REPUBLIKA SLOVENIJA
MINISTRSTVO ZA OKOLJE IN PROSTOR
GEODETSKA UPRAVA REPUBLIKE SLOVENIJE

EVROPSKA UNIJA
EVROPSKI SKLAD ZA REGIONALNI RAZVOJ
NALOŽBA V VAŠO PRIHODNOST

ePROSTOR

METODOLOGIJA STRATEŠKEGA NAČRTOVANJA IN PROCESA RAZVOJA IS

Nadzor dokumenta

Spremenili

Datum	Avtor	Verzija	Opis spremembe
18.6.2017	GURS	1.0	Končna inicialna verzija

Kako brati dokument?

Pričujoči dokument je sestavljen iz več medsebojno povezanih delov. Prvi del je načrtovanje informacijskega sistema. Drugi del je proces razvoja informacijskega sistema. Sledijo vsi ostali procesi, ki so povezani z življenjem informacijskih sistemov. Zaradi obsežnosti dokumenta so koraki označeni s posebnimi znaki, ki omogočajo lažjo berljivost dokumenta.

Oznake v dokumentu:

Oznaka	Opis
	Opozorilo!
	Pomembno!
	Uporabni namigi. Omogočajo dostop do informacij, ki so neposredno povezani s tem dokumentom
	Obvezni koraki metodologije za večje projekte.
	Opcijski koraki metodologije. Lahko jih izpustimo pri manjših projektih
	Zgodovina pomembnih sprememb.
	Izdelki posameznih korakov metodologije. Dokumenti.

Dokument je sistematično zasnovan, da uporabnik izvaja korake, kateri so pomembni za njegov projekt ne glede na vsebino. V podjetju izvajamo projekte, katere lahko razdelimo glede na velikost in pomembnost. Glede na te kriterije metodologija predpostavlja obvezne ter neobvezne korake, ki jih je potrebno izpolniti.

Slovar izrazov

Dodatni izrazi z obrazložitvijo objektno orientirane tehnologije se nahajajo na naslovu: http://lisa.uni-mb.si/cot/slovar.htm

KAZALO VSEBINE

1 POVZETEK 6

2 UVOD IN NAMEN METODOLOGIJE 8

3 CILJI METODOLOGIJE 9

4 POVEZANOST METODOLOGIJE STRATEŠKEGA NAČRTOVANJA IN PROCESA RAZVOJA IS 10

5 STRATEŠKO NAČRTOVANJE RAZVOJA INFORMATIKE 11

5.1 POVZETEK STRATEGIJE RAZVOJA INFORMATIKE 11
5.2 IZHODIŠČA PRI NAČRTOVANJU RAZVOJA INFORMATIKE 11
5.2.1 ANALIZA STRATEŠKIH ELEMENTOV 11
5.2.1.1 POSLANSTVO IN VIZIJA 11
5.2.1.2 KLJUČNI DEJAVNIKI USPEHA (KDU) 12
5.2.1.3 ANALIZA CILJEV 12
5.2.1.4 RAZVRSTITEV CILJEV 12
5.2.2 DOKUMENTACIJA O IZHODIŠČIH 13
5.3 OBSTOJEČI INFORMACIJSKI SISTEM ORGANIZACIJE 13
5.3.1 PREGLEDNI MODEL 13
5.3.1.1 ORGANIZACIJSKA SHEMA 13
5.3.1.2 MODEL POSLOVNIH PROCESOV 14
5.3.1.3 POSLOVNA PRAVILA 14
5.3.1.4 ORGANIZIRANOST INFORMATIKE IN PREGLED KADROV 14
5.3.2 DOKUMENTACIJA O OBSTOJEČEM INFORMACIJSKEM SISTEMU 15
5.4 NAČRT INFORMACIJSKE TEHNOLOGIJE 15
5.4.1 RAZVOJNI TRENDI NA PODROČJU, NA KATEREM DELUJE ORGANIZACIJA 15
5.4.2 PREGLED OBSTOJEČEGA STANJA STROJNE OPREME 15
5.4.3 PREGLED OBSTOJEČEGA STANJA PROGRAMSKE OPREME 16
5.4.4 DOKUMENTACIJA O NAČRTU INFORMACIJSKE TEHNOLOGIJE 16
5.5 STRATEŠKE USMERITVE INFORMATIKE 16
5.5.1 OPREDELITEV PROJEKTOV IN PRIORITETE IZVAJANJA 16
5.5.2 TERMINSKI NAČRT IZVAJANJA PROJEKTOV 17
5.5.3 FINANČNI NAČRT (OCENA STROŠKOV IT) 17
5.5.4 NAČRTOVANJE KADROVSKIH POTREB 18
5.5.5 DOKUMENTACIJA O STRATEŠKIH USMERITVAH INFORMATIKE 18

6 PROCES RAZVOJA INFORMACIJSKEGA SISTEMA 19

6.1 UVOD 19
6.1.1 DINAMIČNI VIDIK 20
6.1.2 STATIČNI VIDIK 21
6.2 ZASNOVA 22
6.2.1 OSEBNA IZKAZNICA 23
6.2.2 PRIPRAVA PROJEKTNEGA OKOLJA 23
6.2.3 MEJNIK IN DOKUMENTACIJA IZ FAZE ZASNOVE 24
6.3 ELABORAT 24
6.3.1 MODEL POSLOVNEGA PROCESA 25
6.3.2 UGOTAVLJANJE FUNKCIONALNIH IN NEFUNKCIONALNIH ZAHTEV 27
6.3.3 PROTOTIP ARHITEKTURE INFORMACIJSKEGA SISTEMA 27
6.3.4 PROTOTIP UPORABNIŠKEGA VMESNIKA 28
6.3.5 NADALJEVANJE PRIPRAVE PROJEKTNEGA OKOLJA 28
6.3.6 PROJEKTNI SLOVAR 30
6.3.7 MEJNIK IN DOKUMENTACIJA IZ FAZE ELABORATA 30
6.4 IZDELAVA 30
6.4.1 IDENTIFIKACIJA AKTERJEV IN PRIMEROV UPORABE 31
6.4.2 ANALIZA ZAHTEV 34

6.4.3 ARHITEKTURA SISTEMA ... 35
6.4.3.1 IZGRADNJA LOGIČNE ARHITEKTURE ... 36
6.4.3.2 IZGRADNJA FIZIČNE ARHITEKTURE ... 37
6.4.4 NAČRTOVANJE ALGORITMOV ... 39
6.4.5 NAČRTOVANJE BAZE SISTEMA ... 44
6.4.6 KRITIČNE TOČKE SISTEMA ... 45
6.4.7 RAZVOJ KODE ... 45
6.4.8 DOPOLNJEVANJE PROJEKTNEGA SLOVARJA ... 46
6.4.9 MEJNIK IN DOKUMENTACIJA IZ FAZE IZDELAVE ... 46
6.5 PREHOD ... 46
6.5.1 TESTIRANJE SISTEMA ... 47
6.5.2 TESTIRANJE UPORABNIŠKIH VMESNIKOV ... 47
6.5.3 OPREDELITEV KOREKCIJ SISTEMA IN UPORABNIŠKIH VMESNIKOV ... 48
6.5.4 TEHNIČNA DOKUMENTACIJA ... 48
6.5.5 NAMESTITEV SISTEMA ... 49
6.5.6 UVAJANJE UPORABNIKOV ... 50
6.5.7 DOKUMENTACIJA ZA NAROČNIKA ... 50
6.5.8 ZAKLJUČEK IN DOKUMENTACIJA IZ FAZE IZDELAVE ... 50
6.6 PRODUKCIJA ... 51
6.6.1 VZDRŽEVANJE ... 52
6.6.2 UPRAVLJANJE Z ZAHTEVAMI PO SPREMEMBAH SISTEMA ... 52
6.6.3 MEJNIKI IN DOKUMENTACIJA IZ FAZE PRODUKCIJE ... 53

LITERATURA IN VIRI ... 55

KAZALO SLIK

SLIKA 1: STRUKTURA METODOLOGIJE NA PODROČJU IS V PODJETJU IGEA, D.O.O. ... 10
SLIKA 2: DVE DIMENZIJI RAZVOJA INFORMACIJSKEGA SISTEMA ... 19
SLIKA 3: FAZA ZASNOVE ... 23
SLIKA 4: FAZA ELABORATA ... 25
SLIKA 5: PRIMER STRUKTURE ORGANIZACIJE ... 26
SLIKA 6: PRIMER MODELA POSLOVNEGA PROCESA ... 26
SLIKA 7: PRIMER UPORABNIŠKEGA VMESNIKA ... 28
SLIKA 8: FAZA IZDELAVE ... 30
SLIKA 9: PRIMER DIAGRAMA PRIMEROV UPORABE ... 33
SLIKA 10: PRIMER RAZREDNEGA DIAGRAMA ... 34
SLIKA 11: PRIMER KOMPONENTNEGA DIAGRAMA ... 36
SLIKA 12: PRIMER NAMESTITVENEGA DIAGRAMA ... 37
SLIKA 13: PRIMER NAMESTITVENEGA IN KOMPONENTNEGA DIAGRAMA ... 39
SLIKA 14: PRIMER DIAGRAMA AKTIVNOSTI ... 41
SLIKA 15: PRIMER ZAPOREDNEGA DIAGRAMA ... 41
SLIKA 16: PRIMER DIAGRAMA SODELOVANJA ... 42
SLIKA 17: PRIMER DIAGRAMA STANJ ... 42
SLIKA 18: FAZA PREHODA ... 46
SLIKA 19: FAZA PRODUKCIJE ... 52

KAZALO TABEL

TABELA 1: STANDARDNI OBRAZEC ZA OPIS PRIMERA UPORABE ... 32
TABELA 2: PRIMER MATRIKE, S KATERO SLEDIMO ZAHTEVAM ... 33
TABELA 3: PRIMER MATRIKE ODGOVORNOSTI ... 34

1 Povzetek

Strateško načrtovanje informatike

Ažuren strateški plan za organizacijo predstavlja bazo znanja, ki jo lahko s pridom uporabljajo tako vodstvo, kot informatiki, pa tudi ključni uporabniki na različnih ravneh.

Strateški načrt informatike povzema poslovno strategijo in je zato uporaben za splošno vodstvo, ne samo za vodjo informatike in predstavlja podlago pri sprejemanju odločitev pri načrtovanju nadaljnjega razvoja informatike v podjetju.

Strateški načrt informatike zajema:

- poslanstvo in vizijo organizacije oziroma navedek, kje se ta dva dokumenta nahajata, ter vloga informatike pri izpolnjevanju poslanstva in vizije;
- seznam ključnih dejavnikov uspeha;
- urejen seznam ciljev s kratkimi komentarji;
- razredni diagram, ki predstavlja organizacijsko shemo;
- procesni diagram ključnih poslovnih procesov;
- seznam strateških poslovnih pravil, če je to potrebno;
- če je informatika razvejana, potem vključimo razredni diagram, ki predstavlja organizacijo informatike in, če je potrebno, vključimo tudi geografsko razporejenost kadrov;
- tabelo s številom ljudi po strokovni izobrazbi ali po vlogah…, ki predstavlja obstoječe stanje;
- kratek povzetek analiz uporabe IT v dejavnosti, s katero se ukvarja organizacija;
- opis obstoječega stanja v obliki namestitvenega in komponentnega diagrama;
- matriko povezav med načrtovanimi projekti ter cilji informatike
- Gantov diagram s terminskim načrtom projektov
- kratek povzetek srednjeročnega finančnega načrta oziroma opomba, kje se le – ta nahaja
- načrt potreb po kadrih ter opredelitev področij, kjer bomo iskali nove zunanje sodelavce.

Proces razvoja informacijskih sistemov

Proces razvoja informacijskih sistemov temelji na objektnem razvoju in je sestavljen iz več faz. Znotraj posameznih faz izvajamo aktivnosti, te pa bolj natančno delimo na opravila in korake. Osnovno vodilo procesa razvoja informacijskih sistemov je naravnanost na iterativen in inkrementalen procesni model. Pri tem posamezne izdelke izdelujemo v več iteracijah, v vsaki iteraciji pa temeljimo na rezultatih predhodnih iteracij, ki jih dopolnimo in nadgradimo.

Ker je potrebno vsak projekt razvoja informacijskega sistema umestiti v obstoječe okolje in vsebinske okvire, obravnavamo opis procesa razvoja kot ogrodje procesa razvoja, katerega prilagajamo zahtevam projektov. Prilagajanje ogrodja procesov razvoja posameznemu projektu zapišemo v dokumentu, ki nastane v fazi zasnove. V tem dokumentu določimo zahtevane izdelke posameznih faz.

Dokumentacija projektov razvoja informacijskega sistema lahko zajema:

- zapisniki sestankov projektne skupine;
- idejne rešitve na papirju;
- organizacijsko shemo organizacije;
- model poslovnih procesov;
- nabor funkcionalnih zahtev;
- nabor nefunkcionalnih zahtev;
- verificirani prototip uporabniškega vmesnika;
- rokovnik projekta;
- osebno izkaznico projekta;
- projektni slovar;
- model primerov uporabe;
- razredni diagram, ki predstavlja konceptualni pogled na informacijski sistem in se ga poslužujemo pri projektih, kjer domene ne poznamo dovolj dobro;
- razredni diagram na specifikacijskem nivoju;
- logično arhitekturo sistema;
- fizično arhitekturo sistema;
- poročilo o analizi zmogljivosti;
- poročilo o testiranju sistema in uporabniškega vmesnika;
- seznam zahtevanih popravkov na sistemu in uporabniškem vmesniku;
- navodila za namestitev sistema;
- navodila za uporabo.

2 Uvod in namen metodologije

Pogosto, ko govorimo o metodologijah, pravzaprav ne vemo dobro, kaj metodologija sploh je. Ko pravimo, da veliko organizacij pri svojem delu ne upošteva nobene metodologije, se motimo, saj ima organizirano in vodeno delo neko metodologijo (Bajec, Krisper, Agilne metodologije, str. 2). Seveda je iz tega konteksta potrebno izvzeti organizacije, kjer delujejo stihijsko, adhoc in popolnoma neurejeno, praktično brez metodologije. Metodologija zajema vse, kar običajno počnemo, da izdelamo končni izdelek oziroma opravimo storitev, ki je cilj našega dela. Vključno s tem, koga pri delu vključimo, kako delo razdelimo, kaj pri tem izdelamo, kako poteka komunikacija ipd. Je skupek postopkov, metod, tehnik, predpisov in standardov, ki jih uporabljajo člani organizacije pri svojem delu. Prežeta je s filozofijo, idejami in pogledi organizacije, v kateri delujemo, in je zato njen originalen izdelek.

Vsaka organizacija ima določeno metodologijo, ki pove, kako ta deluje, kako posamezniki znotraj organizacije komunicirajo in kako poteka odločanje.

