ali pa samo določenim uporabnikom znotraj projekta. Definirane profile spremljanja lahko tudi ustrezno filtriramo, glede na posamezne vire oziroma aktivnosti.
12. **poročanje** v programski rešitvi Primavera TeamPlay vsebuje vnaprej definirana poročila in čarovnike za ustvarjenje poročil, ki omogočajo uporabnikom strukturiran pregled projektnih podatkov. Projektna poročila omogočajo uporabnikom grupiranje, sortiranje in filtriranje projektnih podatkov. Poročila so dostopna tudi v HTML obliki,

tako da so primerna za objavo in dostop preko spleta. Uporabniki so lahko tipična poročila tudi shranijo in na ta način ponovno uporabijo kasneje.

Poleg naštetih funkcionalnosti programske rešitve velja omeniti še veliko fleksibilnost pri administraciji in nastavljivosti rešitve. Za projekte v podjetju lahko nastavimo tipične nastavitve, ki bodo veljale za vse projekte in na ta način zagotovimo enotnost procesa vodenja projekta v podjetju. V rešitvi je dobro podprto tudi varovanje podatkov, saj lahko vsakemu uporabniku nastavimo pravice tako na nivoju rešitve, kakor tudi na nivoju posameznih projektov.



Slika 4-3: Primavera TeamPlay: Spremljanje zasedenosti virov.

4.2.3.3 Tehnične zahteve

Tip	Opis zahtev
Strežnik	• Oracle ali MS SQL relacijska baza, ki teče na operacijskem sistemu Windows ali UNIX • minimalno 128 MB RAMa
Odjemalec	• osebni računalnik z Windows operacijskim sistemom in 50MB prostora na trdem disku • minimalno 32 MB RAMa

Tabela 4-13: Tehnične zahteve rešitve Primavera TeamPlay

4.2.3.4 Implementacija v praksi

Po oceni skupine GartnerGroup (Stang, 2000) ponuja podjetje Primavera Systems vodilna orodja v kategoriji projektnih informacijskih sistemov najvišjega razreda, ki podpirajo projektne zahteve celotnega podjetja. Tako je za velika podjetja s področja gradbeništva in drugih inženirskih panog vodilo orodje Primavera Project Planner, medtem ko je orodje Primavera TeamPlay še posebej namenjeno podjetjem s področja informacijske tehnologije.

4.3 Primerjava preizkušenih rešitev

4.3.1 Testno okolje

Pred dejansko izvedbo testa je potrebno dobro definirati okolje, ki ga bomo v rešitvah implementirali. Za primer bomo vzeli namišljeno, srednje veliko računalniško podjetje (za slovenske razmere), katerega primarna dejavnost je razvoj programskih rešitev za zunanjega naročnika. Namišljeno podjetje zaposluje 30 ljudi, pri testnih projektih pa bodo udeleženi sistemski analitiki, uvajalci programske opreme, sistemski administratorji, programerji in seveda vodje projektov. Vsi testni projekti so projekti za zunanjega naročnika, razlikujejo pa se po vsebini. Testno okolje predstavljajo trije projekti z naslednjo vsebino:

- projekt A: razvoj nove programske rešitve,
- projekt B: uvedba programske rešitve in
- projekt C: vzdrževalni projekt.

Cilj projekta A je razvoj nove programske rešitve, ki bo zadovoljila želje naročnika. Projekt bo vseboval naslednje aktivnosti (v grobem): zajem želja naročnika, izdelava in pregled prototipa, izdelava testne verzije, testiranje, izdelava končne verzije, izdelava uporabniških in skrbniških navodil, izobraževanje končnih uporabnikov, prehod v produkcijo in garancijski rok. Pri izvedbi projekta bodo udeleženi predvsem sistemski analitiki, programerji in preizkuševalci programske opreme.

Namen projekta B je uvedba programske rešitve, ki jo podjetje že trži kot produkt, v novo okolje. Tipične projektne aktivnosti bodo: predstavitev funkcionalnosti produkta, pregled potreb naročnika, nastavitev rešitve za naročnika, izobraževanje ožje skupine, uvedba v pilotnem okolju, izobraževanje končnih uporabnikov, testna uvedba rešitve, prilagoditev navodil, prehod v produkcijo, garancijski rok. Pri izvedbi projekta bodo udeleženi predvsem vodja produkta, uvajalci programske rešitve in izobraževalni center.

Projekt C je projekt namenjen vzdrževanju že implementiranih programskih rešitev pri naročniku. Aktivnosti so opredeljene z vzdrževalno pogodbo, ki določa vzdrževanje že implementiranih programskih rešitev (Rešitev 1, Rešitev 2, Rešitev 3). Vsaka od programskih rešitev ima svojega skrbnika, za vsako od rešitev pa je predvideno največ 10 intervencij mesečno. Vzdrževalna pogodba je podpisana za obdobje pol leta, vključuje pa tudi nadgradnje obstoječih programskih rešitev, ko so pripravljene nove verzije teh rešitev. Čeprav projektnih aktivnosti ne moremo točno definirati (npr. ne poznamo obsega in

pojavnosti intervencij, točnega datuma nadgradnje), zahteva projekt planiranje zasedenosti virov. S projektom želimo tudi spremljati koliko intervencij je bilo mesečno opravljenih za posamezno rešitev in koliko časa je predstavljalo njihovo reševanje.
Vsi projekti so časovno postavljeni v isto obdobje (prvo polletje leta 2002), tako da bo možno v testnem okolju simulirati povezavo med projekti (npr. delitev istih virov med projekti). Poleg samega načrtovanja vseh treh rešitev, želimo v testu preizkusiti tudi spremljanje izvedbe projekta. Kako rešitve podpirajo ustvarjanje projektne dokumentacije, spremljanje dela na aktivnostih, probleme in tveganja na projektu. Vsaka od rešitev bo ocenjena po kriterijih opisanih v poglavju 4.1.
Vse rešitve so bile preizkušene na IBM kompatibilnem osebnem računalniku z operacijskim sistemom Windows 2000, 700 MHz Intelovim procesorjem in 128 MB RAMa. Licence za preizkus rešitev MS Project 2000 in e-Proj (verzija 5.1.1.) je zagotovilo podjetje Genis d.o.o. Testno verzijo produkta Primavera TeamPlay (verzija 2.1.05) pa je zagotovilo hrvaško podjetje DelaComp d.o.o., ki je zastopnik za rešitve Primavera za področje Hrvaške in Slovenije. Vsi rezultati testiranja posameznih rešitev so zbrani v prilogi A.

4.3.2 Ocena posameznih rešitev

4.3.2.1 MS Project 2000

Tabela 4-14 prikazuje ocene dobljene pri preizkusu programske rešitve MS Project 2000 Prikazani so samo rezultati glavnih kriterijev, ocene posameznih podkriterijev, skupaj z opisi ocen, pa so v prilogi A.1.

Kriterij	Ocena
1. Prilagodljivost procesu vodenja projektov	0,6
2. Podpora časovnemu planiranju	8,0
3. Spremljanje izvedbe	2,0
4. Komunikacija med člani projektne skupine	3,3
5. Projektna poročila in baza znanja	3,5
6. Tehnična ustreznost rešitve	2,3
7. Cena	7,7

Tabela 4-14: Ocene glavnih kriterijev za programsko rešitev MS Project 2000

Iz rezultatov je očitno, da izstopa programska rešitev predvsem v časovnem planiranju in ceni. Časovno planiranje omogoča večnivojsko definiranje aktivnosti, določanje njihovih medsebojnih odvisnosti in načrtovanje praktično vseh podatkov, ki nas zanimajo pri spremljanju posameznih aktivnostih (časovni roki, delo, izvajalci, omejitve). Zelo privlačna je tudi cena programske rešitve (nekaj pod $400, za študente in izobraževalne ustanove pa celo $50), kar omogoča dostopnost rešitve praktično vsakomur. Programska rešitev je še posebej primerna za začetnike s področja projektnega managementa, saj ponuja zelo dobro integriranost z okoljem Windows in ponuja način dela, ki je zelo podoben kot v ostalih programih paketa MS Office.

Nižje ocene pri ostalih kriterijih po predvsem posledica dejstva, da je rešitev namenjena delu enega samega uporabnika in ne podpira večuporabniškega projektnega okolja. Tako programske rešitve praktično ne moremo prilagodili lastnemu procesu izvajanja projektov, spremljamo pa lahko zgolj časovni del izvedbe projekta. Projektna poročila vezana na posamezni projekt so dobra. Pomembno je, da jih lahko shranjujemo v HTML obliki, kar omogoča dobro prenosljivost in njihovo enostavno objavo. Oceno poročil zmanjšuje dejstvo, da rešitev ne podpira multiprojektnega okolja in tako težko dobimo oceno o izvajanju projektov na nivoju celotnega podjetja.