Lahko se pa odločijo za eno izmed številnih obstoječih metodologij. Res pa je, da večinoma podjetja svojih metodologij ne zapišejo eksplicitno, temveč te obstajajo preprosto v glavah njihovih posameznikov. Izjema so organizacije, kjer velja stroga urejenost, npr. vojska. V mnogih primerih podjetja preprosto sledijo splošnim, referenčnim procesom. Pri tem si jih prilagodijo, tako da ustrezajo njihovim predstavam in filozofiji. S tem, ko jih prilagodijo, jim dodajo lastno vrednost.

Dokumentirana metodologija je sprva osnova, iz katere se uporabniki učijo in razvijejo svojo percepcijo o načinu dela, komunikaciji, odločanju itd. Skozi uporabo pridobivajo nove izkušnje in bogatijo svoje znanje, kar posledično vpliva tudi na metodologijo samo. Ko postane znanje, ki ga uporabljajo pri svojem delu, dovolj rutinsko, pridobi obliko podatkov in tedaj lahko postane del formalne metodologije (Bajec, Krisper, Agilne metodologije, str. 3).

Pomen vlaganj v informacijsko in komunikacijsko tehnologijo iz leta v leto narašča, saj nam prav ta, ob inovativnih načinih uporabe, lahko prinesejo večje poslovne koristi kot druge vrste vlaganj. Kapital je že dosegel mejno vrednost donosa in v informacijski dobi bo informacijska tehnologija tista, ki bo organizacijam omogočila rast, razcvet in poslovni uspeh, ob neupoštevanju njenega strateškega pomena pa izgubljanje prihodka, trgov in končen propad. Vlaganja v »pravo« informacijsko tehnologijo so se že v preteklosti izkazala za zelo težavna.

3 Cilji metodologije

Cilje, ki jih z metodologijo želimo doseči lahko opredelimo kot tiste stvari katere želimo dolgoročno izboljšati v skladu z izpolnjevanjem vizije ter poslovne strategije organizacije.

Cilji:

- ➔ strateški pristop pri načrtovanju informacijskih sistemov
- ➔ iterativni pristop pri razvoju, modeliranju informacijskih sistemov
- ➔ standardni pristop pri analiziranju in modeliranju informacijskih sistemov
- ➔ poenoten razvoj informacijskih rešitev
- ➔ procesno zasnovani informacijski sistemi
- ➔ uporaba tehnološko stabilnih in zrelih tehnologij
- ➔ skladnost s pridobljenim ISO 9001 standardom

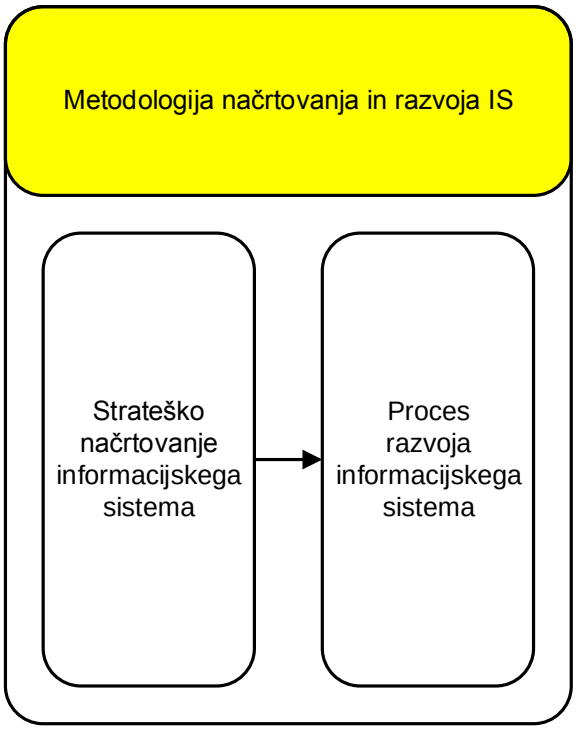
Posledično iz omenjenih ciljev izhaja:

- ➔ bolj kakovostni produkti
- ➔ večje zadovoljstvo naročnikov
- ➔ racionalizacija porabe delovnega časa
- ➔ boljša učinkovitost dela

4 Povezanost metodologije strateškega načrtovanja in procesa razvoja IS

Metodologija, ki se uporabljamo pri načrtovanju in razvoju informacijskih sistemov, je razdeljena na dva sklopa, ki sta lahko samostojna ali povezana. Osnovni koncept metodologije je povzet po EMRIS metodologiji in je prilagojen razmeram in pogojem, ki veljajo od 2008 dalje. V metodologiji smo za modeliranje sistema predvideli uporabo UML jezika. Strateško načrtovanje informacijskega sistema povezujemo z razvojem informacijskega sistema preko izdelkov ki nastajajo v posameznih korakih.

Slika 1: Struktura metodologije na področju IS



5 Strateško načrtovanje razvoja informatike

Projekt izdelave strateškega načrta traja praviloma od 3 do 6 mesecev, odvisno od velikosti in kompleksnosti organizacije, za katero načrt pripravljamo. Pri izvedbi projekta običajno sodelujejo zunanji svetovalci, ki so specialisti na posameznih področjih. Glavno breme pri izvedbi nosijo metodologi, ki prihajajo iz vrst informatikov, praviloma so zaposleni zunaj organizacije. Pri strateškem načrtovanju sodelujejo ključni uporabniki in člani vodstvene strukture, ki s svojim znanjem o ciljih, aktivnostih in podatkih v podjetju nenadomestljivo prispevajo h kakovosti strateškega načrta.

5.1 Povzetek strategije razvoja informatike

Povzetek je namenjen predvsem hitremu pregledu najpomembnejših točk iz vsebine dokumenta, zato vsebuje tri sklope: ozadje, obseg strategije ter zaključke.

Ozadje

Hiter pregled osnovnih smernic, ki smo jih upoštevali pri strateškem načrtovanju informatike.

Obseg strategije

Pregled in opis področij, ki smo jih v strateškem načrtu razvoja informatike podrobneje obdelali.

Zaključki

Opis najpomembnejših ciljev, ki jih želimo doseči v prihodnosti na področju informatike.

5.2 Izhodišča pri načrtovanju razvoja informatike

Najpomembnejše izhodišče pri načrtovanju razvoja informatike je povezati razvoj informatike s poslovno strategijo organizacije. Med ostalimi izhodišči pa lahko izpostavimo še pretok informacij in poslovnih procesov v organizaciji in upravljanje s človeškimi viri.

5.2.1 Analiza strateških elementov

Naloga in poslanstvo obravnavane aktivnosti je analizirati strateške elemente organizacije, da bo razvoj informatike v celoti usklajen s smernicami in cilji organizacije ter podati pregled obstoječega stanja informatike.

5.2.1.1 Poslanstvo in vizija

Poslanstvo je najvišji strateški element, ki opredeljuje smisel obstoja organizacije in dejavnosti, s katerimi se ukvarja ter predstavlja izhodišče vsem ostalim strateškim elementom.

Vizija organizacije jedrnat zapis želenih dosežkov, ki jih bo organizacija s svojim delovanjem skušala uresničiti. Pri definiranju vizije organizacije in njenih usmeritev je potrebno zagotoviti,

da bo tudi informatika imela mesto partnerja pri vsakodnevnem uresničevanju vizije in usmeritev organizacije.

Poslanstvo in vizija organizacije sta pogosto že opredeljena in zapisana v katerem od poslovnih dokumentov, zato pri izdelavi strateškega načrta le navedemo, kje se nahajata. V primeru, da omenjena strateška elementa še ne obstajata, je izdelava strateškega načrta odlična priložnost, da ju s sodelovanjem najpomembnejših predstavnikov vodstva sestavimo.

5.2.1.2 Ključni dejavniki uspeha (KDU)

Informatika je v preteklosti zasipala vodilne delavce z raznimi poročili in dokumenti. Tipično so bili zato vodilni kadri preobremenjeni s preveč podatki, tako da niso mogli izluščiti tistih podatkov, ki bi jim pomagali pri zaznavanju in reševanju problemov. V praksi velja, da je razmeroma malo informacij ključnih za delo vodstvene strukture in število odločitev, ki so še posebej pomembne, ni zelo veliko. Za iskanje najbolj kritičnih dejavnikov se v zadnjem času uspešno uporablja analiza KDU. Vizija in usmeritve organizacije sistema premalo natančno opišejo, katere vidike njegovega delovanja je potrebno dodatno krepiti oziroma katera področja so potrebna posebne pozornosti. Omenjeno pomanjkljivost odpravimo z analizo KDU.

KDU so tisti dejavniki, katerih uspešnost zagotavlja, da organizacija deluje uspešno.

KDU so lahko ekonomskega, tehnološkega ali organizacijskega značaja. Določeni morajo biti z naslednjih vidikov:

- Sporočilnost (bolj razumljivo je sporočilo, večjo moč ima KDU),
- Možnost preizkušanja (možnost določitve do kakšne stopnje je bil uresničen),
- Vpliv na IT (vpliv KDU na informacijsko tehnologijo),
- Organizacijski vpliv (vpliv na zaposlene, delovanje funkcij, organizacijsko shemo itd).

5.2.1.3 Analiza ciljev

Vsaka organizacija ima pri svojem delovanju določene cilje. V fazi strateškega načrtovanja je zaželeno cilje analizirati in jih dokumentirati. Cilje uporabljajmo kot kontrolni mehanizem znotraj organizacije. S cilji začrtamo želeno situacijo, tako da lahko kasneje izmerimo uspeh pri približevanju situaciji.

Dobro definiran cilj ima naslednje lastnosti:

- mora biti natančno določen, tako da niso možne različne razlage pri ocenjevanju uspešnosti njegovega uresničevanja,
- osredotočen naj bo na rezultate, tako da ga je možno razdeliti na aktivnosti, ki morajo biti opravljene in
- mora biti merljiv.

5.2.1.4 Razvrstitev ciljev

Cilje in usmeritve razvrstimo glede na njihovo pomembnost. V ta namen obstajajo različne lestvice in ena, ki vsebuje pet različnih ocen je naslednja:

- ocena 5, skrajno kritično za delovanje OS,
- ocena 4, kritično za nemoteno delovanje OS,
- ocena 3, potrebno za podporo delovanju OS,
- ocena 2, potrebno za podporo delovanju OS, vendar manj pomembno kot pri oceni 3 in

- ocena 1, zaželeno, vendar ne absolutno potrebno za podporo delovanju OS.

Razvrščanje ciljev in usmeritev je zelo zahtevna aktivnost. V izvedbo tega postopka je potrebno vključiti kadre, ki imajo potrebno znanje o organizaciji. Najbolj primerni so kadri iz vodstvene strukture, ki poznajo delovanje področja organizacije, ki ga vodijo, in s tem tudi pomembnost posameznih ciljev.

5.2.2 Dokumentacija o izhodiščih

V zvezi z izhodišči pri načrtovanju razvoja informatike dokumentiramo:

- ➔ poslanstvo in vizijo organizacije oziroma navedek, kje se ta dva dokumenta nahajata, ter vloga informatike pri izpolnjevanju poslanstva in vizije;
- ➔ seznam ključnih dejavnikov uspeha;
- ➔ urejen seznam ciljev s kratkimi komentarji.

5.3 Obstoječi informacijski sistem organizacije

Pred začetkom razvoja informacijskega sistema so nekatere funkcije ali posamezni procesi že informacijsko podprti. Zato izdelamo pregled obstoječega stanja (strojna oprema, programska oprema, komunikacijska oprema, pregled informatike in kadrov).

5.3.1 Pregledni model

Pregledni model organizacije je izdelek strateškega načrtovanja, ki nam nudi pregledno širino organizacije, globina pa je predmet kasnejših projektov razvoja.

5.3.1.1 Organizacijska shema

Organizacijska shema je prvi izdelek preglednega modela. Sestavljena je iz grafičnega prikaza in opisa organizacijskih enot. V praksi pojem organizacijska shema uporabljamo tudi zgolj za grafični prikaz organizacijske strukture. Izdelava organizacijske sheme je razmeroma enostavna, saj imajo praktično vse organizacije že izdelano neko obliko predstavitve svoje strukture.

Za grafično prikazovanje organizacijske strukture uporabljamo hierarhično drevesno strukturo, ki jo oblikujemo z razrednim diagramom.

Hierarhična struktura se začne na vrhu z eno samo vseobsegajočo enoto, ki jo imenujemo po hierarhični terminologiji koren strukture (ang. root). Po najbolj enostavni tehniki so podrejene enote nanizane vodoravno en nivo nižje v hierarhiji in na podoben način do najnižjega nivoja. Elemente na najnižjem nivoju lahko imenujemo liste zaradi podobnosti z drevesno strukturo.

Število nivojev in enot na enem nivoju ni omejeno, čeprav pri strateškem načrtovanju velja priporočilo, da organizacijske sheme ne rišemo globlje kot štiri nivoje. Za vsako enoto velja, da ima lahko nič, eno ali več podrejenih enot (vej) in da vedno pripada natanko eni nadrejeni enoti na prvem višjem nivoju.

Pri izdelavi organizacijske sheme dovoljujemo odstopanja od čiste hierarhije. Odstopanja so potrebna zaradi same "vsebine", kot tudi zaradi prostorskih problemov pri risanju kompleksnih shem.

Opis organizacijskih enot

Organizacijske enote, prikazane v organizacijski shemi, po potrebi opišemo. V takšnem primeru k opisu organizacijske enote sodi:

- vodja,
- kratka predstavitev področja dela organizacijske enote in
- podatki o številu zaposlenih in njihovi izobrazbi.

Za večje organizacijske enote izdelamo njihove organizacijske sheme. Zaradi preglednosti, te sheme prikažemo kar pri opisu organizacijskih enot.

5.3.1.2 Model poslovnih procesov

Poslovni procesi, ki se izvajajo v sklopu organizacije imajo zanjo strateški značaj, saj vplivajo na učinkovitost poslovanja in s tem na njeno konkurenčnost. Strateške cilje in usmeritve lahko organizacija doseže le s preglednim in učinkovitim izvajanjem poslovnih procesov.

S terminom *poslovni proces* označujemo množico povezanih aktivnosti, ki se izvajajo v organizaciji in posredno ali neposredno vplivajo na dodano vrednost pri uresničevanju skupnega cilja organizacije.

Modeliranje poslovnih procesov razkrije postopkovni vidik organizacije. Rezultat modeliranja predstavimo z *globalnim modelom poslovnih procesov*, ki je sestavljen iz grafične predstavitve ter besednega opisa najpomembnejših poslovnih procesov. Model poslovnih procesov grafično predstavimo z analitičnim diagramom.

5.3.1.3 Poslovna pravila

Poslovno pravilo je organizacijska kategorija. Le podmnožica poslovnih pravil, ki veljajo v organizaciji, je zajetih tudi v njenem informacijskem sistemu.

Poslovna pravila so močno izpostavljena spremembam, zato jih v strateški načrti razvoja informatike vključujemo le pod posebnimi pogoji – ko gre za poslovno pravilo, ki zajema kakšno pomembno strateško usmeritev.

5.3.1.4 Organiziranost informatike in pregled kadrov

Organizacija informatike v neki organizaciji temelji na:

1. povezavi s skupnim podatkovnim centrom, ki je del širšega sistema, v katerega sodi tudi OS,
2. oddelku/službi/sektorju za informatiko (znotraj OS),
3. povezavi z zunanjimi izvajalci (outsourcing) oziroma
4. na poljubni kombinaciji navedenih možnosti.

V pregledu kadrov evidentiramo kadre s področja informatike. Kadre lahko razvrstimo na več načinov, eden od njih je izobrazbena struktura, drugi možni način so na primer vloge. V primeru, ko je informatika projektno organizirana, je bolj aktualna izobrazbena struktura, če pa gre za funkcionalno organizirano informatiko, pa kadre preštejemo po vlogah. Vloge so lahko naslednje:

- vodja službe za informatiko,
- direktor informacijskega centra,
- svetovalec za IS,
- sistemski analitik,

- sistemski inženir,
- razvijalec - načrtovalec,
- skrbnik podatkovnih baz,
- razvijalec - programer,
- podpora uporabnikom,
- skrbnik aplikacij in
- operater.

5.3.2 Dokumentacija o obstoječem informacijskem sistemu

V dokumentacijo strateškega načrta uvrstimo:

- ➔ razredni diagram, ki predstavlja organizacijsko shemo;
- ➔ procesni diagram ključnih poslovnih procesov;
- ➔ seznam strateških poslovnih pravil, če je to potrebno;
- ➔ če je informatika razvejana, potem vključimo razredni diagram, ki predstavlja organizacijo informatike in, če je potrebno, vključimo tudi geografsko razporejenost kadrov;
- ➔ tabelo s številom ljudi po strokovni izobrazbi ali po vlogah…, ki predstavlja obstoječe stanje.

5.4 Načrt informacijske tehnologije

Načrt informacijske tehnologije obsega pregled obstoječega stanja ter specifikacijo predlaganih rešitev strojne opreme, komunikacijske opreme in programske opreme. Načrt se lahko izdela v več primernih različicah, ki ustrezajo danim omejitvam in zahtevam in med katerimi kasneje izberemo optimalno.

5.4.1 Razvojni trendi na področju, na katerem deluje organizacija

Pri pripravi strateškega načrta razvoja informatike so seveda zelo pomembni razvojni trendi na področju, ki ga informatika podpira. Zato pregledamo dostopne analize in zabeležimo trende, ki so aktualni za področje, na katerem deluje organizacija, katere poslovanje podpira informatika. Te trende nato upoštevamo pri pripravi načrtovanega stanja informacijske tehnologije.

5.4.2 Pregled obstoječega stanja strojne opreme

V pregledu strojne opreme naštejemo strojno računalniško in periferno opremo, operacijske sisteme, ki jih uporabljajmo ter sisteme za upravljanje podatkovnih baz. Pregled lahko razdelimo po organizacijskih enotah in bolj podrobno po zaokroženih sklopih strojne opreme, če je to potrebno zaradi preglednosti dokumenta. V opisu lahko naštejemo skrbnike, ki vzdržujejo operativnost strojne in programske opreme in leto nabave opreme.

Podobno tudi v pregledu komunikacijske opreme naštejemo komunikacijsko opremo, operacijske sisteme, ki jih uporabljajmo, standardne protokole, povezave z ostalimi omrežji in skrbnike opreme. Pregled lahko tudi v tem primeru razdelimo po organizacijskih enotah in bolj podrobno po zaokroženih sklopih komunikacijske opreme, v kolikor to prispeva h jasnejši sliki stanja.

Grafični pregled obstoječega stanja strojne opreme pripravimo z namestitvenim diagramom, ki ga rišemo na visoki ravni in zajema predvsem strežnike, delovne postaje, glavna vozlišča…ter povezave med njimi.

Pogledu na obstoječe stanje dodamo še pogled na načrtovano stanje in ga opišemo na enak način.

5.4.3 Pregled obstoječega stanja programske opreme

V pregledu programske opreme ob programski opremi lahko naštejemo odgovorne osebe, ki skrbijo za opremo, uporabnike, tehnologijo, v kateri je izdelana in sistemsko platformo, na kateri se uporablja. Dodamo še pregled varnosti, sistemov za upravljanje in nadzor ter pregled načinov računalniške izmenjave podatkov

Aplikacije, ki podpirajo ključne poslovne procese v organizaciji, opišemo tudi podrobneje. Pogosto imamo v organizaciji veliko število različnih aplikacij, zato težko vse opišemo. V takem primeru opišemo zgolj aplikativne sisteme, ki združujejo aplikacije istega področja.

Grafični pregled obstoječega stanja programske opreme pripravimo s komponentnim diagramom, ki ga rišemo na visoki ravni in zajema predvsem sklope aplikacij ter povezave med njimi.

Pogledu na obstoječe stanje na področju programske opreme dodamo še pogled na načrtovano stanje ter ga opišemo.

5.4.4 Dokumentacija o načrtu informacijske tehnologije

V okviru načrta informacijske tehnologije zabeležimo v strateški načrt naslednje:

- ➔ kratek povzetek analiz uporabe IT v dejavnosti, s katero se ukvarja organizacija;
- ➔ kot opis obstoječega stanja lahko združimo namestitveni diagram, ki predstavlja stanje strojne opreme ter komponentni diagram, ki predstavlja stanje programske opreme ter dodamo kratek opis platform, baz podatkov, orodij za razvoj sistemov, komunikacij, varnosti, sistemov za upravljanje in nadzor, načinov računalniške izmenjave podatkov…
- ➔ kot opis načrtovanega stanja pa lahko združimo namestitveni diagram načrtovanega stanja strojne opreme ter komponentni diagram, ki predstavlja načrtovano stanje programske opreme ter dodamo kratek opis platform, baz podatkov, orodij za razvoj sistemov, komunikacij, varnosti, sistemov za upravljanje in nadzor, načinov računalniške izmenjave podatkov… po en stavek na vsako od naštetih tem; kratek povzetek analiz uporabe IT v dejavnosti, s katero se podjetje ukvarja.

5.5 Strateške usmeritve informatike

Celoten razvoj informacijskega sistema traja običajno nekaj let, zato je nujna odločitev o usmeritvah. Pri tem običajno sledimo ključnemu poslovnemu procesu organizacije, ki ga podpira informatika.

5.5.1 Opredelitev projektov in prioritete izvajanja

Načrt razvoja informacijskega sistema neke organizacije so terminsko in po sredstvih opredeljene aktivnosti, ki jih izvajamo zato, da uresničimo strateški načrt. Nekatere opredeljene aktivnosti so samostojni projekti, druge pa se izvajajo v okviru rednih nalog.

Identificirane projekte razvoja informacijskega sistema razvrstimo po prioritetni izvedbeni listi. Najvišje na listo uvrščamo projekte, ki rešujejo kritične probleme in imajo visoke učinke.

Kriteriji, ki jih uporabljamo pri pripravi prioritetne liste, so lahko naslednji: obstoječe stanje, vpliv na poslovanje, vpliv na zunanje subjekte, strateški pomen, pokritost, kritičnost kadrov, zaporedje izvedbe in drugi.

Določitev prioritet projektom je ena od zelo pomembnih nalog pri pripravi strateškega načrta razvoja informatike. Informacijski projekti morajo biti tekom procesa določitve prioritet dobro usklajeni s poslovnimi usmeritvami. H določitvi prioritet projektov poskušamo pritegniti vodstvene delavce, ker je njihovo aktivno sodelovanje zelo zaželeno. Ker pa vsak vodstveni delavec daje najvišjo prioriteto projektom s svojega funkcionalnega področja, je moramo dobro definirati proces, po katerem bomo določali prioritete. Proces nato vodstvenim delavcem razumljivo prikažemo, če jih želimo aktivno vključiti vanj. Tudi informacije, ki smo jih pridobili v predhodnih korakih strateškega načrtovanja, lahko s pridom uporabimo pri določitvi prioritet posameznih projektov.

Določitev prioritet projektom je odločitveni proces. To je proces, pri katerem iz množice variant poskušamo izbrati takšno, ki najboljše ustreza ciljem, ki jih postavi odločevalec. V našem primeru pa gre za ovrednotenje variant (projektov) od najpomembnejše do najmanj pomembne za poslovanje organizacije. Prioritetno lestvico projektov lahko sestavimo z oceno posameznih kriterijev in vpliva projektov nanje z vrednostmi z določene merske lestvice, npr. od 0 do 10, oziroma medsebojno primerjavo projektov vsakega z vsakim, pri čemer pomembnejši pridobi točko, ali pa se odločimo za kvalitativno ocenjevanje z večparametrskim odločanjem.

Pripravljeno prioritetno lestvico projektov prikažemo v povezovalni matriki s cilji informatike, iz katere je razvidno pokrivanje projektov ter ciljev.

5.5.2 Terminski načrt izvajanja projektov

V terminskem načrtu časovno opredelimo aktivnosti oz. projekte, katerih izvajanje smo predvideli v fazi opredelitev projektov in prioritet izvajanja. Gre zgolj za okvirni načrt, saj je pri velikih sistemih potrebno, predvsem pri aplikativnih projektih, dopolniti fazo strateškega načrtovanja s podrobnejšim načrtovanjem na funkcionalnem področju

Strateški načrt je običajno pripravljen za srednjeročno obdobje treh let. Potrebnega obsega sredstev in kadrov, seveda, ni mogoče popolnoma natančno določiti, so pa podane ocene zelo dobra orientacija potreb.

Grafično predstavimo terminski načrt izvajanja projektov z Gantovim diagramom.

5.5.3 Finančni načrt (ocena stroškov IT)

Finančni načrt informatike običajno pripravljamo skupaj s finančnimi načrti ostalih sektorjev v organizaciji. Zato na tem mestu podamo le kratek povzetek tega načrta ali pa navedemo, kje se le – ta nahaja.

5.5.4 Načrtovanje kadrovskih potreb

Na osnovi načrtovanih projektov zabeležimo morebitna potrebna nova znanja ter pripravimo načrt potrebnih kadrov. Načrtujemo nove zaposlitve pa tudi sodelovanje z zunanjimi izvajalci. Načrt spet zabeležimo v tabeli, kjer načrtovane novo zaposlene kadre razvrščamo po strokovni izobrazbi ali pa po vlogi, ki jo bodo imeli v informatiki, medtem ko nove povezave z zunanjimi izvajalci zabeležimo tako, da opredelimo področja, kjer bomo iskali nove povezave.

5.5.5 Dokumentacija o strateških usmeritvah informatike

V sklop strateških usmeritev informatike zapišemo:

- matriko povezav med urejenim seznamom načrtovanih projektov ter urejenim seznamom ciljev informatike
- Gantov diagram s terminskim načrtom projektov
- kratek povzetek srednjeročnega finančnega načrta oziroma opomba, kje se le – ta nahaja
- načrt potreb po kadrih ter opredelitev področij, kjer bomo iskali nove zunanje sodelavce.

6 Proces razvoja informacijskega sistema

Proces razvoja informacijskih sistemov temelji na objektnem razvoju in je sestavljen iz več faz. Znotraj posameznih faz izvajamo aktivnosti, te pa bolj natančno delimo na opravila in korake. Osnovno vodilo procesa razvoja informacijskih sistemov je naravnanost na iterativen in inkrementalen procesni model. Pri tem posamezne izdelke izdelujemo v več iteracijah, v vsaki iteraciji pa temeljimo na rezultatih predhodnih iteracij, ki jih dopolnimo in nadgradimo.

Ker je potrebno vsak projekt razvoja informacijskega sistema umestiti v obstoječe okolje in vsebinske okvire, obravnavamo opis procesa razvoja kot ogrodje procesa razvoja, katerega prilagajamo zahtevam projektov. Prilagajanje ogrodja procesov razvoja posameznemu projektu zapišemo v dokumentu, ki nastane v fazi zasnove. V tem dokumentu določimo zahtevane izdelke posameznih faz.

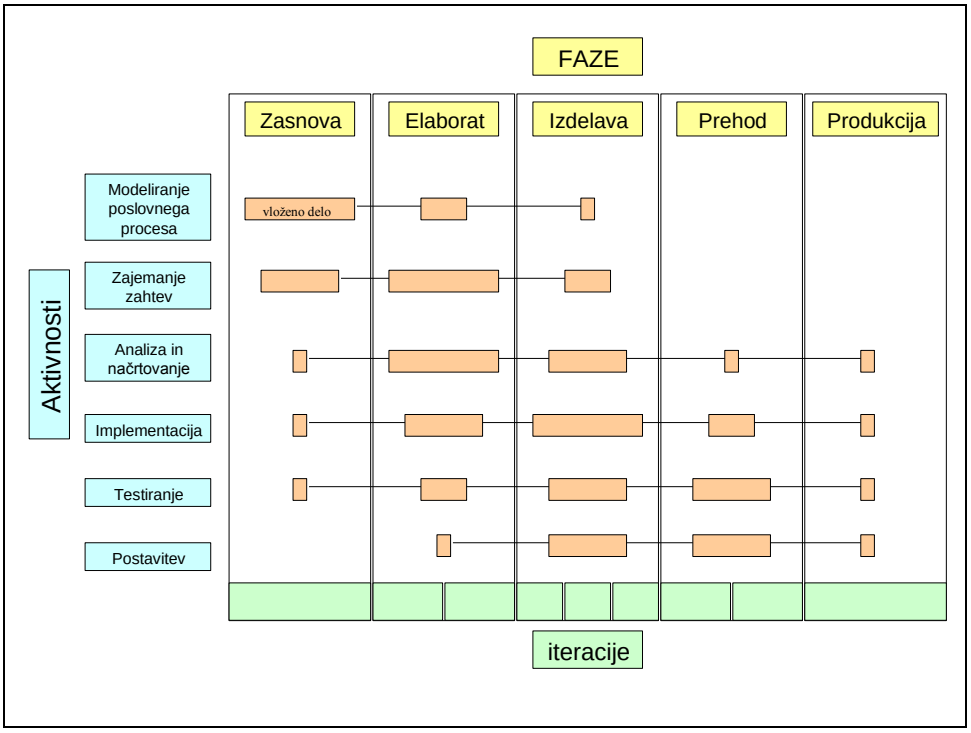
Sodelovanje z naročnikom izvajamo na formalni ravni preko podpisanih zapisnikov s sestankov, ki jih beležimo na za to predviden obrazcu ter preko potrjevanja izdelkov, ki nastajajo tekom razvoja. Vse potrditve in soglasja beležimo z zato predvidenem obrazcem, v katerem navedemo, na katere izdelke se soglasje oziroma otrditev nanaša ter priložimo izdelke h obrazcu, ki ga podpisanega tako s strani naročnika kot s strani izvajalca, vlagamo v projektno dokumentacijo.