4.3.2.2 e-Proj

Ocene glavnih kriterijev za programsko rešitev e-Proj so zbrane v spodnji tabeli (podroben opis ocen je v prilogi A.2):

Kriterij	Ocena
1. Prilagodljivost procesu vodenja projektov	5,6
2. Podpora časovnemu planiranju	3,3
3. Spremljanje izvedbe	5,5
4. Komunikacija med člani projektne skupine	7,3
5. Projektna poročila in baza znanja	4,0
6. Tehnična ustreznost rešitve	3,7
7. Cena	5,0

Tabela 4-15: Ocene glavnih kriterijev za programsko rešitev e-Proj

Programska rešitev je bila nadpovprečno ocenjena pri prilagajanju rešitve procesu vodenja projektov, spremljanju izvedbe, komunikaciji med člani projektnega skupine in ceni.
Še posebej dobro podporo nudi rešitev na področju komunikacije med člani projektne skupine. Ugotovitev vsekakor izvaja iz dejstva, da rešitev deluje v okolju Lotus Notes, ki je orodje za podporo skupinskemu delu. Zaradi primarnega okolja je rešitev tudi zelo dobra pri spremljanju projektne dokumentacije, delegiranju dokumentov, obveščanju o dogodkih na projektu in delu več uporabnikov na istih dokumentih. V ospredju so skratka funkcije, ki poudarjajo delo v večuporabniškem okolju in temeljijo na komunikaciji med člani projektne skupine.
Programska rešitev ponuja tudi veliko možnosti za podporo obstoječemu procesu projektnega vodenja v podjetju. Osnovna prilagoditev temelji na določanju vrst in tipov projektov, ki nastopajo v organizaciji. Delitev na vrste in tipe omogoča, da projekte multiprojktnega okolja ločimo po vsebini in obsegu (npr. marketinški projekti, razvojni projekti). Za vsako kombinacijo vrsta - tip nato določimo še ostale projektne parametre: privzet časovni plan, tipe dokumentov, ki nastopajo pri projektu, varnostno shemo, faze projekta in še nekatere druge parametre. Za posamezne tipe projektne dokumentacije lahko tudi predpišemo obliko. Na ta način lahko uvedemo standardno obliko določenih obrazcev, ki se pojavljajo pri poslovanju podjetja.
Nekoliko slabše ocene je dobila rešitev e-Proj pri časovnem planiranju in poročilih. Pomembno je poudariti, da so ocene časovnega planiranja določene na osnovi funkcionalnosti, ki jih ponuja sama rešitev e-Proj (ocena ni upoštevala možnosti

povezovanja z programom MS Project). Planiranje v rešitvi omogoča vnos aktivnosti, določanje virov in časovnih rokov, ne moremo pa določiti odvisnosti med aktivnosti in planirati v različnih načinih (npr. fiksno trajanje, delo). Za obvladovanje časovnega dela projekta pa rešitev e-Proj omogoča tudi povezavo z rešitvijo MS Project. Ta je bila ocenjena v prejšnjem poglavju. Če pri časovnem delu upoštevamo oceno za MS Project (8,0), to bistveno vpliva tudi na končno oceno.
Pri kriteriju poročil ponuja rešitev dobra vnaprej nastavljena poročila v multiprojektnem okolju. Manjka pa možnost, da bi si uporabnik sam oblikoval poročila in jih priredil svojim potrebam.
Rešitev je dobro prenosljiva med različnimi platformami, omejujoče je samo, da je njeno delovaje vezano na Domino strežnik. Zelo dobro je tudi, da rešitev omogoča delo preko spletnega brskalnika. Cenovno je rešitev zelo ugodna predvsem za okolja, ki že imajo vpeljano Lotus Notes okolje. To pomeni, da podjetje nima dodatnih stroškov z licencami Lotus okolja. Manjši so tudi stroški izobraževanja, saj so uporabniki že navajeni na delo v okolju.

4.3.2.3 Primavera TeamPlay

Ocene glavnih kriterijev za rešitev Primavera TeamPlay so zbrane v tabeli 4-17 (podroben opis ocen je v prilogi A.3):

Kriterij	Ocena
1. Prilagodljivost procesu vodenja projektov	8,2
2. Podpora časovnemu planiranju	9,0
3. Spremljanje izvedbe	8,8
4. Komunikacija med člani projektne skupine	5,7
5. Projektna poročila in baza znanja	9,0
6. Tehnična ustreznost rešitve	7,0
7. Cena	2,3

Tabela 4-17: Ocene glavnih kriterijev za programsko rešitev Primavera TeamPlay

Rešitev je dobila visoke ocene praktično v vseh kategorijah, nekoliko nižjo le pri komunikaciji med člani projektne skupine in ceni. Nižja ocena pri ceni seveda ne pomeni, da je rešitev precenjena, ampak da je predstavlja njena implementacija precejšen finančni zalogaj.
Rešitev Primavera TeamPlay omogoča zelo dobro prilagoditev procesu vodenja projektov v podjetju. Na nivoju podjetja lahko definiramo projektno strukturo, ki ima poljubno število nivojev, vsak od nivojev pa lahko vsebuje poljubno število projektov. Pri kreiranju projekta lahko na projekt povežemo veliko privzetih parametrov. S povezavo definiramo privzete aktivnosti, njihove rezultate, cilje in referenčne dokumente. Še posebej navdušuje podpora različnim projektnim metodologijam. Nekaj metodologij ponuja že orodje samo, npr. ISO 12207 Razvoj programskih rešitev, lahko pa kreiramo tudi svoje nastavitve, ki se skladajo s procesom obvladovanja projektov v podjetju.
Pri časovnem planiranju in spremljanju realizacije nam rešitev ponuja ogromno možnosti. Za vsako aktivnost lahko definiramo človeške in materialne vire, določimo trajanje in

planiramo po različnih metodah (fiksiramo lahko enega ali več od parametrov: trajanje, delo in zasedenost). Planiranje je podprto z gantogramom, mrežnim diagramom in grafičnim spremljanjem odvisnosti med aktivnostmi. Aktivnost lahko povežemo z referenčno dokumentacijo, problemi in tveganji. V primeru, da se projekt izvaja na več lokacijah, lahko aktivnosti dodelimo tudi kje je oddelek, ki izvaja aktivnost. Pri obsežnejših aktivnostih lahko definiramo korake izvedbe, jih utežimo in samostojno spremljamo.
Pri spremljanju aktivnosti lahko na WBS strukturo ali aktivnosti povežemo problemska stanja. Problemsko stanje ima definiran status, odgovorno osebo, rok izvedbe in opis stanja. Probleme lahko prožimo ročno, ali pa se kreirajo avtomatično iz prednastavljenih pogojev. Pri spremljanju projekta imajo uporabniki z različnimi vlogami na voljo različne preglede, vedno pa lahko izberemo katere projekte želimo pregledovati (od enega do vseh projektov v podjetju).
Nekoliko nižjo oceno je rešitev dobila pri kriteriju komunikacije med uporabniki. Uporabniki, ki uporabljajo rešitev preko spletnega dostopa, namreč ne morejo pošiljati elektronske pošte in delegirati dela drugim uporabnikom.
Pri izdelavi poročil nam je na voljo veliko število že vnaprej pripravljenih poročil (npr. stroški po projektih, odprti problemi po roku rešitve, zasedenost virov na naslednje časovno obdobje, tveganja po WBS strukturi …). Vsakega od poročil lahko uporabnik še ustrezno prilagodi lastnim potrebam. Vsa poročila lahko shranimo v HTML obliki, kot navadno besedilo ali jih direktno natisnemo. Posebej zanimiva je možnost objave projekta na spletni strani. Pred objavo izberemo katere poglede in podatke želimo vključiti, rešitev pa nam sama zgradi spletno stran projekta s podatki in navigacijo.
Skupno oceno rešitve nekoliko znižuje le cena rešitve. Ta znaša $50.000 za strežniško licenco in okoli $1000 za vsakega uporabnika (cena na uporabnika je nekoliko odvisna od števila uporabniških licenc). Vsekakor je glede na učinkovitost sistema cena primerna. Kar nekaj stroškov predstavlja tudi izobraževanje uporabnikov. Uporabniški vmesnik je ustrezno prilagojen delu v Windows okolju, zaradi kompleksnosti rešitve pa je potrebno kar nekaj časa, da se uporabnik dodobra seznani z delovanjem rešitve.

4.3.3 Primerjava preizkušenih rešitev

Pri primerjavi preizkušenih rešitev moramo vedno opredeliti pomembnost posameznih kriterijev. V prejšnjem poglavju (4.2) smo določili ocene za posamezne kriterije na absolutni lestvici od 1 do 9. Končna ocena rešitve pa je seveda odvisna še od tega, kako te kriterije ovrednotimo, oziroma kakšna razmerja postavimo med njimi. Skupno oceno rešitve dobimo tako, da ocene posameznih kriterijev pomnožimo z utežjo, potem pa otežene ocene med sabo seštejemo. Ker so utežni normirane (njihova skupna vsota je 1), je tudi končna ocena rešitve na skali med 1 in 9.
Pri primerjavi rešitev bomo upoštevali tri različne primere določanja prioritet za posamezne kriterije: enakomerno ovrednoteni kriteriji, kriteriji za potrebe majhnega podjetja, kriteriji za potrebe velikega podjetja.