6.1 Uvod

Proces razvoja informacijskega sistema je razdeljen na posamezne faze. Pri objektnem razvoju informacijskih sistemov ločimo pet osnovnih faz (zasnova, elaborat, izdelava, prevzem, produkcija). Faze prikazujejo časovno delitev objektnega procesa. Iterativno inkrementalna narava objektnega pristopa narekuje predstavitev procesnega modela v dveh dimenzijah kot je prikazano na spodnji sliki:

- po času – predstavlja življenjski cikel procesa – dinamični vidik,
- po aktivnostih – predstavlja sestavne dele procesa – statični vidik.

Slika 2: Dve dimenziji razvoja informacijskega sistema



6.1.1 Dinamični vidik

Dinamični vidik procesa razvoja informacijskega sistema je vidik, ki mu bomo sledili tudi v nadaljevanju dokumenta. Sestavljen je iz petih faz: zasnove, elaborata, izdelave, prehoda in produkcije.

Zasnova

Projekt definiramo s poslovnega stališča in določimo njegov obseg. V ta namen poiščemo vse zunanje entitete, s katerimi bo sistem sodeloval, ter v grobem definiramo način sodelovanja. Poiskati moramo vse primere uporabe, opišemo pa le najpomembnejše. Poslovno definiranje projekta zajema določitev kriterija uspeha oziroma neuspeha projekta, ocenitev tveganj, presojo potrebnih virov in fazni načrt, iz katerega so razvidni glavni časovni mejniki.

Elaborat

Analiziramo problemsko področje, postavimo osnovno ogrodje arhitekture sistema, izdelamo projektni načrt in razrešimo najbolj tvegane elemente projekta. Arhitekturne odločitve morajo upoštevati sistem kot celoto, kar zahteva podroben opis večine primerov uporabe in upoštevanje nekaterih dodatnih omejitev. Prototipno realiziramo sistem do te mere, da lahko prikažemo vse glavne primere uporabe in ovrednotimo izbrano arhitekturo. Na koncu te faze pregledamo podrobne cilje sistema in njegov obseg, izbiro arhitekture in morebitna tveganja.

Izdelava

Z iterativno inkrementalnim postopkom izdelamo celoten proizvod, ki je pripravljen za prenos k uporabniku. Faza konstrukcije zajema opis preostalih primerov uporabe, načrtovanje, zaključek implementacije in testiranje. Na koncu te faze moramo presoditi ali je izdelana programska oprema kot tudi uporabniki pripravljeni za opravljanje vsakodnevnih nalog.

Prehod

Osnovni cilj te faze je predaja izdelka v uporabnikovo okolje. S tem se pogosto pojavi potreba po dodatnih popravkih, ki prilagodijo sistem, popravijo neodkrite probleme ali dokončajo nekatere funkcije, ki so bile namenoma izpuščene. Tipično ta faza nastopi z beta verzijo proizvoda. Na koncu te faze ocenimo nivo zadovoljitve zadanih ciljev in ali je potrebno pričeti z razvojem nove generacije sistema. Seveda je to prav tako primeren trenutek za izboljšanje obstoječega procesa izhajajoč iz pridobljenih izkušenj.

Produkcija

V času, ko je sistem v uporabi, je na njem potrebno vršiti vzdrževalna dela, pa tudi, na zahtevo uporabnikov, pripravljati nadgradnje, spreminjati sistem, nuditi podporo uporabnikom.

6.1.2 Statični vidik

Znotraj vsake faze izvajamo aktivnosti, ki predstavljajo statični vidik razvoja informacijskega sistema. Aktivnosti, ki jih izvajamo so hkrati oporne točke pri načrtovanju, modeliranju bodočega sistema. Aktivnosti razdelimo kot je prikazana na spodnji sliki in jih izvajamo v odvisnosti od faze v kateri se nahajamo.

Aktivnost Modeliranje poslovnega procesa

Z modeliranjem poslovnega procesa ugotavljamo, kje v poslovnem procesu bo bodoči informacijski sistem umeščen, s katerimi drugimi obstoječimi informacijskimi sistemi bo komuniciral in komu od zaposlenih bo bodoči informacijski sistem koristil.

Aktivnost Zajemanje zahtev

Poglavitni namen aktivnosti je pridobiti uporabnikove zahteve za nastajajoč sistem v obliki primerov uporabe, lastnosti in ne-funkcionalnih zahtev. Izhod te aktivnosti je model primerov uporabe, scenariji primerov uporabe in opis nefunkcionalnih zahtev.

Aktivnost Analiza in načrtovanje

Skozi analizo in načrtovanje sistema definiramo, katere operacije in objekti bodo prisotni v razvijajočem se sistemu, ne oziraje se na to, kako bomo te operacije in objekte izdelali. V nadaljevanju pa operacije priredimo objektom in sprejmemo odločitve glede dedovanja, vidljivosti in predstavitve povezav (asociacij). Opišemo tudi pristope k sočasnemu izvajanju posameznih delov sistema in s tem v povezavi tudi sinhronizacijo. Določimo strukturo sistema v smislu komponent in povezav med njimi. Izhod je razredni diagram načrtovanja, diagram sodelovanja objektov, začetna konfiguracija objektov in fizična arhitektura sistema.

Aktivnost Implementacija

Rezultate načrtovanja implementiramo v enem izmed programskih jezikov. Pri tem upoštevamo lastnosti izbranega programskega jezika. Potrebno je razviti tako tehnološke kot tudi poslovne razrede. Pri razvoju je potrebno upoštevati izbrano arhitekturo sistema. Upoštevati je potrebno tudi nefunkcionalne zahteve.

Aktivnost Testiranje

Izvedemo testiranje izdelka. V sklopu te aktivnosti je potrebno izvesti tako testiranje enot kot tudi integracijsko testiranje. Rezultat aktivnosti je poročilo o testiranju, nastane pa tudi množica testnih primerov in testnih vzorcev, s katerimi zagotovimo ponovljivost testiranja. V primeru, ko želimo testiranje avtomatizirati, razvijemo testne razrede.

Aktivnost Postavitev

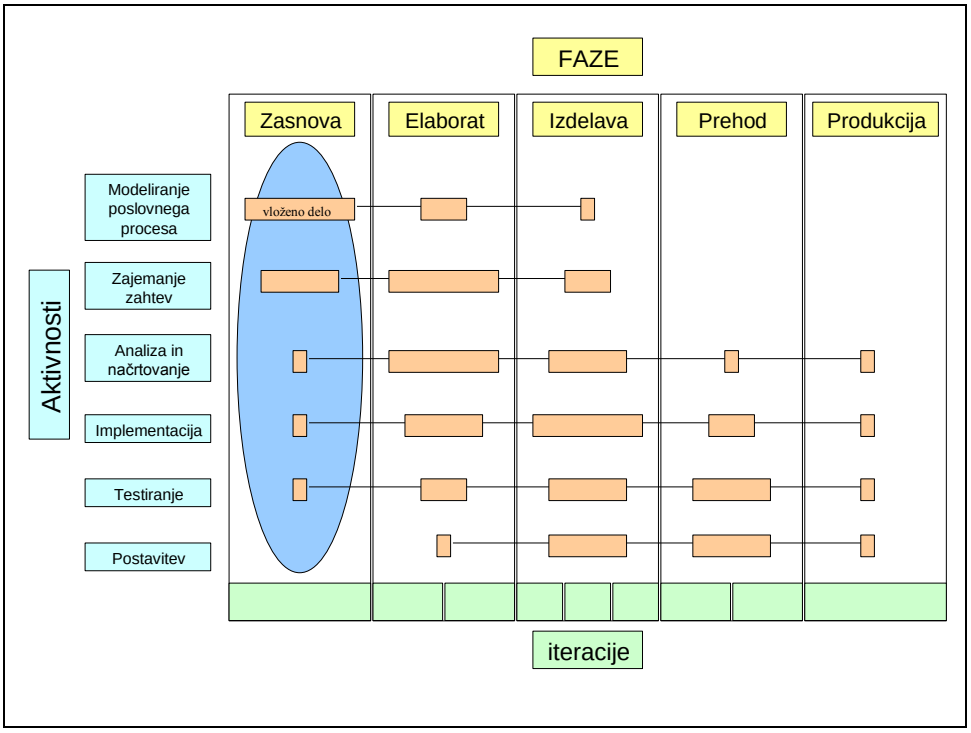
V aktivnost Postavitev štejemo priprave produkcijskega okolja, organiziranje uvajanja sistema in namestitev sistema.

6.2 Zasnova

Osnovni cilj Zasnove je soglasje vseh zainteresiranih za informacijski sistem o ciljih projekta in o priznanem fondu sredstev za projekt.

Fazo zasnove lahko izvajamo na več načinov. Pri nekaterih projektih jo lahko izvedemo skozi kratek razgovor, če pa gre za večji projekt, bomo mogoče samo za študijo izvedljivosti projekta potrebovali mesec ali več. Med fazo zasnove pripravljamo poslovne primere uporabe za projekt. To v grobem pomeni, da skušamo dobiti čim boljši občutek za to, kakšni so cilji projekta, zaradi česar bomo mogoče izvedli kakšno predhodno analizo, da ugotovimo velikost projekta. Nadalje ugotavljamo, kakšni bodo stroški projekta in kakšen bo zaslužek s projektom. V splošnem pa velja pravilo, da prvo fazo izpeljemo v nekaj dneh, kolikor pač potrebujemo, za razmislek o tem, ali se nam izplača investirati v natančnejše raziskovanje, ki bo potekalo v fazi elaborata. V tej točki naj bi sponzor projekta pristal na resno raziskavo v okviru projekta. Dokumentacija faze zasnove so zapisniki sestankov, ki jih pripravljamo na standardiziranem obrazcu podjetja, idejne rešitve na papirju, ter grobi finančni in terminski načrt, ki ga potrdi interna projektna skupina. V tej fazi pripravimo tudi dokument, v katerem opredelimo zahtevane izdelke posameznih faz. Ta dokument je nato vodilo pri pripravi projektne dokumentacije.

Slika 3: Faza zasnove



6.2.1 Osebna izkaznica

Osnovni cilj prve faze razvoja informacijskih sistemov je zasnova projekta. V tej fazi se zato ukvarjamo z ugotavljanjem možnosti izvedbe projekta in z opisom poslovnega problema, k kateremu bo projekt prispeval dodano vrednost. Poleg tega, poskušamo v tej fazi zajeti bistvene zahteve, ki nam pri kasnejšem delu služijo za orientacijo, da ne izgubimo rdeče niti projekta ter identificiramo možna tveganja. Vse to opišemo v Osebni izkaznici projekta, ki hkrati predstavlja strateško komponento razvoja.

Osebna izkaznica je sestavljena iz ključnih podatkov, kateri nam na strateškem nivoju predstavljajo projekt in povezave na ostale projekte. Segmenti osebne izkaznice so:

- glavni podatki o projektu
- opis projekta
- povzetek projekta
- izhodišča
- cilji
- omejitve
- informacijska rešitev
- organizacijska struktura projekta
- dejavniki tveganja
- terminski načrt
- finančni načrt

6.2.2 Priprava projektnega okolja

V pripravo projektnega okolja vključujemo preverjanje razpoložljivosti ljudi, ki jih bomo potrebovali na projektu, pripravo okvirnega rokovnika projekta, pripravo orodij, ki jih

potrebujemo takoj ter pripravo liste programske in strojne opreme, ki jo bomo morali še pridobiti.

6.2.3 Mejnik in dokumentacija iz faze zasnove

Ob zaključku faze zasnove morajo sodelujoči v interni projektni skupini doseči soglasje o naslednjih točkah:

- obseg projekta (sodelujoči se strinjajo z obsegom projekta),
- definicije začetnih zahtev (soglasje o pravilnem naboru zahtev, pa čeprav le na visokem nivoju, ter enako razumevanje teh zahtev),
- okvirni stroški (soglasje o ocenjenih okvirnih stroških celotnega projekta),
- dejavniki tveganja (soglasje o dejavnikih tveganja in strategijah za odpravljanje teh dejavnikov),
- razvojni proces (razvojni proces sooblikujejo vsi udeleženci),
- poslovni cilji projekta (soglasje o ciljih, ki projekt upravičujejo v poslovnem smislu),
- okvirni rokovnik projekta (soglasje o okvirnem rokovniku, ki ga nadgradimo v naslednji fazi),
- združljivost projekta z obstoječim informacijskim sistemom (po potrebi opravimo portfolio analizo).

Dokumentacija, ki nastane v fazi zasnove:

- ➔ zapisniki sestankov;
- ➔ idejne rešitve na papirju;
- ➔ soglasje interne projektne skupine o točkah iz osebne izkaznice.

6.3 Elaborat

S potrditvijo zasnove osebne izkaznice projekta imamo zeleno luč za začetek. V tej točki poznamo okvirne ideje o zahtevah, zato je naša prva naloga, da se spoznamo s problemsko domeno. Poskušamo si odgovoriti na naslednja vprašanja:

- Kaj bomo dejansko gradili?
- Kako bomo to izvedli?

Pri odločanju o tem, na kaj bomo še posebej pozorni v tej fazi, nam bodo vodilo tveganja v našem projektu. Tveganja lahko razdelimo na štiri kategorije:

- Tveganja v povezavi z zahtevami. Kakšne so zahteve sistema? Soočiti se moramo z možnostjo, da bomo zgradili sistem, ki ne bo počel tega, kar želijo uporabniki.
- Tveganja v povezavi s tehnologijo. Kakšna so tehnološka tveganja v našem projektu? Ali so izbrali tehnologijo, ki bo zmogla opraviti svoj posel? Ali bodo različni delčki delovali povezano kot smo si zamislili?
- Tveganja v povezavi z izkušenostjo. Ali lahko pridobimo osebje in strokovnjake, ki jih potrebujemo?
- Tveganja v povezavi s politiko. Ali obstajajo politične sile, ki nam lahko preprečijo izvedbo projekta?