4.3.3.1 Enakomerno ovrednoteni kriteriji

Ocene pri enakomerno ovrednotenih kriterijih so prikazane v tabeli 4-18 (do vrednosti v posameznih celicah pridemo z množenjem ocene kriterija iz poglavja 4.2 in uteži, ki je podana v tabeli):

Kriterij	*MS Project 2000*	*e-Proj*	*Primavera TeamPlay*	*Utež*
1. Prilagodljivost procesu vodenja projektov	0,1	0,8	1,2	14%
2. Podpora časovnemu planiranju	1,1	0,5	1,3	14%
3. Spremljanje izvedbe	0,3	0,8	1,3	14%
4. Komunikacija med člani projektne skupine	0,5	1,0	0,8	14%
5. Projektna poročila in baza znanja	0,5	0,6	1,3	14%
6. Tehnična ustreznost rešitve	0,3	0,5	1,0	14%
7. Cena	1,1	0,7	0,3	14%
Končna ocena programske rešitve	**3,9**	**4,9**	**7,1**	
Procent največje možne ocene	**44%**	**55%**	**79%**	

Tabela 4-18: Primerjava rešitev pri enakomerno ovrednotenih kriterijih.

V primeru enakomerno ovrednotenih kriterijev je najbolje ocenjena rešitev Primavera TeamPlay (79%), sledi e-Proj (55%) in MS Project 2000 (44%). Če za rešitev e-Proj pri kriteriju podpore časovne izvedbe upoštevamo povezavo z orodjem MS Project 2000 (ocena kriterija 8,0), dobimo končno oceno rešitve 5,6 (62%).

Najboljša ocena rešitve Primavera TeamPlay je posledica največje prilagodljivosti procesu vodenja projekta, najboljši podpori časovnemu planiranju in spremljanju izvedbe ter najbolj preglednih projektnih poročil. Rešitev predvsem odstopa pri podpori projektnim metodologijam, spremljanju problemov in tveganja na projektu ter prilagodljivi pregledih multiprojektnega okolja. Omogoča časovno planiranje virov, z osebnimi koledarji posameznikov in skupni seštevek obremenjenosti virov glede na različna časovna obdobja in projekte.

V primerjavi z ostalima rešitvama rešitvama e-Proj izstopa pri podpori komunikaciji med člani projektne skupine. Rešitev omogoča delegiranje nalog ostalim članom tima, avtomatično obveščanje ob posebnih dogodkih na projektu, beleženje zgodovine obveščanja, skupinsko delo več ljudi nad istimi dokumenti. Rešitev lahko sorazmerno dobro prilagodimo obstoječemu procesu vodenja projektov, dobra je tudi spremljava izvedbe. Spremljava omogoča povezovanje dokumentacije, ciljev in izdelkov s posameznimi aktivnostmi. Možno je tudi določiti poročila in tokove dokumentov potrebne za končanje posameznih aktivnosti, kontrolnih točk oziroma faz projekta. Glede na rešitev Primavera TeamPlay izgublja rešitev e-Proj predvsem pri časovnem planiranju in fleksibilnosti projektnih poročil v multiprojektnem okolju.

Najslabša ocena rešitve MS Project je posledica dejstva, da je rešitev namenjena samostojnim uporabnikom in da ne podpira multiprojektnega okolja. Neposredna posledica tega so slabše ocene pri prilagodljivosti projektnemu okolju, spremljanju izvedbe in poročilih multiprojektnega okolja. Dobra stran rešitve je podpora časovnemu planiranju in

cena rešitve. Časovno planiranje omogoča vse potrebne funkcionalnosti, cenovno pa je rešitev dostopna praktično vsakomur.

4.3.3.2 Kriteriji za majhno podjetje

V realnosti niso odločitveni kriteriji za izbiro projektnega informacijskega sistema praktično nikoli enakovredni. Za majhna podjetja je tako v ospredju predvsem možnost časovnega planiranja, spremljanja izvedbe in seveda cena. Ker pri projektu sodeluje majhno število ljudi in so ti večinoma na isti lokaciji, je manj pomembna podpora večuporabniškemu okolju. Udeleženi na projektu se medsebojno dobro poznajo in izmenjajo veliko informacij na delovnem mestu, zato v rešitvi ni nujna zelo dobra podpora komunikaciji med uporabniki. Poudarek je tako na časovnem planiranju, oceni stroškov in spremljanju, katere aktivnosti so že končane in kako je z izvedbo aktivnosti, ki so trenutno v teku. Glavni faktor pri odločitvi majhnega podjetja za programsko rešitev za vodenje projektov pa je praviloma cena programske rešitve, stroški uvedbe, izobraževanja in vzdrževanja. Če vse našteto ustrezno ovrednotimo, dobimo ocene prikazane v tabeli 4-19:

Kriterij	*MS Project 2000*	*e-Proj*	*Primavera TeamPlay*	*Utež*
1. Prilagodljivost procesu vodenja projektov	0,0	0,1	0,1	1%
2. Podpora časovnemu planiranju	2,4	1,0	2,7	30%
3. Spremljanje izvedbe	0,2	0,6	0,9	10%
4. Komunikacija med člani projektne skupine	0,3	0,7	0,6	1%
5. Projektna poročila in baza znanja	0,2	0,3	0,6	7%
6. Tehnična ustreznost rešitve	0,0	0,0	0,1	1%
7. Cena	3,8	2,5	1,2	50%
Končna ocena programske rešitve	**7,0**	**5,1**	**6,1**	
Procent največje možne ocene	**78%**	**57%**	**68%**	

Tabela 4-19: Primerjava rešitev: Kriteriji za majhno podjetje.

Za primer majhnega podjetja, ki zaposluje nekaj zaposlenih in obvladuje naenkrat le nekaj projektov, je najboljšo oceno dobila rešitev Microsoft Project 2000 (78%), sledi Pimavera Teamplay (68%) in e-Proj (57%). Glede na opisano okolje, je rezultat je pričakovan, saj je rešitev MS Project namenjena predvsem uporabi v ožjem krogu sodelavcev in je cenovno najugodnejša. Za majhno podjetje uvedba kompletnega informacijskega sistema kot je Primavera TeamPlay seveda ni smotrna, saj samo strežniška licenca tega programa za več kot 100 krat presega stroške rešitve MS Project. Podjetje, ki uporablja rešitev MS Project kot osnovo za spremljanje projektov, mora upoštevati faktorje zaradi katerih lahko postane rešitev nezadostna. Nekaj teh faktorjev je povečanje števila zaposlenih, večje število projektov, različne lokacije zaposlenih, želja po vključevanju zunanjih sodelavcev za delo na projektu, ipd.

Nekoliko preseneča, da se je kljub večji ceni na drugo mesto uvrstila rešitev Primavera TeamPlay. Razlog je predvsem v kriteriju za časovno planiranje, kjer je rešitev Primavera TeamPlay ocenjena precej bolje od rešitve e-Proj. Če za rešitev e-Proj v tem kriteriju upoštevajo integracijo z rešitvijo MS Project, rešitev e-Proj dobi skupno oceno 73% in se

po pričakovanju uvrsti na drugo mesto. Drugi razlog zakaj je rešitev Primavera TeamPlay pred rešitvijo e-Proj pa je tudi, da z oteženo vrednostno lestvico ne moremo določiti izločitvenih kriterijev. Tako recimo ne moremo določiti omejitve, da je določena cena uvedbe rešitve izločitveni faktor. Stroški uvedbe rešitve e-Proj so namreč vsaj 10-krat nižji od uvedbe rešitve Primavera TeamPlay.

4.3.3.3 Kriteriji za večja podjetja

Za razliko od projektov v majhnih podjetjih, je pri projektih v velikih podjetjih udeleženo veliko zaposlenih, ti niso nujno zaposleni na isti lokaciji, komunikacija med člani projekta in projektnim vodjem je manjša ipd. Takšno okolje seveda zahteva popolnoma drugačen informacijski sitem za vodenje projektov. V ospredju so prilagodljivost rešitve obstoječemu okolju, komunikacija med člani tima, spremljanje izvedbe in projektna poročila. V tem primeru je cena uvedbe projekta manj pomembna. V primeru uspešne uvedbe je strošek uvedbe majhen v primerjavi s podporo procesu in podatki, ki jih podjetje na ta način pridobi. V spodnji tabeli so prikazani dobljeni rezultati pri oteženi vsoti za primer velikega podjetja:

Kriterij	*MS Project 2000*	*e-Proj*	*Primavera TeamPlay*	*Utež*
1. Prilagodljivost procesu vodenja projektov	0,1	1,1	1,6	20%
2. Podpora časovnemu planiranju	1,3	0,5	1,4	16%
3. Spremljanje izvedbe	0,4	1,0	1,6	18%
4. Komunikacija med člani projektne skupine	0,6	1,3	1,0	20%
5. Projektna poročila in baza znanja	0,6	0,6	1,4	16%
6. Tehnična ustreznost rešitve	0,1	0,2	0,4	5%
7. Cena	0,4	0,3	0,1	5%
Končna ocena programske rešitve	**3,4**	**5,0**	**7,6**	
Procent največje možne ocene	**38%**	**56%**	**84%**	

Tabela 4-20: Primerjava rešitev: Kriteriji za veliko podjetje.