Faza elaborata običajno zavzame petino časa celotnega projekta. Lahko pa se orientiramo še po dveh ključnih indikatorjih (Fowler et al., 2001, str. 24):

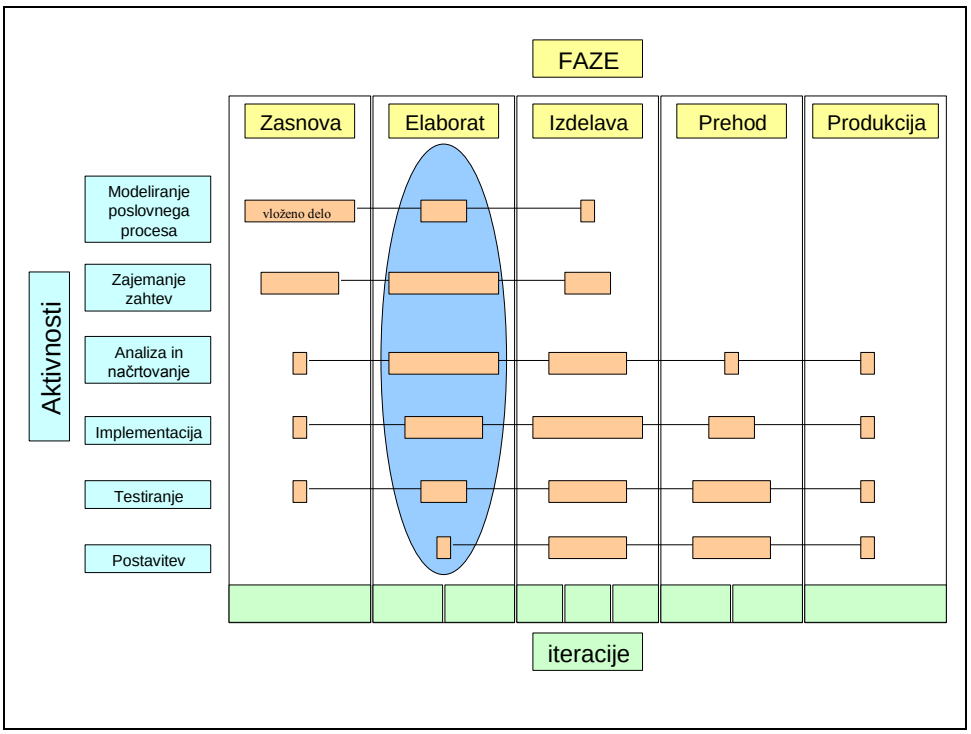
- Razvijalci lahko dokaj dobro ocenijo čas v človek – tednih, ki ga bodo potrebovali za izgradnjo vsakega od primerov uporabe.
- Identificirali smo vsa pomembna tveganja, glavna smo tudi raziskali do te mere, da jih dobro razumemo in vemo, kako bomo ravnali, da se jim izognemo.

Naš osnovni cilj v drugi fazi razvoja informacijskega sistema predstavlja priprava *osnovne izvedljive arhitekturne rešitve*. Pripravljamo delno rešitev, ki bo na koncu predstavljala del

celotne rešitve. O delni rešitvi govorimo iz preprostega razloga. V tej točki se vodstvo podjetja odloči, ali bomo sistem, ki je predmet projekta, razvijali kot več podsistemov, ali pa bomo pripravili načrt celotnega sistema. Drugi razlog je dejstvo, da običajno v prvi iteraciji ugotavljanja zahtev ne zajamemo vseh zahtev, ki naj jih izpolni sistem in nenazadnje, obstaja možnost, da bomo osnovno rešitev širili z dodatnimi funkcionalnostmi, ki jih bodo naročniki izrazili šele kasneje, ko bodo dobili občutek za to, kaj lahko z informacijskim sistemom pridobijo. V tej fazi se ukvarjamo tudi s pripravo podrobne ocene tveganj, natančneje definiramo atribute, po katerih bomo sledili napredku projekta, določimo projektno skupino, določimo ostale potrebne vire, tako sredstva, kot opremo in pripravimo rokovnik za projekt.

Druga faza je najbolj kritična in zato tudi najzahtevnejša faza projekta, kajti vse naslednje faze črpajo iz rezultatov Elaborata.

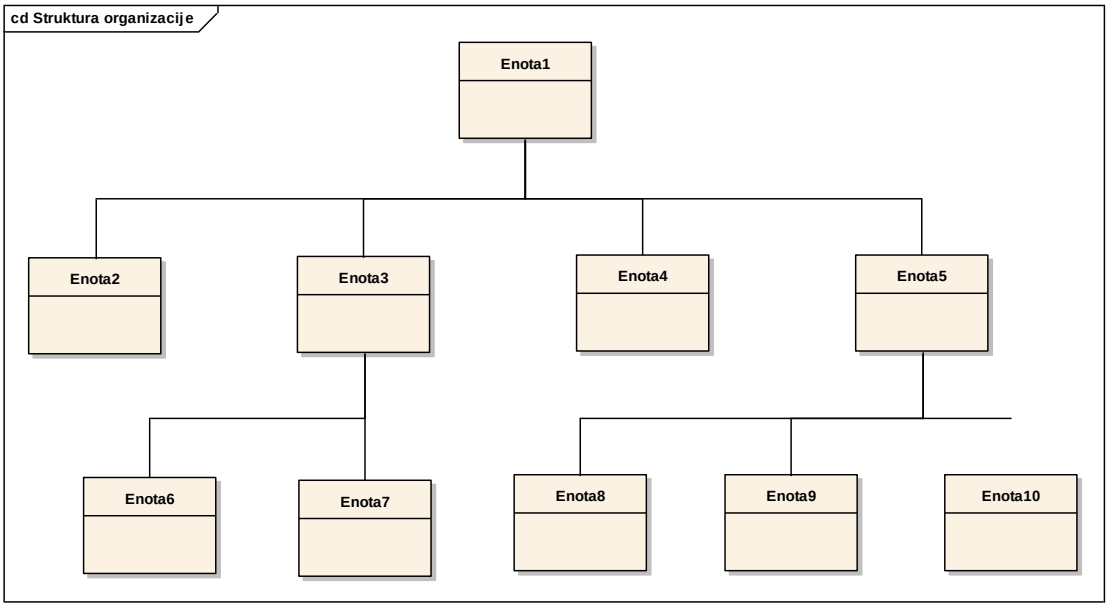
Slika 4: Faza elaborata



6.3.1 Model poslovnega procesa

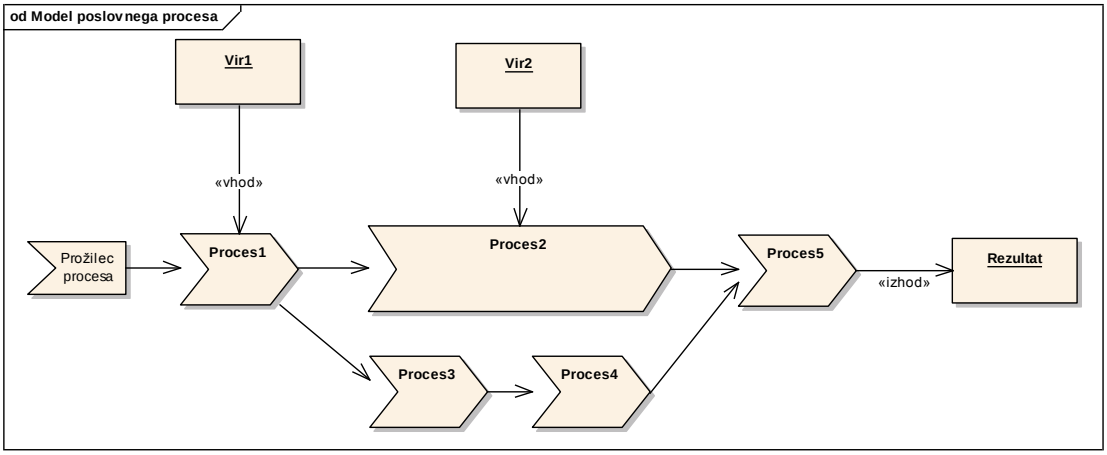
Na sistem najprej pogledamo s širšega zornega kota tako, da najprej proučimo strukturo organizacije, ki jo narišemo z razrednim diagramom.

Slika 5: Primer strukture organizacije



Z modeliranjem poslovnega procesa nato ugotavljamo, kje v poslovnem procesu bo bodoči informacijski sistem umeščen, s katerimi drugimi obstoječimi informacijskimi sistemi bo komuniciral in komu od zaposlenih bo bodoči informacijski sistem koristil, pri čemer se v prvi fazi ne obremenjujemo z iskanjem meja sistema in zato pri modeliranju širšega pogleda na sistem upoštevamo tako neposredne kot posredne uporabnike. Za modeliranje poslovnega procesa uporabljamo analitični diagram.

Slika 6: Primer modela poslovnega procesa



Pri analizi modela poslovnega procesa lahko naletimo na področja, kjer bi bilo smiselno razmisliti o prenovi procesov, analiziramo lahko poslovna pravila, definiramo potrebne vhodne in izhodne dokumente…

Seveda pa modeliranje poslovnega procesa ni vedno nujno. V primeru, ko pripravljamo razširitev funkcionalnosti obstoječega sistema kot samostojni projekt, lahko ta korak spustimo. Ravno tako lahko ta korak spustimo v primeru, ko sistem zelo dobro poznamo ali pa ga imamo celo že orisanega v katerem od sorodnih projektov. V tem primeru je dovolj že, če preverimo, ali so v času odkar smo proces modelirali do sedaj, nastale kakšne razlike. Če so, jih vnesemo v model poslovnega procesa. V tem koraku pripravimo tri dokumente, in sicer

organizacijsko shemo, ki jo rišemo s pomočjo razrednega diagrama, model poslovnega procesa, orisan z notacijo analitičnega diagrama in prvo videnje umeščenosti bodočega informacijskega sistema v okolje podjetja. Model poslovnega procesa rišemo na konceptualnem nivoju, a v primeru, ko je zaradi razumevanja potrebna podrobnejša razlaga, ga uporabimo tudi na nižjem nivoju, pri čemer z dogovorjenimi oznakami opišemo da gre za diagram, ki podrobneje razlaga neko aktivnost z višjega nivoja.

6.3.2 Ugotavljanje funkcionalnih in nefunkcionalnih zahtev

Naša glavna naloga pri ugotavljanju zahtev je zbiranje podatkov o trenutno delujočem sistemu, pa naj si gre za ročni ali računalniško podprt sistem. Iščemo torej pogoje za izpolnjevanje poslovnih zahtev. Do teh informacij si lahko pomagamo tako, da poskusimo odgovoriti na naslednja vprašanja (Shelly et al., 1998, str. 3):

- Katere poslovne funkcije podpira trenutni sistem?
- Katere trenutne pa tudi bodoče poslovne funkcije naj bi podpiral novi sistem?
- Katere postopke in katere dokumente uporabljajo uporabniki trenutnega sistema?
- Kdo izvaja posamezne operacije?
- Katere transakcije izvaja sistem?
- Katere informacije sistem generira in katere informacije potrebuje?
- Katere informacije naj bi sistem zagotavljal uporabnikom in managerjem?
- Katere so slabosti in katere so prednosti trenutnega sistema?
- Katere postopke bi lahko opustili?

Z odgovori na ta vprašanja si lahko ustvarimo sliko o trenutnem sistemu in o izboljšavah, ki jih lahko uvedemo z novim sistemom. Kar nekaj študij govori o tem, da je eden glavnih vzrokov za neuspeh projektov, ki se ukvarjajo z razvojem informacijskih sistemov, ravno neustrezno zajemanje zahtev.

Naslednji zelo pomemben faktor, ki vpliva na uspeh ali neuspeh projektov, pa je premajhna vpletenost uporabnikov (Arlow et al., 2002, str. 49). Temu se lahko poskusimo izogniti tako, da uporabnike vpletemo v razvoj informacijskega sistema že pri pripravi zahtev. Uporabniki so gotovo tisti, ki lahko najbolje zahteve opišejo, hkrati pa zaradi sodelovanja pri razvoju, sistem kmalu posvojijo in na tak način se izognemo kasnejšemu zavračanju sprememb, ki jih uvajanje novega informacijskega sistema prinaša.

Pri ugotavljanju zahtev, ki naj jih izpolni informacijski sistem, smo pozorni na dve vrsti zahtev: *zahteve, ki se nanašajo na funkcionalnost sistema* in govorijo o obnašanju sistema ter *zahteve, ki niso funkcionalne*, govorijo pa o omejitvah, na katere moramo biti pozorni, ko sistem načrtujemo in gradimo. Nefunkcionalna zahteva je na primer zahtevani programski jezik, v katerem naj bo zapisana koda. Nefunkcionalna zahteva je lahko zahteva po točno določenem številu nepravilnih prijav na sistem, preden sistem uporabniško ime zaklene za uporabo, in podobno.

Pri ugotavljanju funkcionalnih in nefunkcionalnih zahtev izdelamo celoten nabor funkcionalnih in nefunkcionalnih zahtev sistema na visokem nivoju abstrakcije oziroma s poslovne perspektive. Zajeti moramo zahteve za prvo generacijo sistema kot tudi zahteve za predvidene prihodnje generacije. Da bi lahko zagotovili učinkovit pregled zahtev, moramo upoštevati celoten življenjski cikel od zasnove preko izdelave in vzdrževanja pa do popravkov oziroma zastarelosti.

6.3.3 Prototip arhitekture informacijskega sistema

Pri prototipiranju gre za strategijo, ki jo lahko uporabljamo pri večini aktivnosti. Namen prototipiranja je iskanje informacij, potrebnih za sprejem odločitev. Zmanjšuje tveganje glede napačnih odločitev pri zajemanju zahtev in načrtovanju arhitekture sistema.

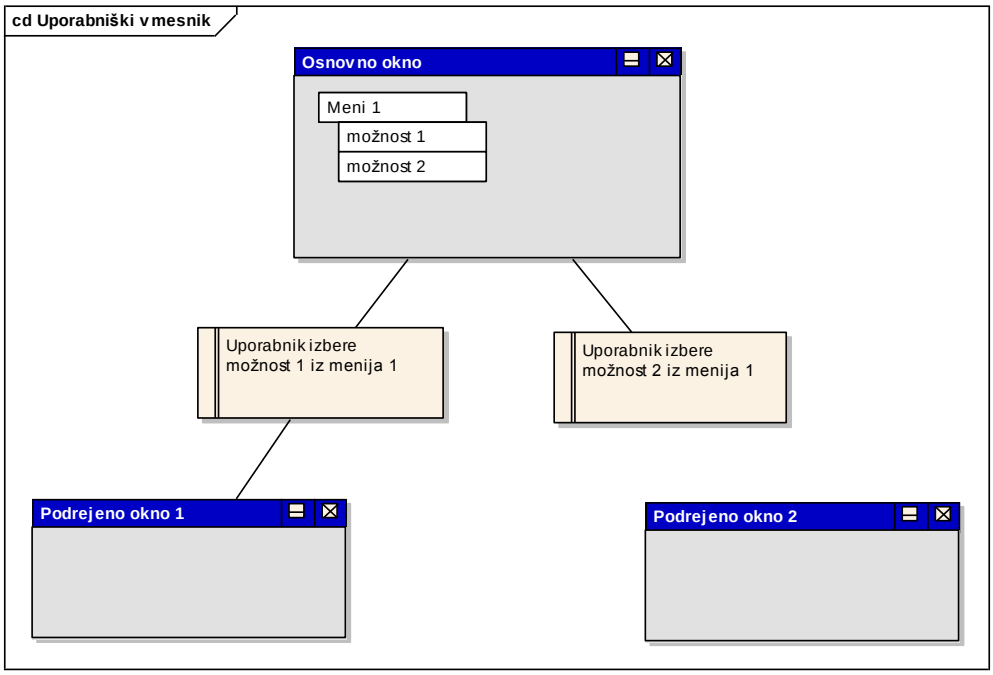
Prototip je predhodna in namenoma nepopolna verzija sistema. Navadno ni prepričljivo izdelan in ne zadosti zahtevam robustnosti končnega proizvoda. S svojim obstojem povečuje zaupanje, da je želeni produkt moč izdelati.