Dobljen vrstni red je isti kot pri enakomerno ovrednotenih kriterijih, le da so se razlike med rešitvami še povečale. Največji procent je pripadel rešitvi Primavera TeamPlay (84%), sledi e-Proj (56%) in na koncu MS Project (38%).

Dobljeni rezultati so pričakovani. Vrstni red rešitev se sklada s tistim pri enakomerni porazdelitvi, le da je rešitev Primavera TeamPlay še pridobila na skupni oceni, rešitev MS Project pa še izgubila nekaj dodatnih procentov. Očitno je, da bolj kot je zahtevno projektno okolje, bolj pridejo o izraza kvalitete programske rešitve Primavera TeamPlay. Prilagodljivost, spremljaje izvedbe, poročila in skupinsko delo so najpomembnejše kvalite apliakcije.

Solidno oceno si zasluži tudi rešitev e-Proj (nad 50%). Primerna je predvsem za okolja, kjer je v ospredju komunikacija med uporabniki, tokovi podatkov in projektna dokumentacija. Glavna prednost rešitve e-Proj pred Primavero je vsekakor cena, ki je lahko ključni faktor pri izbiri projektnega informacijskega sistema.

Rešitev MS Project za zahtevna okolja z veliko uporabniki ni primerna (manj kot 40%). Ugotovitev je logična, saj rešitev ni namenjena večuporabniškemu multiprojektnemu okolju. Za kompleksnejša okolja je isti proizvajalec namenil rešitev Project Central, s katero želi zadovoljiti tudi potrebe kompleksnega projektnega okolja.

5 Zaključek

5.1 Ugotovitve magistrskega dela

Projekti s področja informacijske tehnologije in razvoja programskih rešitev postavljajo projektnemu managementu velik izziv. Razvoj programske opreme poteka v dinamičnem, hitro spreminjajočem se okolju, brez dobro definiranih industrijskih standardov in brez zanesljivih kvalitativnih podatkov iz sorodnih projektov. Tehnologija se praktično spreminja iz dneva v dan, tako da panoga zahteva nenehno sledenje spremembam in izobraževanje zaposlenih. Projektni pristop pri razvoju programskih rešitev pomeni delo v dinamičnem okolju, izraža potrebe po visoki učinkovitosti, zahteva optimalno izrabo kadrov in hiter pogled v tekoče stanje del na projektu.

Glede na teoretične osnove s področja projektnega managementa pri razvoju programskih rešitev in praktičnih izkušenj, je v delu izpostavljenih 5 glavnih dejavnikov neuspeha projektov s področja razvoja programskih rešitev:

- nerealen plan,
- nedefiniran proces razvoja,
- slabo opredeljene uporabniške zahteve,
- neustrezno testiranje
- slaba komunikacija.

Če je pri projektu navzoča katerakoli od naštetih težav, je velika verjetnost, da razvita programska rešitev ne bo dosegla želenih rezultatov oziroma, da projekt ne bo uspešen. Za vsako od naštetih težav so nam na voljo različne metode in orodja, ki jih prav tako lahko združimo v pet sklopov (našteti sklopi predstavljajo odgovore na zgoraj naštete težave):

- obvladovanje časa in stroškov,
- uporaba razvojnega modela,
- obvladovanje uporabniških zahtev,
- testiranje in zagotavljanje kvalitete,
- programska rešitev za vodenje projektov.

Osnova za obvladovanje časa in stroškov pri projektih razvoja informacijskih rešitev, je ocena dela in trajanja projektnih aktivnosti. Ocena je težavna zaradi: nasprotujočih si ciljev projekta, pomanjkanja podrobnega opisa funkcionalnosti, potrebe po uporabi novih tehnologij, itd. Ključni problem ocenjevanja je, da je ocena potrebna v zgodnji fazi projekta, ko produkt še ni dobro definiran. Tehnike ocenjevanja trajanja in stroškov projekta lahko razdelimo v več skupin: algoritmični modeli, ekspertne ocene, analogija iz sorodnih projektov, pristop od vrha proti dnu in pristop od dna proti vrhu. V praksi je najbolj uporabljan algoritmični model COCOMO (Constructive COst MOdel). Model uporablja za osnovo ocene število vrstic izvorne kode, na oceno pa vplivajo tudi korekcijski faktorji, npr. zahtevana zanesljivost rešitve, velikost podatkovne baze, kompleksnost produkta, izkušnje programerjev in drugi. Trenutno je v uporabi verzija

modela COCOMO 2.0, ki pri oceni upošteva tudi uporabo razvojnih modelov, ponovno uporabljivost programskih modulov in objektno orientirano programiranje.
Razvojni model je skupek aktivnosti, ki omogočajo prehod od naročnikovih zahtev do delujoče programske rešitve. Žal v praksi ne obstaja edinstven razvojni model, ki bi bil primeren za vsa razvojna okolja. Kateri model izbrati je odvisno od področja apliakcije, tipa organizacije in zahtevnosti rešitve. Pionirsko vlogo med modeli razvoja programskih rešitev je odigral kaskadni oz. Waterfall razvojni model. Faze modela si sledijo zaporedno in se med seboj ne prekrivajo.Pogoj za uspešnost so dobre znane programske zahteve v zgodnji fazi projekta, kar je v praksi zelo težko izvedljivo. Težave kaskadnega modela odpravljata evolucijski in spiralni razvojni model. Bistvo evolucijskega modela je uporaba ciklov. Vsak cikel se ukvarja z malim segmentov projekta, cikli pa hitro sledijo eden drugemu. Spiralni razvojni model temelji na identifikaciji in obvladovanju tveganja na projektu. Tudi spiralni model je sestavljen iz več ciklov, ki imajo predpisano zaporedje korakov. Vsak cikel se začne z ugotavljanjem ciljev, sledi ocena tveganja, razvoj in verifikacija ter načrtovanje naslednjega cikla. V zadnjem času je zelo popularen Rational Unified Process (RUP), ki združuje najboljše primere iz industrijske prakse. Model predvideva štiri faze: ugotovitev zahtev, izdelava načrta, izgradnja in prehod v produkcijo. Vsaka od faz ima več iteracij in delovnih tokov: podatkovno modeliranje, definiranje zahtev, analiza in načrt, izgradnja, testiranje in prehod v produkcijo.Model je dobro podrt s programsko opremo, ki podpira jezik UML.
Uporabniške zahteve predstavljajo podroben opis funkcionalnosti in lastnosti programske rešitve. Z obvladovanjem zahtev se ukvarja konfiguracijski management, ki podpira tudi obvladovanje sprememb programske kode, dokumentiranje procesov in zaznavanje problemov.Glavna področja delovanja konfiguracijskega managementa so razvojno okolje, programska koda, obvladovanje različic programske kode, verziranje in izgradnja rešitve.
Testiranje programskih rešitev je namenjeno preventivi in odkrivanju napak. Pri testiranju nastopajo posamezniki z različnimi vlogami, npr. vodja testne skupine, vsebinski vodja testa, tehnični vodja testa in preizkuševalci. Testiranje poteka na različnih nivojih, po končanem testiranju posameznega nivoja pa je potrebno potrebo uskladiti rezultate testa s procesom konfiguracijskega managementa. Proces testiranja se sestoji iz določanja strategije testa, načrtovanja in izvedbe.
Ne glede na to kakšen projektni pristop uporablja podjetje, potrebuje za uspešno vodenje projekta ustrezen informacijski sitem. Obstaja več kategorij programskih rešitev za vodenje projektov, tako recimo orodja za planiranje in oceno dela, za spremljanje projekta, komunikacijska orodja, orodja za obvladovanje tveganja in kontrolo na projektu. Obstajajo seveda tudi orodja, ki združujejo več od naštetih kategorij. Za podjetje je seveda ključno vprašanje katerega od številnih orodij, ki nastopajo na trgu, izbrati. Pri izvedbi je najbolje upoštevati naslednje korake: razumevanje lastnega procesa vodenja projektov, priprava pisnih specifikacij, priprava modela za ocenjevanje, izvedba ocene in izbira orodja.
V delu je razvit in opisan model za ocenjevanje informacijskih sistemov za vodenje projektov. Odločitveno drevo vsebuje 7 glavnih kriterijev:

- prilagodljivost procesu vodenja projektov,
- podpora časovnemu planiranju,
- spremljanje izvedbe,
- komunikacija med člani projektne skupine,
- projektna poročila in baza znanja,
- tehnična ustreznost rešitve,
- cena.