Prototip arhitekture informacijskega sistema pripravljamo le v primeru, ko je to zaželeno zaradi morebitnih ugotovljenih dejavnikov tveganja povezanih z arhitekturo.

6.3.4 Prototip uporabniškega vmesnika

S pomočjo orodja (razvojno ali risarsko) izdelamo prototip uporabniškega vmesnika, ki kaže samo na pristop k implementaciji realnega vmesnika. Pri tem se osredotočimo na uporabniške zahteve glede vmesnika (če obstojajo) oziroma na predvideno okolje uporabe sistema in tem zahtevam prilagodimo prototip uporabniškega vmesnika. Ker je prototip namenjen izključno identifikaciji novih zahtev in predstavlja le komunikacijsko sredstvo, je potrebno jasno podati mejo med prototipom in dejanskim uporabniškim vmesnikom, ki bo razvit kasneje. Prototip lahko pripravimo s katerim od razvojnih orodij, lahko pa je tudi skica, ki jo zrišemo na papir.

Slika 7: Primer uporabniškega vmesnika



Ker je prototip izključno namenjen delu z uporabnikom in identificiranju novih uporabniških zahtev, uporabniški vmesnik verificiramo s pomočjo uporabnika. Pri tem se ne osredotočamo na implementacijo vmesnika, ampak na njegovo funkcionalnosti.

6.3.5 Nadaljevanje priprave projektnega okolja

V pripravo projektnega okolja v tej fazi vključujemo pripravo rokovnika projekta, razdeljevanje nalog sodelujočim na projektu, pripravo razvojnih orodij ter morebitnih dodatnih delovnih mest.

6.3.6 Projektni slovar

Skozi vso fazo elaborata beležimo projektni slovar, ki nam pomaga v komunikaciji z naročnikom in nam je v oporo v naslednji fazi, pri sami izdelavi informacijskega sistema.

6.3.7 Mejnik in dokumentacija iz faze elaborata

Osnovni cilj faze elaborata je priprava robustne osnovne arhitekturne rešitve, ki pokrije vse naročnikove zahteve. Zato sodelujoči na projektu ob zaključku faze elaborata pregledajo stanje na projektu. Sodelujoči na projektu se morajo na tem mejniku strinjati o naslednjih točkah:

- stabilnost vizije (realistična vizija projekta),
- poslovne zahteve (soglasje o ugotovljenih poslovnih zahtevah),
- stabilnost arhitekture (soglasje o arhitekturi, ki izpolnjuje poslovne zahteve),
- dejavniki tveganja (razumevanje in sprejem dejavnikov tveganja za fazo izdelave ter strategij za njihov odpravo),
- stroški (sprejem dosedanjih stroškov in ocene stroškov za fazo izdelave),
- realistična ocena možnosti uspeha projekta (soudeleženi na projektu se strinjajo z ugotovitvijo o možni uspešni realizaciji projekta na pripravljenih osnovah),
- rokovnik naslednje faze (sprejem rokovnika za fazo izdelave),
- ujemanje načrtovane arhitekture s trenutno arhitekturo (soglasje o komplementarnosti arhitektur).

Dokumentacija, ki jo zabeležimo v fazi elaborata je najobsežnejša, zajema pa:

- ➔ zapisniki sestankov projektne skupine;
- ➔ organizacijsko shemo organizacije;
- ➔ model poslovnih procesov;
- ➔ nabor funkcionalnih zahtev;
- ➔ nabor nefunkcionalnih zahtev;
- ➔ verificirani prototip uporabniškega vmesnika;
- ➔ rokovnik projekta;
- ➔ verificirano dopolnjeno osebno izkaznico projekta;
- ➔ projektni slovar.

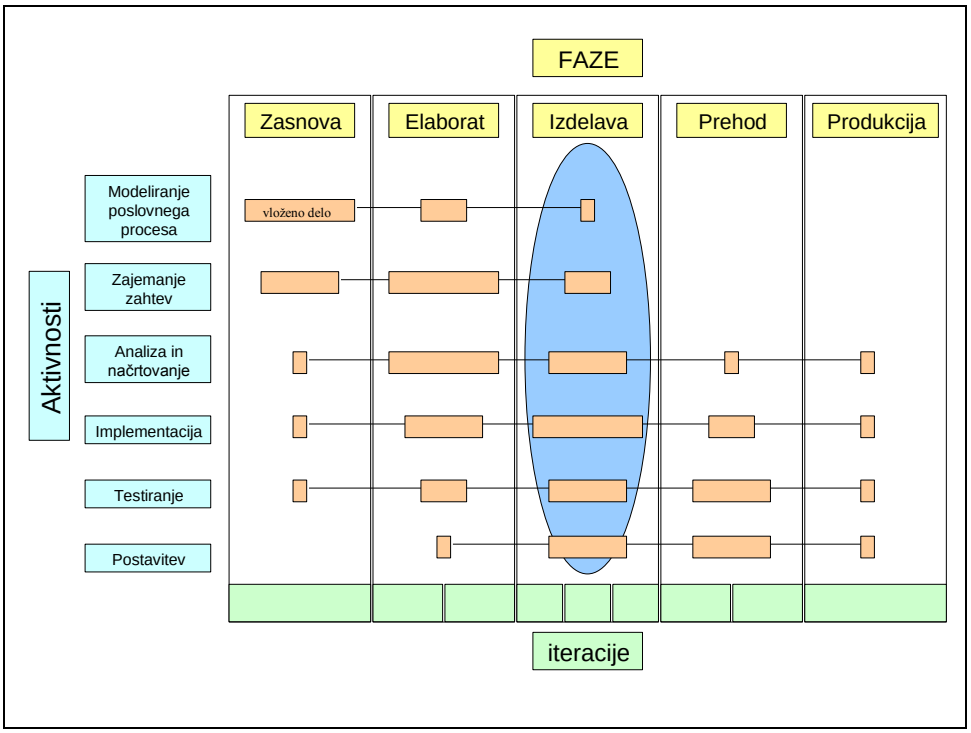
6.4 Izdelava

V fazi izdelave zgradimo sistem do točke, ko je sistem pripravljen za testiranje pred prevzemom. Zato se osredotočimo na zahteve, ki jih prioritetno specificiramo, analiziramo, kreiramo rešitve, ki tem zahtevam zadostijo, rešitve kodiramo in testiramo razvito programsko opremo. V se to počnemo v seriji iteracij. Vsaka iteracija je mini projekt, kar pomeni, da za vsako iteracijo izvedemo analizo, načrt, kodiranje, testiranje in integracijo za primere uporabe, ki smo jih določili za določeno iteracijo.

Iteracije v tej fazi so tako inkrementalne kot iterativne. Inkrementalne so po svoji funkciji. V vsaki iteraciji zgradimo primer uporabe, ki smo ga razvili v prejšnjih iteracijah. Zato pravimo, da so iteracije inkrementalne. Iterativne pa zato, ker v vsaki iteraciji razvijemo in popravljamo obstoječo kodo, ker želimo doseči večjo elastičnost.

Integracija pa je nepretrgan proces, ki jo v začetku izvajamo kot del vsake iteracije, A kasneje lahko ugotovimo, da je bolje integracijo izvajati pogosteje in se na ta način izogibamo možnosti, da bi nam integracija prestavljala težavo.

Slika 8: Faza izdelave



6.4.1 Identifikacija akterjev in primerov uporabe

Funkcionalne zahteve so temelj vsakega sistema. Govorijo o tem, kaj naj bi sistem počel, pri tem pa nas zaenkrat še ne zanima, kako naj bi sistem to počel. Ni nam treba še razmišljati o tem, kako bomo posamezno funkcionalnost sistema realizirali. Pri ugotavljanju zahtev se torej osredotočimo na identificiranje uporabnikov oziroma akterjev in iskanje meja sistema. Nato poiščemo primere uporabe, jih specificiramo in kreiramo scenarije za primere uporabe. Diagram, ki prikazuje naše ugotovitve, je diagram primerov uporabe. Diagram vsebuje štiri komponente: akterje, primere uporabe, povezave in sistemske meje. Akterji predstavljajo vloge, ki jih imajo ljudje oziroma stvari, ki uporabljajo sistem. S primeri uporabe opišemo stvari, ki jih akter lahko s sistemom počne. Povezave predstavljajo pomenske povezave med akterji in primeri uporabe. Sistemske meje pa so meje, ki omejujejo sistem, ki ga modeliramo. Uporabnik pomeni z vidika UML notacije le stereotip razreda s svojo posebno ikono. Zato lahko uporabimo pri zapisu primerov uporabe tako figurico »možička« kot »pravokotnik«. Možna kombinacija pa je, da pri vlogah, ki jih bomo imeli ljudje, uporabimo figurico, v primeru, ko opisujemo vloge, ki jih bodo imeli drugi sistemi, pa uporabimo »pravokotnik«. Uporabnike sistema lahko identificiramo tako, da si poskusimo odgovoriti na naslednja vprašanja:

- Kdo ali kaj uporablja sistem?
- Kakšne vloge prevzemajo pri delu s sistemom?
- Kdo namešča sistem?
- Kdo starta in kdo ugaša sistem?
- Kdo vzdržuje sistem?
- Kateri so drugi sistemi, ki uporabljajo sistem?
- Kdo dobiva informacije od sistema?
- Kdo zagotavlja informacije v sistemu?
- Ali se dogajajo kakšne na urnik vezane aktivnosti?

Pri modeliranju akterjev smo pozorni na naslednje točke:

- Akter se vedno nahajajo izven sistema – mi jih torej ne nadzorujemo.

- Akter prihajajo s sistemom neposredno v stik. Ta trditev nam je v pomoč takrat, ko iščemo meje sistema.
- Akterji predstavljajo vloge, ki jih prevzemajo ljudje in stvari, ko pridejo v stik s sistemom. To niso določeni ljudje in določene stvari.
- Človek ali stvar lahko v stiku s sistemom prevzame več vlog. Te vloge lahko izvaja simultano ali preko določenega obdobja.
- Vsakemu akterju določimo kratko ime, ki opisuje njegovo funkcijo v poslovnem procesu.
- Vsakega akterja tudi na kratko opišemo s poslovnega vidika.
- Kadar modeliramo sistem, ki določen trenutek (na določeno uro) izvede nek sklop aktivnosti, uvedemo kot akterja še čas.

Primer uporabe rišemo kot oval. Ime primera uporabe lahko pišemo v oval ali izven ovala. Primer uporabe predstavlja sklop aktivnosti, ki naj jih izvede sistem na uporabnikovo zahtevo. Gre za različne načine na katere uporabnik uporablja sistem. Primere uporabe vedno proži uporabnik in vedno jih opisujemo z uporabnikovega zornega kota. Opis primera uporabe pripravimo na osnovi intervjuja z uporabnikom. Skozi intervju naj uporabnik odgovori na nekaj ključnih vprašanj:

- V katerih vlogah se pojavlja uporabnik in kakšna je njegova naloga v posamezni vlogi?
- Ali se akter pojavlja v več kot eni vlogi?
- Ali akter kreira, popravlja, briše ali shranjuje podatke v sistem?
- Kakšne informacije potrebuje akter pri izvajanju svoje naloge?
- Kakšne so možne napake, ki jih lahko akter “zagreši” pri uporabi sistema?

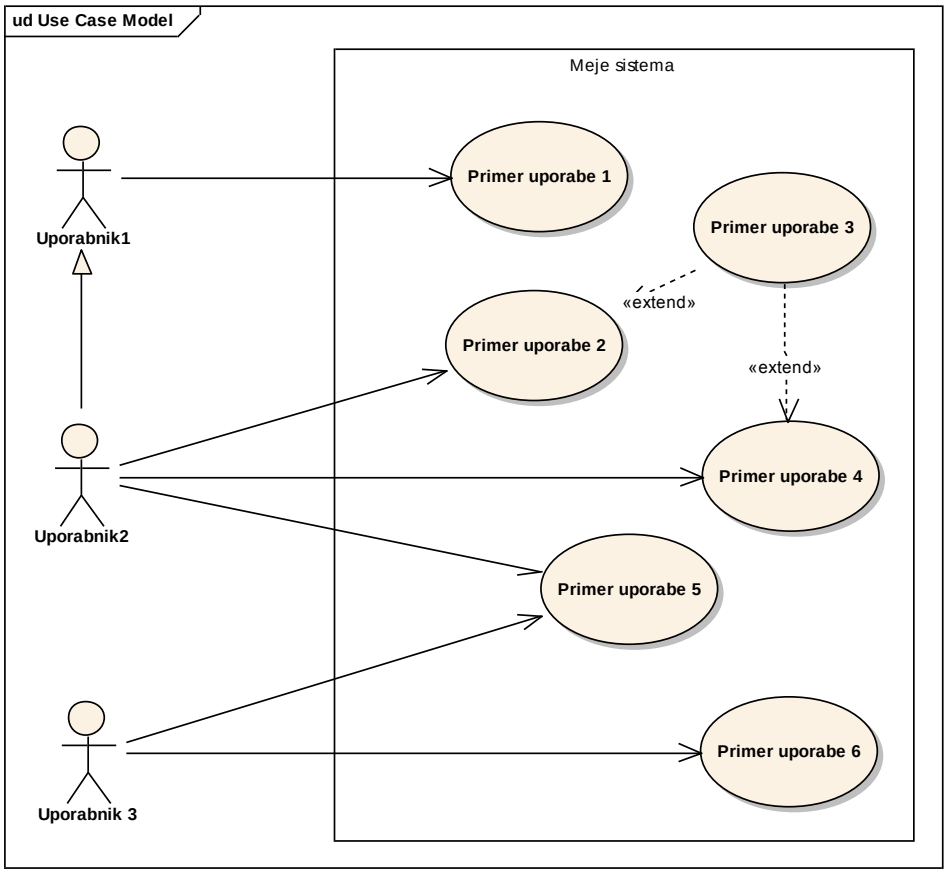
Primer uporabe nam govori o tem, na kakšen način uporabnik uporablja sistem in zato tudi scenarij opisujemo z njegovega zornega kota (Arlow, Neustadt. 2002, str. 58 – 62). Opis primera uporabe in scenarij, ki je del tega opisa, pripravimo po naslednjem obrazcu (Arlow, Neustadt. 2002, str. 71 – 75), ki smo ga opisali v naslednji tabeli.

Tabela 1: Standardni obrazec za opis primera uporabe

Atribut primera uporabe	Opis
Ime:	Uporabno ime za vsak primer uporabe.
ID primera uporabe:	Šifra primera uporabe.
Opis:	Namen primera uporabe.
Akter:	Akter, ki bo izvajal aktivnosti, opisane v primeru uporabe.
Vključitve:	<ID primera uporabe, ime primera uporabe> Vsi primeri uporabe, ki jih primer uporabe uporablja «uses» oziroma «includes».
Razširitve:	<ID primera uporabe, ime primera uporabe> Vsi primeri uporabe, ki uporabljajo ta primeru uporabe kot razširitev.
Predpogoj:	Pogoj, ki mora biti izpolnjen, da se primer uporabe izvede.
Scenarij:	Popoln opis vsega, kar se dogaja.
Alternativni scenarij:	Kateri koli dogodek, ki se lahko zgodi in ni zajet v običajnem toku aktivnosti. Tu vključimo vse pogoje, ki povzročijo napake.
Merljiv rezultat:	Opis sistema, ko se primer uporabe zaključi.

Odločili smo se, da mora vsak opis primera uporabe vsebovati najmanj vse podčrtane točke, če pa je za razumevanje potrebno, lahko sodelavec, ki izvaja intervjuje, doda še katero od točk, ki niso podčrtane. Diagram primerov uporabe predstavlja primarni vir za modeliranje razrednega diagrama, zato so se odločili, da tej faz posvetimo posebno skrb in jo običajno izvajajo sodelavci, ki so z domeno, ki jo modeliramo, domači. Kot dodaten dokument pa ves čas nastaja še projektni slovar, kamor beležimo poslovne termine in definicije.

Slika 9: Primer diagrama primerov uporabe



Pri risanju diagrama primerov uporabe se ne trudimo zajeti vseh podrobnosti, ker primeri uporabe ne pokažejo celotne slike sistema.

Primeri uporabe so osnovno orodje pri ugotavljanju zahtev ter pri načrtovanju in kontroli iteracij projekta. Zelo uporabno orodje za sledenje primerom uporabe ter pokritost formalnih zahtev, je matrika, v kateri v vrsticah zapišemo zahteve, ki naj jih sistem izpolnjuje, v stolpcih pa nanizamo primere uporabe. V matriki si nato označujemo primere uporabe, ki pokrivajo določeno zahtevo. Tako lahko kadar koli preverimo, ali smo že pokrili vse zahteve oziroma, če najdemo primer uporabe, ki ne zadosti nobeni zahtevi, ponovno pregledamo nabor zahtev, ki je verjetno pomanjkljiv.

Tabela 2: Primer matrike, s katero sledimo zahtevam

	Primer uporabe 1	Primer uporabe 2	Primer uporabe 3	Primer uporabe 4	Primer uporabe 5	Primer uporabe 6
Zahteva 1		X				
Zahteva 2		X	X	X		
Zahteva 3				X		
Zahteva 4						
Zahteva 5					X	

Večino primerov uporabe bomo zajeli v prvi fazi projekta, a verjetno jih bomo našli tudi v kasnejših fazah še nekaj.
Primeri uporabe predstavljajo zunanji pogled na sistem, zato ne iščemo korelacij med njimi in razredi v sistemu.

Pri pripravi primerov uporabe se osredotočimo na pripravo besedila ob primeru uporabe. Ta bo dragoceno gradivo pri pripravi razrednega diagrama.

6.4.2 Analiza zahtev

Z analizo zahtev in načrtovanjem arhitekture sistema zgradimo most od ugotovljenih zahtev do implementacijske specifikacije. Z analizo zahtev ugotavljamo funkcionalnosti, ki naj jih pokriva bodoči informacijski sistem, pri tem pa običajno velja, da se ne ukvarjamo z ne funkcionalnimi omejitvami in okoljem, kjer bomo sistem implementirali. Opozoriti velja, da se analiza mnogokrat prekriva z ugotavljanjem zahtev, govorimo lahko celo, da sta v določenih primerih ugotavljanje zahtev in analiza soodvisni. Mnogokrat se nam zgodi, da analiziramo zahteve zato, da si razjasnimo, kaj pomenijo in da odkrijemo spregledane in izkrivljene zahteve. Kar pomeni, da z analizo zahtev odkrivamo nove zahteve.

Z analizo zahtev pripravimo matriko odgovornosti, ki povezuje organizacijsko shemo oziroma uporabnike in primere uporabe in analitičnega modela, ki pojasnjuje, kaj naj sistem počne, podrobnosti o tem, kako naj to počne, pa pustimo za naslednji korak, načrtovanje arhitekture sistema.

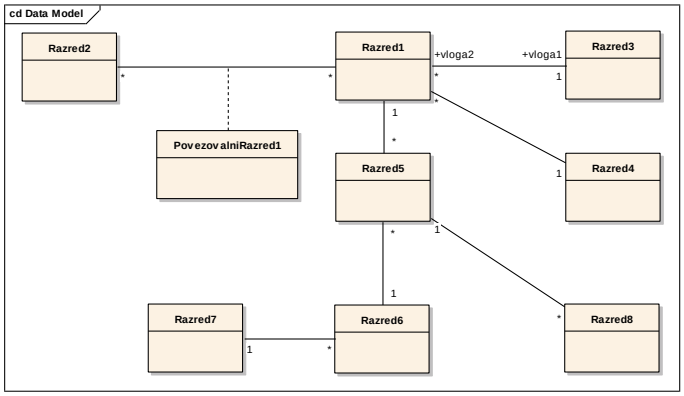
Matrika je ena od zelo pogostih tehnik, s katero lahko na enostaven način izrazimo odnose med običajno dvema dimenzijama. Pri nas se matrike poslužujemo pri opisovanju odnosov med uporabniki in primeri uporabe, za katere so uporabniki odgovorni. Matriko odgovornosti pripravljamo tako, da v vrstice nizamo uporabnike, v stolpce primere uporabe in v stičišča vpisujemo odnose. Običajno nas zanimata dve možnosti, in sicer kateri uporabnik izvaja kateri primer uporabe in kdo je za ta primer uporabe odgovoren.

Tabela 3: Primer matrike odgovornosti

	Primer uporabe 1	Primer uporabe 2	Primer uporabe 3	Primer uporabe 4	Primer uporabe 5	Primer uporabe 6
Uporabnik 1	I, O	O	O	O		
Uporabnik 2		I		I		
Uporabnik 3				I	I, O	I, O

Analitični model rišemo z razrednim diagramom. Tehnika risanja razrednega diagrama je središče vseh objektno orientiranih metod, kajti vsaka objektno orientirana metoda vključuje neko variacijo te tehnike. Z razrednim diagramom predstavljamo statično naravo konceptov, tipov in razredov. S koncepti prikazujemo uporabnikov pogled na svet, tipi predstavljajo vmesnike in programske komponente, z razredi pa predstavimo implementacijo programskih komponent. Z razrednim diagramom opisujemo različne tipe objektov v sistemu in različne statične povezave med njimi. Opazujemo dve osnovni različici statičnih povezav: povezave oziroma asociacije in podtipe.

Slika 10: Primer razrednega diagrama



Pri analizi zahtev nas zanima konceptualni nivo in osnovno pravilo pri pripravi takšnega modela se glasi: omejimo se le na tiste razrede, ki jih najdemo tudi v besedišču problemske domene. Vse odločitve, o načrtovanju in implementaciji sistema pa obravnavamo v nadaljnjih korakih. Če opisujemo konceptualno perspektivo, potem rišemo diagram, ki predstavlja koncepte domene, ki jo raziskujemo. Te koncepte sicer lahko intuitivno povežemo z razredi, ki jih implementirajo, ne moremo pa jih direktno izluščiti, niti to ni potrebno. Če nam je domena, v kateri pripravljamo informacijski sistem dobro znana, lahko ta korak tudi izpustimo.

Pri pripravi analitičnega modela informacijskega sistema vedno uporabljamo jezik problemske domene. Model, ki ga kreiramo, moramo pripraviti tako, da bo že sam povedal zgodbo. Vsak od pripravljenih diagramov nam osvetli neko želeno obnašanje sistema. V nasprotnem primeru nam diagram ni potreben. Analitični model pripravljamo na konceptualnem nivoju in zato smo pozorni na to, da se ne zapletamo v podrobnosti temveč ustvarjamo celovito podobo sistema. Zavedamo se tega, da mora model zadovoljiti vse vpletene v pripravo sistema: tako poslovne uporabnike, kot načrtovalce in razvijalce. Vse te zahteve izpolnjujemo tako, da vključujemo v proces sodelujoče pri projektu, kadar koli bi nam njihovi nasveti in pogledi na sistem koristili ter že pripravljene modele pogosto pregledamo skupaj s projektno skupino.

6.4.3 Arhitektura sistema

Še vedno se ukvarjamo s pripravo analitičnega modela sistema ter z analizo zahtev. Usmerimo pa se v načrtovanje arhitekture sistema, kjer vzamemo v zakup tudi prej izpuščene ne funkcionalne zahteve in omejitve, ki nam jih nalaga informacijska arhitektura. V tem koraku se ukvarjamo tako z definiranjem arhitekture, z analizo obnašanja sistema, torej s funkcionalnostjo sistema, načrtovanjem komponent in načrtovanjem baze. Vsak sistem je zgodba zase, zato nimamo natančnega recepta, kako pripraviti analitični model sistema, temveč podajamo le smernice za pripravo le-tega.

Na arhitekturo sistema vplivajo primeri uporabe (funkcionalne zahteve), nefunkcionalne zahteve, izkušnje razvojne skupine in druge omejitve (npr. obstoječi sistem, določena podatkovna baza).

Namen arhitekturnega opisa:

- *razumeti sistem* – kompleksno obnašanje sistema in kompleksno okolje, kamor bomo sistem namestili, zahtevajo podrobno razumevanje. Seznanjeni moramo biti z uporabo novih tehnologij, različnih proizvodov in načinom koordiniranja več projektov, ki tvorijo celoto,
- *organizirati razvoj* – znano je, da se z velikostjo projekta povečuje delež komunikacije, kar ima za posledico zmanjšano produktivnost. Delitev na podsisteme in jasna določitev vmesnikov zmanjša potrebo po komunikaciji, saj lahko posamezne komponente sistema razvijamo ob minimalni komunikaciji z ostalimi razvojnimi skupinami,

- *spodbujati ponovno uporabo* – če želimo izvajati ponovno uporabo, je potrebno doseči standardizacijo. Dobra arhitektura in jasni vmesniki predstavljajo korak v tej smeri,
- *izboljšati sistem* –razvoj sistema moramo obravnavati kot sistem sprememb, ki ob slabi arhitekturi močno vplivajo na fleksibilnost sistema, v primeru dobro zasnovanega sistema pa omogočajo lokalne spremembe v posameznih podsistemih oz. komponentah brez vpliva na druge podsisteme.

Iz seznama nefunkcionalnih zahtev izločimo tiste zahteve, ki vplivajo na arhitekturo sistema. Pri tem se osredotočimo na zahteve glede obstoječih sistemov (sistemi, s katerimi bo potreben soobstoj in obstoječe podatkovne baze), zahteve glede porazdeljenosti, okolje, kamor bomo namestili sistem, zakonske omejitve. Upoštevati moramo tudi vrsto sistema, ki ga gradimo in zagotoviti primerno arhitekturo, ki bo zadovoljevala zahteve (npr. za kritične sisteme vpeljavo redundantnih elementov).

Izbrane nefunkcionalne zahteve zapišemo v obliki seznama, kjer podamo nefunkcionalno zahtevo in njen vpliv na arhitekturo sistema. V nadaljevanju bomo dodali še informacijo o tem, kako smo vpliv upoštevali.

6.4.3.1 Izgradnja logične arhitekture

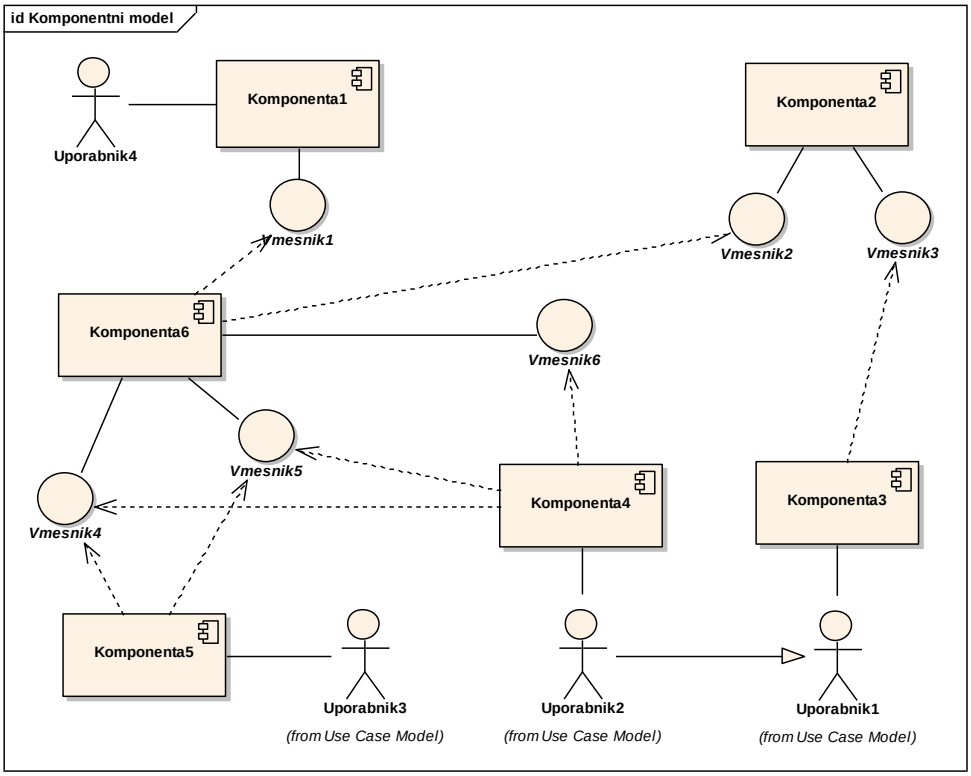
Logično arhitekturo informacijskega sistema pripravimo na podlagi spoznanj in dognanj iz zajemanja in analize zahtev. Na osnovi informacij, ki jih pridobimo iz naštetih dveh korakov določimo logične.komponente.