Vsak od kriterijev je naprej razčlenjen na več podkriterijev. Vsakega od podkrieterijev ocenjujemo z ocenami od 0 do 9. Podkriteriji so ustrezno uteženi, uteži so normirane (njihova vsota je 1), kar pomeni, da je tudi končna ocena kriterija med 0 in 9. Pri računanju končne ocene programske rešitve se postopek izračuna ponovi, tokrat ne enem nivoju višje. Take je tudi končna ocena produkta nekje med 0 in 9. V delu so posamezni kriteriji odločitvenega drevesa podrobno opisani, skupaj z navodili za določanje ocene posameznih kriterijev. V okviru naloge je narejena tudi računalniška podpora odločitvenemu drevesu za izbiro ustreznega informacijskega sistema (Excelova datoteka je priložena nalogi).V delu so bile ocenjene tri rešitve za vodenje projektov:Microsoft Project 2000, e-Proj in Primavera TeamPlay.

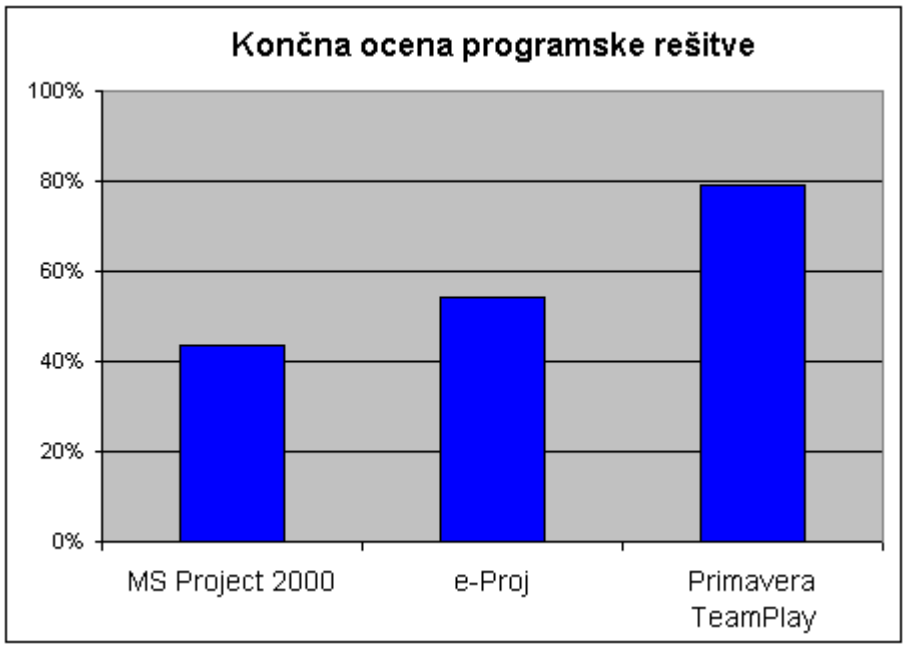
Programska rešitev Microsoft Project 2000 je produkt podjetja Microsoft Corp. Funkcije orodja so načrtovanje, časovno planiranje in spremljanje stroškov ter virov projekta. Namenjeno je delu na samostojnem osebnem računalniku in je primerno za začetnike in izkušene končne uporabnike. Orodje zavzema vodilno mesto v kategoriji projektnih rešitev namenjenih uporabi na enem osebnem računalniku.

Rešitev e-Proj je programska rešitev podjetja Genis d.o.o.Rešitev omogoča boljše obvladovanje posameznega projekta ter vodenje in usklajevanje dejavnosti z drugimi projekti. Cilji uporabe rešitve so zagotoviti preglednost postopkov in projektne dokumentacije, omogočiti organizirano vodenje in spremljanje nalog, vzpostaviti skupno bazo o projektih, zagotoviti izgradnjo baze znanja in povečati motiviranost zaposlenih na projektu. Rešitev je zgrajena iz več medsebojno povezanih modulov: projektni elaborat, časovni plan, dokumentacija, poročila, stroški in kartica projekta.

Primavera TeamPlay je programska rešitev podjetja Primavera Systems, Inc. Glavni cilji programske rešitve so najbolj izkoristiti vložena sredstva, povečati preglednost projektnih podatkov, povečati hitrost in učinkovitost izvedbe in zagotoviti ponovljivost. Rešitev omogoča, da so cilji projektov v skladu s strategijo projekta in da je dodana vrednost pri projektu največja. Orodje je vodilno v kategoriji projektnih informacijskih sistemov najvišjega razreda, ki podpirajo projektne zahteve celotnega podjetja.

Vse rešitve so bile preizkušene v namišljenem projektnem okolju. Testno podjetje je bilo srednje veliko podjetje, ki se ukvarja z razvojem programskih rešitev. Implementirani so bili trije projektni z različno vsebino; projekt razvoja nove programske rešitve, projekt uvedbe produkta in vzdrževalni projekt. Vse preizkušene rešitve so bile ocenjene v vseh definiranih kriterijih, ocene pa zbrane v prilogi A.

Pri primerjavi preizkušenih programskih rešitev moramo vedno opredeliti pomembnost posameznih kriterijev. Najbolj celostna primerjava je primerjava, pri kateri so enakovredno ovrednoteni vsi kriteriji. Pri enakomerno ovrednotenih kriterijih dobimo skupne ocene, kot jih prikazuje spodnji graf:



Slika 5-1: Primerjava preizkušenih rešitev

Pri enakomerno ovrednotenih kriterijih je najbolje ocenjena rešitev Primavera TeamPlay (79%), sledi e-Proj (55%) in MS Project 2000 (44%). Najboljša ocena rešitve Primavera TeamPlay je posledica največje prilagodljivosti procesu vodenja projektov, najboljši podpori časovnemu planiranju in spremljanju izvedbe ter najbolj preglednih projektnih poročil. Rešitev e-Proj izstopa pri podpori komunikaciji med člani projektne skupine, omogoča delegiranje nalog, avtomatično obveščanje ob posebnih dogodkih in podpira skupinsko delo. Najslabša ocena rešitve MS Project je posledica dejstva, da je rešitev namenjena samostojnim uporabnikom in da ne podpira večuporabniškega multiprojektnega okolja.

V dejanskem okolju niso odločitveni kriteriji za izbiro projektnega informacijskega sistema praktično nikoli enakovredni. Za majhna podjetja je tako v ospredju predvsem možnost časovnega planiranja, spremljanja izvedbe in seveda cena. Pri takšnih kriterijih se razmerja med preizkušenimi rešitvami seveda precej spremenijo. V primeru majhnega podjetja je tako najbolj ustrezna rešitev MS Project 2000 (78%), sledita pa Primavera TeamPlay (68%) in e-Proj (57%).

Za razliko od majhnih podjetij je pri projektih v velikih podjetjih udeleženo veliko zaposlenih, ti niso nujno zaposleni na isti lokaciji, komunikacija med člani projekta in projektnih vodjem je majhna ipd. Za takšno okolje so v ospredju prilagodljivost rešitve obstoječemu okolju, komunikacija med člani tima, spremljanje izvedbe in projektna poročila. Dobljen vrstni red je isti kot pri enakomerno ovrednotenih kriterijih, le da so se razlike med rešitvami še povečale. Največjo oceno je dobila rešitev Primavera TeamPlay

(84%), sledi e-Proj (56%) in na koncu MS Project (38%). Očitno je, da bolj kot je zahtevno projektno okolje, bolj pridejo do izraza kvalitete programske rešitve Primavera TeamPlay.

5.2 Analiza zastavljenih ciljev

Prvi glavni cilj magistrskega dela, je bila ugotovitev ključnih dejavnikov uspeha projektov s področja razvoja programskih rešitev. V delu so bili ugotovljeni naslednji ključni dejavniki:

- obvladovanje časa in stroškov,
- uporaba razvojnega modela,
- obvladovanje uporabniških zahtev,
- testiranje in zagotavljanje kvalitete,
- programska rešitev za vodenje projektov.

Našteti dejavniki odločilno vplivajo na uspešnost projektov razvoja programskih rešitev. Glede na raziskavo svetovalnega podjetja KPMG, znaša verjetnost za uspeh projekta, ki bo zadovoljil naštete kriterije, 86%.

V magistrskem delu je opisan način zagotavljanja ključnih dejavnikov uspeha. Pri obvladovanju časa in stroškov so predstavljene metode časovnega planiranja (npr. metoda kritične poti) in različni modeli za oceno stroškov (npr. COCOMO). V delu je opisanih nekaj različnih razvojnih modelov (npr. kaskadni, spiralni, evolucijski), poudarjene so njihove prednosti in slabosti ter primernost za različna razvojna okolja. Za obvladovanje uporabniških zahtev je v delu opisan konfiguracijski management, predstavljen pa je tudi celoten proces testiranja programskih rešitev.

Drugi glavni cilj magistrskega dela je bil opredeliti pomen ustrezne programske rešitve za vodenje projektov in določiti kako naj se podjetje odloči med številnimi ponudniki na trgu. Programska rešitev za vodenje projektov podjetju omogoča optimizirati vire, izboljša komunikacijo med člani projektnega tima, poveča preglednost projektnih podatkov ter omogoča kontrolo nad tveganji in problemi. Je osnova za merljivost projektih podatkov, določanje uspešnosti posameznikov udeleženih pri projektu in omogoča podjetju izboljšati lastno poslovanje.

Kot pomoč podjetjem pri izbiri ustrezne rešitve, je bil v delu razvit ocenjevalni model za primernost različnih rešitev za vodenje projektov. Z modelom so bile preizkušene tri programske rešitve MS Project 2000, e-Proj in Primavera TeamPlay. Pred oceno in ovrednotenjem posameznih rešitev, je bilo pričakovati, da bo rešitev MS Project najbolj primerna za nezahtevna projekta okolja, rešitev Primavera TeamPlay pa za zelo zahtevna, večuporabniška projektna okolja. Rezultati preizkusa so pričakovanja potrdili in postavili še razmerja med preizkušenimi rešitvami. Za primer majhnega podjetja (nezahtevno projektni okolje) je bila najuspešnejša programska rešitev MS Project 2000 (78% najvišje možne ocene), za zahteva projektna okolja pa rešitev Primavera TeamPlay (84%).

Poleg dveh glavnih ciljev, so bili v delu doseženi tudi ostali cilji definirani v poglavju 1.2.

5.3 Kako naprej ?

Proces razvoja programskih rešitev je podvržen silovitim tehničnim spremembam in spremembam v načinu razvoja programske opreme. Glede na trenutno stanje pri razvoju programskih rešitev, so najbolj pogosti vzroki za neuspeh, nezmožnost obvladovanja rokov in stroškov, slabo obvladovanje uporabniških zahtev, pomanjkanje ustreznega razvojnega modela in nezadostno testiranje. V bližnji prihodnosti sicer ni pričakovati, da bi se ti vzroki drastično spremenili, zagotovo pa se bo spreminjala njihova pomembnost in porajali se bodo vedno novi vzroki.

V prihodnje lahko računajo na uspeh pri razvoju programskih rešitev predvsem podjetja, ki imajo dobro definiran proces razvoja programskih rešitev in zelo dobro poznajo svoj način dela in trženja rešitev. Dobro definiran razvojni proces omogoča podjetju ponovljivost postopkov, ponovno uporabo programske kode in precejšnjo neodvisnost pred begom zaposlenih. S ponovno uporabo kode se drastično zmanjša število programskih napak, stroški za testiranje modulov, poveča pa se tudi zadovoljstvo zaposlenih, ki jim ni potrebno vedno znova razvijati »istih« programskih modulov. Seveda pa je pogoj za takšno delo precej široko razmišljanje v fazi načrtovanja programskih rešitev in usmerjenost podjetja za uspeh na daljši časovni rok.

Drugi ključni segment pri razvoju programskih rešitev ostaja model ocenjevanja rokov in stroškov projekta. Oceno je potrebno podati v zgodnji fazi projekta, ko uporabniške zahteve še niso dobro definirane. Trenutno je ocena stroškov in trajanja projekta v veliki meri domena ekspertne ocene. Podjetja, ki bodo razvila ustrezne modele za oceno rokov in stroškov projekta bodo vsekakor imela konkurenčno prednost na trgu, saj bodo lažje in uspešneje izpolnila pričakovanja in želje naročnikov.

Ne glede na vse spremembe pri razvoju programskih rešitev, bo informacijska podpora procesu razvoja dobivala vedno večji pomen. Ustrezna programska rešitev za vodenje projektov, je osnova za merljivost projektnih rezultatov, analizo napak, učenje o projektnem načinu dela v podjetju in omogoča nenehno izboljševanje razvojnega procesa. Na trgu se bo pojavljalo vedno več ponudnikov rešitev, tako da bo izbira ustrezne rešitve vedno težja. V tem pogledu je za podjetje zelo pomembno, definirati ocenjevalno metodo, ki bo z določeno verjetnostjo omogočala podjetju, da se odlogi za najbolj primerno programsko rešitev.

Primer takšnega modela je bil razvit tudi v okviru tega magistrskega dela. Trenutno vsebuje model 7 glavnih kriterijev in 26 podkriterijev, v delu pa je bil preizkušen na 3 programskih orodjih. Za preizkušena programska orodja je model postregel s pričakovanimi rezultati, vendar je statistična množica še mnogo premajhna za resnejšo oceno razvitega modela.

V prihodnje bi bilo potrebno model preizkusiti na večjem številu programskih rešitev za vodenje projektov in rezultate ustrezno statistično obdelati.Obsežnejša analiza bi nam omogočila tudi ustreznejšo ovrednotene posameznih podkriterijev, oziroma poudarila njihovo pomembnost na končno oceno programske rešitve. S takšno analizo bi postal

model zanesljivejši, bolj natančen in s tem tudi uporabnejši pri izbiri ustrezne programske rešitve za vodenje projektov.

Zavedati se tudi moramo, da so kriteriji za oceno rešitev prilagojeni na trenutno stanje programske opreme za vodenje projektov. Zaradi hitrih sprememb na področju informacijske tehnologije, je pričakovati, da bodo kriteriji podvrženi nenehnim spremembam. Najverjetneje bo vedno pomembnejši kriterij dostopnost projektnih podatkov preko interneta in drugih medijev (mobilni telefoni, dlančniki, ...). Vedno večji poudarek pri orodjih pa bo tudi povezljivost projektnih podatkov z ostalimi segmenti poslovanja in možnost izmenjave projektnih podatkov med različnimi podjetji (B2B).

Priloga A: Ocena preizkušenih rešitev po posameznih kriterijih

V prilogi A so predstavljene ocene preizkušenih programskih rešitev po posameznih kriterijih, skupaj z opisom ocen.

5.4 A.1. MS Proejct 2000

Kriterij in opis ocene	Ocena
1. Prilagodljivost procesu vodenja projektov	**0,6**
1.1 Nastavljive faze projekta Spremljamo lahko načrtovane in dejanske projektne podatke. Celotnega procesa ne moremo implementirati, možna je samo primerjava med planiranjem in izvedbo	3
1.2. Podpora različnim tipom projektov Funkcija ni podprta.	0
1.3. Nastavljivi tipi projektne dokumentacije Funkcija ni podprta.	0
1.4. Podpora privzetim parametrom Funkcija ni podprta.	0
1.5. Podpora projektnim metodologijam Funkcija ni podprta.	0
2. Podpora časovnemu planiranju	**8,0**
2.1. Strukturirano definiranje aktivnosti Aktivnosti lahko naštejemo, grupiramo in določimo odvisnosti med njimi.	9
2.2. Načrtovanje trajanja in dela na aktivnostih Možnost definiranja človeških virov (materialnih težje), planiranje po različnih kriterijih in dodajanje omejitev za aktivnost. Manjše težave pri določanju fiksnih količin na aktivnosti (fiksiramo lahko samo eno od količin delo, trajanje in obremenitev virov)	7
2.3. Podpora grafičnim tehnikam Prikaz podatkov v gantogramu, mrežnem diagramu, grafična spremljava plana in izvedbe.	7
2.4. Podpora tehnikam časovnega planiranja Rešitev omogoča izračun in grafični prikaz kritične poti, spremljanje dejanskega izvajanja aktivnosti in PERT analizo.	9
3. Spremljanje izvedbe	**2,0**
3.1. Spremljanje statusa aktivnosti Za vsako aktivnost, lahko določimo koliko dela je opravljenega in primerjamo načrtovane in dejanske datume začetka in konca aktivnosti.	3
3.2. Podpora projektni dokumentaciji Za vsako aktivnost lahko dodamo komentar, ki pa je omejen na enostavno tekstovno sporočilo.	1
3.3. Pregled problemskih stanj Funkcija ni podprta.	0
3.4. Preglednost podatkov Aktivnosti lahko filtriramo po različnih kriterijih, jih grupiramo, možno je spremljanje obremenitve zaposlenih po posameznikih in dnevih.	4
4. Komunikacija med člani projektne skupine	**3,3**

4.1. Določene elektronske pošte uporabnikov Za posamezne vire lahko določimo njihove elektronske naslove (povezava je povezana na uporabo programa Outlook Express).	5
4.2. Ročno obveščanje o dogodkih Posameznikom lahko pošljemo elektronsko pošto, ni pa nobenega beleženja zgodovine poslanih sporočil.	5
4.3. Avtomatično obveščanje Funkcija ni podprta.	0
5. Projektna poročila in baza znanja	**3,5**
5.1. Prednastavljena poročila Rešitev podpira različna poročila v okviru projekta, ki jih lahko shranimo tudi v HTML obliki. Ne obstaja neposredna povezava med poročili, izdelki in izvajanjem projekta.	5
5.2. Poročila na zahtevo uporabnika Poleg prednastavljenih poročil, lahko uporabnik sam oblikuje katere podatke želi vključiti v poročilo in jih izvozi v HTML obliki.	7
5.3. Poročila multiprojektnega okolja Možno je spremljanje virov skozi več projektov, vendar je funkcija nezadostno implementirana. Pravo multiprojetno okolje ne obstaja.	1
5.4. Baza znanja Beleženje zgodovine je mogoče s shranjevanjem starih MS Project datotek.	1
6. Tehnična ustreznost rešitve	**2,3**
6.1. Podpora različnim platformam Rešitev temelji na operacijskem sistemu Windows. Glede na namen rešitve (uporaba na enem osebnem računalniku), je omejenost še nekako sprejemljiva.	3
6.2. Način dostopa končnih uporabnikov Uporabniki morajo imeti instalirano rešitev MS Project 2000. Rešitev ne podpira večuporabniškega okolja.	3
6.3. Podpora različnim relacijskim bazam Rešitev omogoča shranjevanje podatkov tudi MS Access relacijski podatkovni bazi.	1
7. Cena	**7,7**
7.1. Cena osnovnega paketa Cena programske rešitve je zelo sprejemljiva.	9
7.2. Cena nadgradnje posameznih modulov Rešitev podpira programski jezik VBA, zato je nadgradnja načeloma mogoča, ampak zelo omejena in ni podprta s strani proizvajalca.	5
7.3. Cena izobraževanja končnih uporabnikov Uporabniško izobraževanje ne predstavlja večjih stroškov, saj je rešitev zelo usklajena z delom v ostalih MS Office rešitvah.	9

Tabela A-1: Ocene programske rešitve MS Project 2000

5.5 A.2. e-Proj

Kriterij in opis ocene	**Ocena**
1. Prilagodljivost procesu vodenja projektov	**5,6**
1.1 Nastavljive faze projekta Možno je nastaviti projektne faze in prehode med njimi.	7
1.2. Podpora različnim tipom projektov	7

V rešitvi lahko nastavimo katere vrste in tipe projektov spremljamo.	
1.3. Nastavljivi tipi projektne dokumentacije Možna je večnivojska nastavitev projektne dokumentacije, ki jo lahko povežemo tudi na posamezne tipe in vrste projekta.	8
1.4. Podpora privzetim parametrom Glede na vrsto in tip projekta lahko definiramo privzet časovni plan, varnostno shemo in skrbnika rešitve.	6
1.5. Podpora projektnim metodologijam Funkcija ni podprta.	0
2. Podpora časovnemu planiranju	**3,2**
2.1. Strukturirano definiranje aktivnosti Aktivnosti lahko naštejemo, ne moremo pa jih medsebojno strukturirati (brez uporabe povezave z rešitvijo MS Project)	5
2.2. Načrtovanje trajanja in dela na aktivnostih Lahko definiramo odgovorno osebo in izvajalce, trajanje, ne moremo pa planirati aktivnosti z različnimi omejitvami (npr. aktivnost se me more začeti pred)	5
2.3. Podpora grafičnim tehnikam Rešitev ponuja zelo grob ganrogram (po mesecih), ki mu ne moremo definirati spreminjati časovne skale.	3
2.4. Podpora tehnikam časovnega planiranja Funkcija ni podprta.	0
3. Spremljanje izvedbe	**5,5**
3.1. Spremljanje statusa aktivnosti Za vsako aktivnost lahko spremljamo status, opravljeno delo izvajalcev, časovni in izvedbeni del realizacije, predvidene rezultate in izdelke.	6
3.2. Podpora projektni dokumentaciji Projektno dokumentacijo lahko dodajamo na projekt ali posamezne projektne aktivnosti. Možen je strukturiran pregled in tipizacija dokumentacije.	8
3.3. Pregled problemskih stanj Problemska stanja lahko definiramo kot poseben tip dokumentacije. Rešitev ne določa posebne programske logike za spremljanje problemskih stanj.	1
3.4. Preglednost podatkov Posamezniki z različnimi vlogami imajo na voljo različne preglede nad podatki. Možno je spremljanje samo aktivnosti na katerih posameznik dela oziroma za katere je odgovoren.	7
4. Komunikacija med člani projektne skupine	**7,3**
4.1. Določene elektronske pošte uporabnikov Če je elektronska pošta podprta z Domino strežnikom, je povezava med uporabniki in elektronskimi naslovi definirana že z okoljem samim. V primeru uporabe drugega poštnega strežnika, lahko povežemo uporabnike rešitve in elektronske naslove.	7
4.2. Ročno obveščanje o dogodkih Za vsak del projekta je možno ročno obveščanje uporabnikov. Beleži se zgodovina poslanih obvestil. Možno je tudi delegirati izdelavo posameznih dokumentov in potrjevanje preko elektronske pošte.	8
4.3. Avtomatično obveščanje V programski rešitvi je možno nastavljati avtomatično obveščanje ob različnih dogodkih (zamujanje aktivnosti, presežen obseg dela, preobremenjenost virov...). Izboljšan bi lahko bil uporabniški vmesnik za naslavljanje alarmov.	7
5. Projektna poročila in baza znanja	**4,0**

5.1. Prednastavljena poročila Rešitev ponuja obrazec za periodična in mesečna poročila na aktivnostih in projektu. Kartica projekta predstavlja sumarno poročilo za projekt. Poročila omogočajo pregled nad stanjem, bi pa lahko šla bolj v globino.	4
5.2. Poročila na zahtevo uporabnika Uporabnik je v veliki meri omejen na prednastvljene preglede in poročila. Ta so sicer pregledna, a ne preveč fleksibilna.	3
5.3. Poročila multiprojektnega okolja Rešitev ponuja nazoren prikaz stanja v multiprojektnem okolju, s statusi projektov, dokumentacijo in oceno stanja. Spet pogrešamo nastavljivost poročil.	5
5.4. Baza znanja Del rešitve je tudi arhivska baza, ki ponuja iste preglede in poročila kot baza aktivnih projektov. Dobro je, da arhiv ohranja strukturo in preglede kot produkcijska baza.	4
6. Tehnična ustreznost rešitve	**3,7**
6.1. Podpora različnim platformam Rešitev je vezana na Domino strežnik. Prednost je, da Domino strežnik podpira široko paleto operacijskih sistemov, pomanjkljivost pa, da potrebujemo Domino strežnik za delovanje rešitve.	5
6.2. Način dostopa končnih uporabnikov Končni uporabniki lahko dostopajo do rešitve z Lotus Notes odjemalcem ali pa preko spletnega brskalnika. Manjši minus je, da preko spletnega brskalnika še niso podprte čisto vse funkcije.	6
6.3. Podpora različnim relacijskim bazam Funkcija v rešitvi ni podprta.	0
7. Cena	**5,0**
7.1. Cena osnovnega paketa Glede na ponujene funkcionalnosti je cena programske rešitve zelo ugodna, sploh za podjetja, ki že uporabljajo Lotus Notes okolje	5
7.2. Cena nadgradnje posameznih modulov Razširljivost rešitve je dokaj enostavna, poveča pa stroške vzdrževanja in nadaljnjih nadgradenj programske rešitve.	5
7.3. Cena izobraževanja končnih uporabnikov Cena je v veliki meri odvisna od vrste končnih uporabnikov. Če uporabljajo Notes odjemalce in poznajo osnovne funkcije okolja, je zahtevan obseg izobraževanja majhen.	5

Tabela A-2: Ocene programske rešitve e-Proj.

5.6 A.3. Primavera TeamPlay

Kriterij in opis ocene	Ocena
1. Prilagodljivost procesu vodenja projektov	**8,2**
1.1 Nastavljive faze projekta Rešitev loči med projekti v izvajanju, planiranimi in neaktivnimi projekti. Nastavljive so tudi projektne faze, ki jih povežemo na WBS strukturo projekta.	7
1.2. Podpora različnim tipom projektov Na nivoju podjetja lahko definiramo strukturo projektov, ki ima poljubno število nivojev, vsak od nivojev pa lahko vsebuje poljubno število projektov.	9
1.3. Nastavljivi tipi projektne dokumentacije Projektna dokumentacija je nastavljiva na nivoju multiprojektnega okolja, lahko pa definiramo samo en nivo dokumentacije (dokumentacije ni možno grupirati po tipih).	7

1.4. Podpora privzetim parametrom Pri kreiranju projekta, lahko projekt povežemo na že končan projekt ali na projektno metodologijo. Povezava definira privzete aktivnosti, njihove rezultate, cilje in referenčne dokumente.	9
1.5. Podpora projektnim metodologijam Rešitev že sama po sebi ponuja podporo nekaterim metodologijam in standardom (npr. ISO 12207 Razvoj programskih rešitev). S pomočjo orodja Methodology Manager pa lahko kreiramo tudi svoje nastavitve, ki se skladajo s procesom obvladovanja projektov v podjetju.	9
2. Podpora časovnemu planiranju	**9,0**
2.1. Strukturirano definiranje aktivnosti Aktivnosti lahko naštejemo, grupiramo in določimo odvisnosti med njimi.	9
2.2. Načrtovanje trajanja in dela na aktivnostih Za vsako aktivnost lahko definiramo človeške in materialne vire, določimo trajanje in planiramo po različnih metodah (fiksiramo lahko enega ali več od parametrov: trajanje, delo in zasedenost). Aktivnostim lahko definiramo tudi omejitve.	9
2.3. Podpora grafičnim tehnikam Rešitev podpira spremljanje aktivnosti z gantogramom, mrežnim diagramom, grafično spremljanje odvisnosti med aktivnostmi in grafični prikaz virov za posamezne aktivnosti po obdobjih.	9
2.4. Podpora tehnikam časovnega planiranja Rešitev omogoča izračun in grafični prikaz kritične poti, spremljanje dejanskega izvajanja aktivnosti in PERT analizo.	9
3. Spremljanje izvedbe	**8,8**
3.1. Spremljanje statusa aktivnosti Rešitev omogoča spremljanje procenta izdelave aktivnosti (časovno, dejansko ali po delu). Aktivnost lahko povežemo z referenčno dokumentacijo, problemi in tveganji. V primeru, da se projekt izvaja na več lokacijah, lahko aktivnosti dodelimo tudi kje je oddelek, ki izvaja aktivnost. Pri obsežnejših aktivnostih lahko definiramo korake izvedbe, jih utežimo in samostojno spremljamo.	9
3.2. Podpora projektni dokumentaciji Projektno dokumentacijo lahko dodajamo na projekt ali posamezne projektne aktivnosti. Možen je strukturiran pregled in tipizacija dokumentacije.	8
3.3. Pregled problemskih stanj Na WBS strukturo ali aktivnosti lahko povežemo problemska stanja. Problemsko stanje ima definiran status, odgovorno osebo, rok izvedbe in opis stanja. Probleme lahko prožimo ročno, ali pa se kreirajo avtomatično iz prednastavljenih pogojev.	9
3.4. Preglednost podatkov Možni so različni pregledi multiprojeknega okolja. Vedno lahko zberemo nad koliko projekti pregledujemo podatke (od enega do vseh projektov v podjetju). Vsebina in varnostna shema je prilagojena projektni vlogi uporabnika.	9
4. Komunikacija med člani projektne skupine	**5,7**
4.1. Določene elektronske pošte uporabnikov Ob definiciji uporabnikov lociramo tudi njihove elektronske naslove in poštne strežnike.	7
4.2. Ročno obveščanje o dogodkih Možno je pošiljanje pošte med člani tima, če ti uporabljajo dostop do baze preko odjemalca. Preko spletnega odjemalci (TeamPlayer) pošte in delegiranje ni mogoče.	5
4.3. Avtomatično obveščanje Rešitev omogoča avtomatično spremljanje odstopanj, vendar o pojavu odstopanj ne obvešča po	5

elektronski pošti.	
5. Projektna poročila in baza znanja	**9,0**
5.1. Prednastavljena poročila Rešitev že ima prednasatvljena praktično vsa "smiselna" poročila. Najbolj verjetno, da bo uporabnik moral izbrati katera ga zanimajo. Poročila lahko naredimo v HTML obliki, kot navado besedilo ali pa jih pošljemo na tiskalnik.	9
5.2. Poročila na zahtevo uporabnika Kljub velikemu številu prednastavljanih poročil, lahko uporabnik sestavlja tudi poročila po lastni meri. Izredno uporabna opcija je objava projekta na spletni strani, kjer izberemo katere podatke in preglede želimo vključiti, rešitev pa nam zgradi projektno stran (skupaj z navigacijo).	9
5.3. Poročila multiprojektnega okolja Vsa poročila lahko zgradimo nad posameznim projektom, izbranimi projekti ali vsemi projekti v bazi.	9
5.4. Baza znanja Zaradi pregleda nad tveganji in problemi, je arhivska baza še posebej uporabna. Na podlagi uspešno izvedenega projekta lahko zgradimo svojo projektno šablono, ki služi kot osnova za naslednje projekte.	9
6. Tehnična ustreznost rešitve	**7,0**
6.1. Podpora različnim platformam Strežniška verzija programa podpira operacijska sistema Windows in Unix.	7
6.2. Način dostopa končnih uporabnikov Končni uporabniki lahko dostopajo do rešitve preko posebnega odjemalca ali pa preko spletni (TemaPlayer). Pri spletnem dostopu jim niso omogočene vse funkcionalnosti.	7
6.3. Podpora različnim relacijskim bazam Rešitev podpira relacijski podatkovni bazi Oracle in Microsoft SQL strežnik	7
7. Cena	**2,3**
7.1. Cena osnovnega paketa Sorazmerno s funkcionalnostmi rešitve je določena tudi cena ($50.000 za strežnik in cca $1000 za uporabnika). Vsekakor je glede na učinkovitost sistema cena primerna.	1
7.2. Cena nadgradnje posameznih modulov Nadgradnja sistema je namenjena redkim naročnikom, saj je velika večina kupcev sploh ne bo potrebovala.	1
7.3. Cena izobraževanja končnih uporabnikov Uporabniški vmesnik je ustrezno prilagojen delu v Windows okolju, je pa zaradi kompleksnosti rešitve potrebno kar nekaj časa, da se uporabnik dodobra seznani z rešitvijo.	5

Tabela A-3: Ocene programske rešitve Primavera TeamPlay.

6 Viri

Boehm B. W.: *Software Engineering Economics*. New York: Prentice-Hall, 1981.

Boehm B. W.: *Software Engineering Economics, Software Engineering Project Management*. Washington: Computer Society Press, 1984.

Boehm B.: *A spiral model of software development and enhancement*. IEEE Computer, 21(5):61-72, 1988.

Boehm B., Clark B., Horowitz E., Madachy R., Shelby R., Westland C. : *The COCOMO 2.0 Software Cost Estimation Model*, 1995, dostopno z: http://sunset.usc.edu/research/COCOMOII/index.html

Drucker P.: The Practice of Management. New York: Harper & Row, 1954.

Frame J.:*Managing projects in organizations: how to make the best use of time, techniques, and people.* San Francisco: Jossey-Bass, 1995.

Gašparini S., Kožman M.: *Informacijska podpora vodenju projektov*, članek, dostopno z: www.ipmit.si

Genis: *Vodenje projektov v praksi: e-Proj predstavitev.* Ljubljana: gradivo podjetja Genis d.o.o., 2002.

Gilb, T.,: *Principles of Software Engineering Management.* Wokingham:

Addison-Wesley Pub Co., 1988.

Isaacs F., Robinson E.: *The Project Management Puzzle.* Software Development, 1999, dostopno z http://www.sdmagazine.com/

Jacobson I., Booch G., Rumbaugh J.: *The Unified Software Development Process.* Reading, Massachusetts: Addison-Wesly, 1999

Kehoe R., Jarvis A.: *ISO 9000-3, A Tool For Software Product and Process Improvemen.* Berlin: Springer, 1996

Kovačič A., Vintar M.: *Načrtovanje in izgradnja informacijskih sistemov.* Ljubljana: DZS, 1994

Lipovec F.: *Razvita teorija organizacije*. Maribor: Založba Obzorja, Maribor, 1987.

Malotaux,N: *Evolutionary Development Methods,* 2001, dostopno z: http://www.malotaux.nl/nrm/pdf/MxEvo.pdf

MCNeice Filler S.: *Project Failures Spur Management Back to Basics*. 2001, dostopno z: http://www.billingworld.com/archive-detail.cfm?archiveId=4963&hl=PMI

Mills, H.D.: *Top-Down Programming in Large Systems. In Debugging Techniques in Large Systems.* Ed. R. New Jearsy, Prentice Hall. 1971.

Pointe Technology Group: *Software Testing and Quality Assurance White Papers*, 2001, dostopno z: http://www.pointetechnology.com/new/updated_text/qa_whitepaper.pdf

PMI Standards Committee: *A Guide to the Project Management Body of Knowledge*. B. k.: Upper Darby: Project Management Institute, 1996.

Pyron T.: *Using Microsoft Project 98*, Indianapolis:Que Corporation, 1997

Rant M., Jeraj M., Ljubič T.: *Vodenje projektov*. Radovljica: POIS Radovljica, d.o.o., 1995

Royce W.: *Managing the Development of Large Software Systems: Concepts and Techniques.* Los Angeles :Western Electronic Show and Convention, 1970

Royce W.: *Software Project Manageent, A Unified Framework*, Reading, Massachusetts: Addison-Wesly, 1998

Rozman R., Kovač J., Koletnik F.: *Management*. Ljubljana: Gospodarski vestnik, 1993.

Solina F., Križaj F.: *Organizacijski, prihološki in sociološki vidiki projektnega dela.* Ljubljana: Univerza v Ljubljani, Fakulteta za elektrotehniko in računalništvo, 1991

Stang D.: *Primavera Systems, Inc. TeamPlay*. članek: GartnerGroup, 2000.

Toplišek J.: *Elektronsko poslovanje*. Ljubljana: Atlantis Založba, 1998

Verzuh: E. *The Fast Forward MBA in Project Management*, New York:John Wiley & Sons., 1999

Williams P., *Getting a Project Done on Time : Managing People, Time, and Results,* b.k.: AMACOM 1996

Wingerd L., Seiwald C.: *High-Level Best Practicies in Software Configuration Management.* Berlin: Springer, ECOOP 98, SCM-8, str 57-66, 1998

ZPM Projektni forum 2001: *Projektni management v novi ekonomiji*, Maribor: ZPM, 2001