Napotki za razpoznavanje komponent:

- potencialne komponente lahko izdelamo s preureditvijo razrednega diagrama analize tako, da razrede združimo v logično celoto z nizko stopnjo zunanje povezanosti (z instancami združenih razredov dobimo komponento),
- pojavi se lahko potreba po posebnih komponentah, ki bodo zadostile nefunkcionalnim zahtevam (dodaten strežnik za zagotovitev neprekinjenega delovanja sistema),
- pregledamo obstoječe knjižnice ponovno uporabnih komponent in obstoječe sisteme, kjer poskušamo poiskati koristne komponente (takšne ponovno uporabne ali obstoječe komponente lahko uporabimo v procesu načrtovanja z obstoječimi vmesniki ali spremenjene),
- če primeri uporabe predvidevajo interakcijo med sistemom in ljudmi bomo potrebovali komponento uporabniškega vmesnika, ki jo načrtujemo v skladu z navodili procesa izdelave uporabniškega vmesnika,
- za podatke, ki so skupni več primerom uporabe, je potrebno ugotoviti, ali gre za trenutne podatkovne strukture ali za podatke, ki jih je potrebno shraniti med uporabami sistema in je za to potrebna komponenta podatkovne baze (komponenta podatkovne baze naj bo načrtovana v skladu s procesom načrtovanja podatkovne baze).

Logično arhitekturo informacijskega sistema rišemo s komponentnim diagramom.

Slika 11: Primer komponentnega diagrama

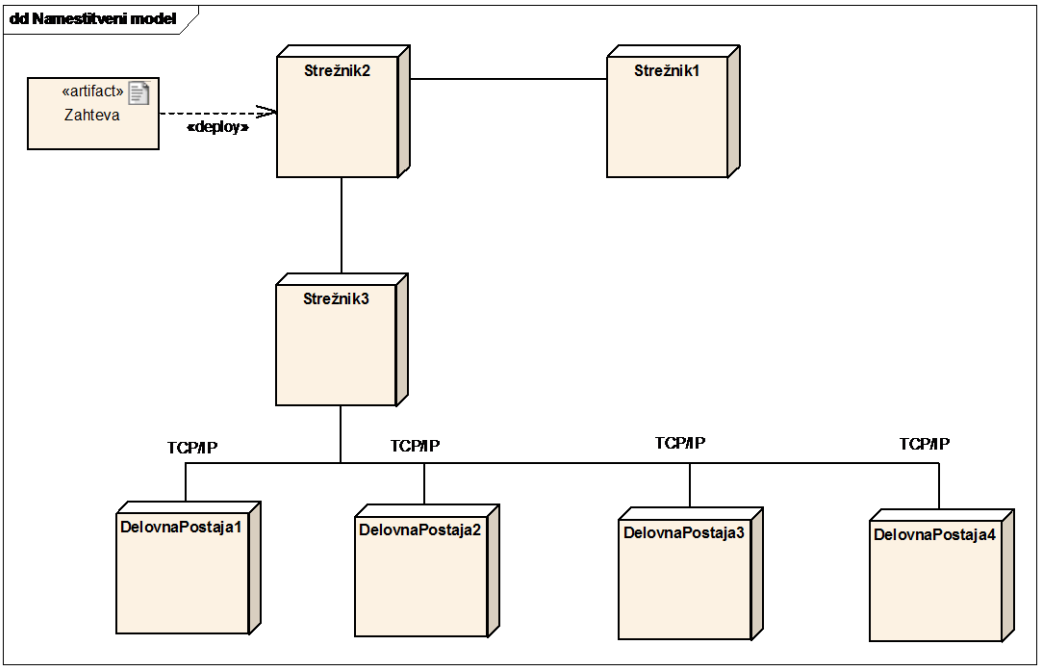


6.4.3.2 Izgradnja fizične arhitekture

Na podlagi logične arhitekture izdelamo eno ali več fizičnih arhitektur. Vsako alternativo ovrednotimo.

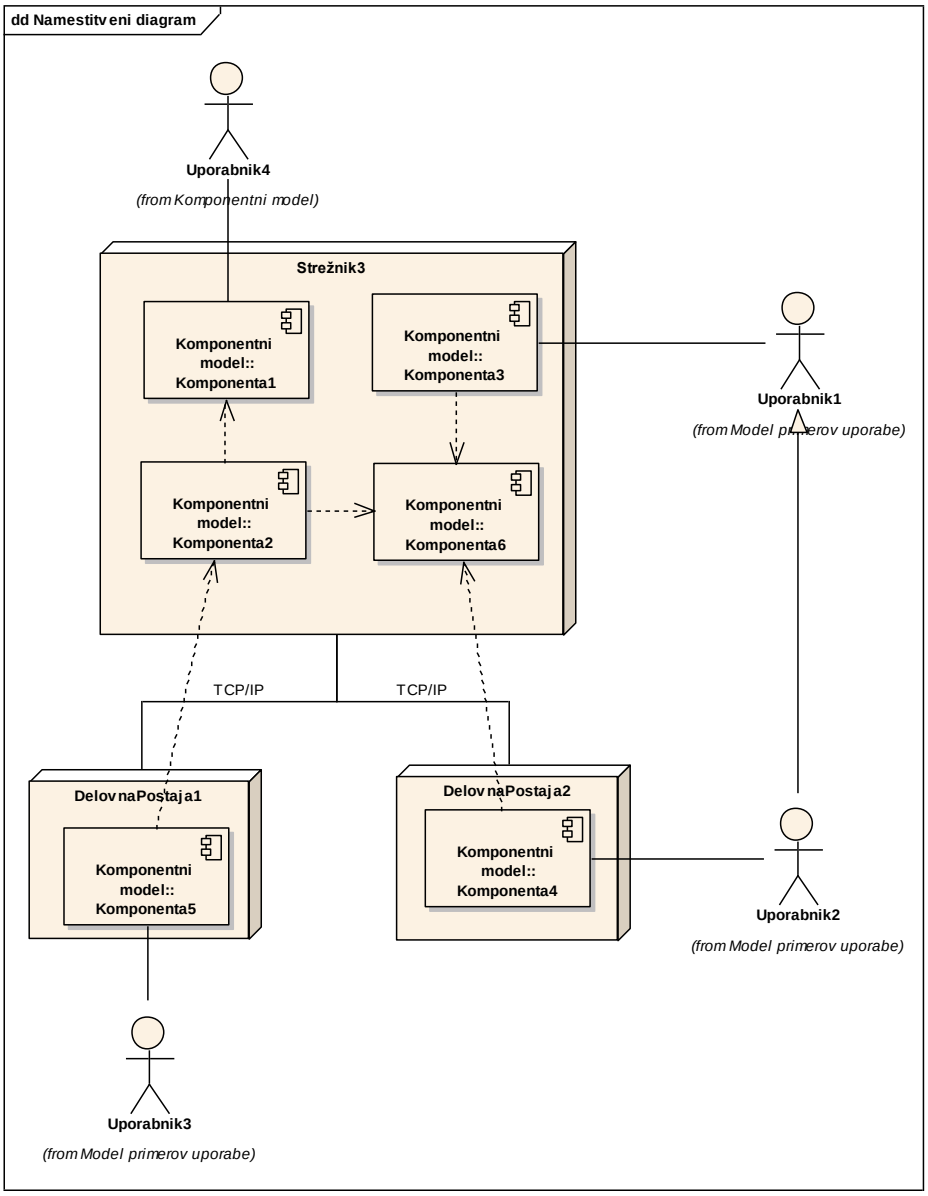
Izberemo tehnološko osnovo, katero želimo vključiti v fizično arhitekturo sistema (vrsta podatkovne baze, model porazdeljenosti, pristop varnosti). Fizično arhitekturo rišemo z namestitvenim diagramom.

Slika 12: Primer namestitvenega diagrama



Diagram, ki predstavljata logično arhitekturo in diagram, ki predstavlja fizično arhitekturo lahko zdržimo v en diagram, iz katerega je razvidna tako fizična arhitektura kot tudi porazdelitev logičnih komponent po posameznih elementih fizične arhitekture.

Slika 13: Primer namestitvenega in komponentnega diagrama



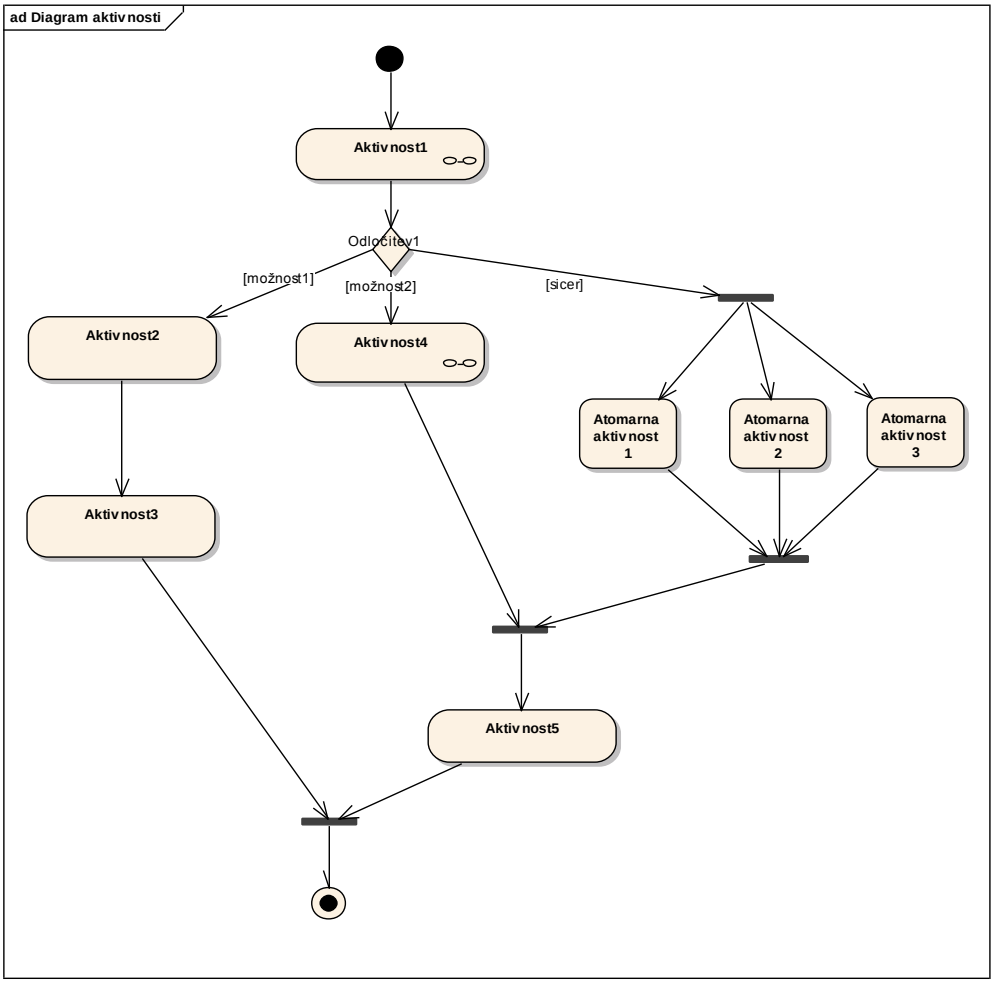
6.4.4 Načrtovanje algoritmov

Definiramo algoritme, ki so skladni z ugotovitvami iz faze analize. Pred načrtovanjem algoritmov pregledamo in ovrednotimo obstoječe vzorce, ki bi jih lahko uporabili. Algoritme predstavimo z diagrami sodelovanja, diagrami zaporedja, diagrami aktivnosti ali diagrami stanj. V algoritem lahko vključimo tudi večnitnost in model pošiljanja sporočil.

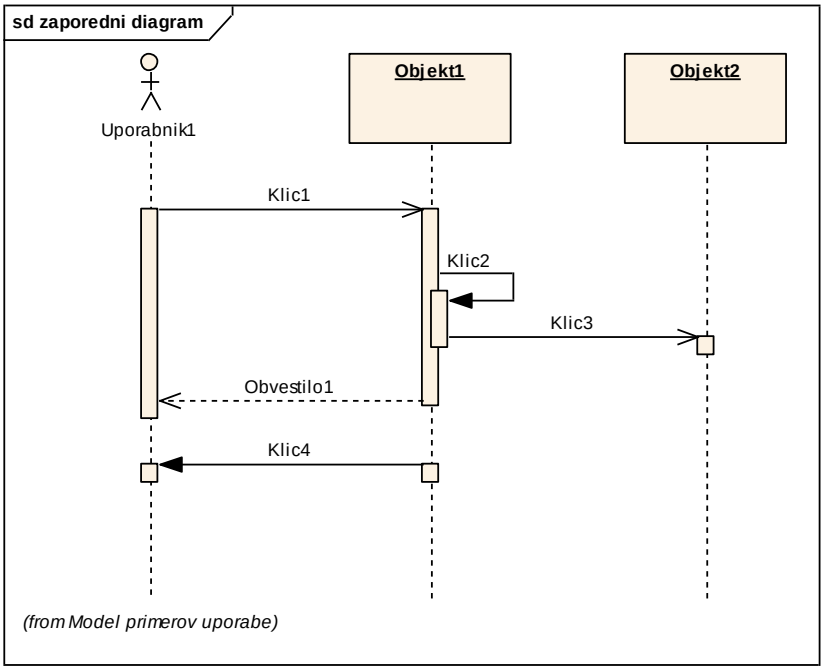
Na podlagi funkcionalnih zahtev in opisov scenarijev primerov uporabe določimo kompleksne algoritme. Ti algoritmi izhajajo iz informacijske domene.

Na podlagi identificiranih kompleksnih algoritmov izdelamo diagrame aktivnosti, v katerih lahko modeliramo hkratnost in sinhronizacijo, po potrebi pa posamezne aktivnosti razdelamo s podrejenimi diagrami aktivnosti oziroma diagrami sodelovanja in / ali diagrami zaporedja. Diagrame stanja uporabimo na primer za opis življenjskega cikla pojavitev objekta.

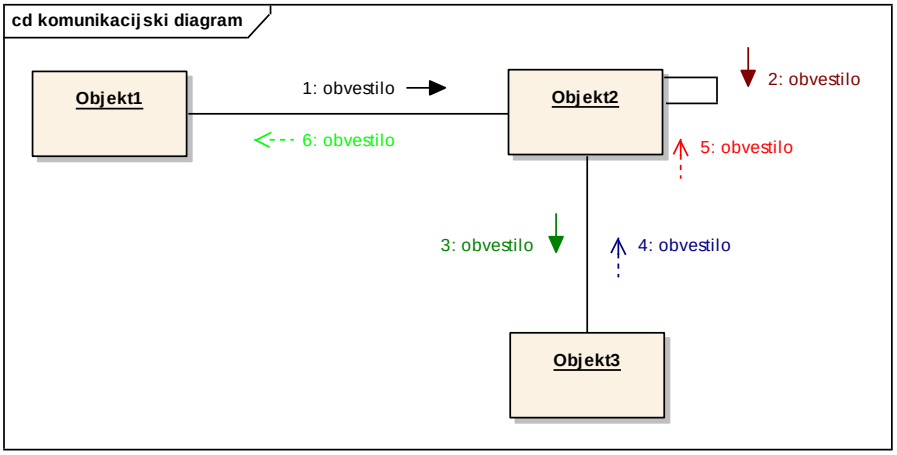
Slika 14: Primer diagrama aktivnosti



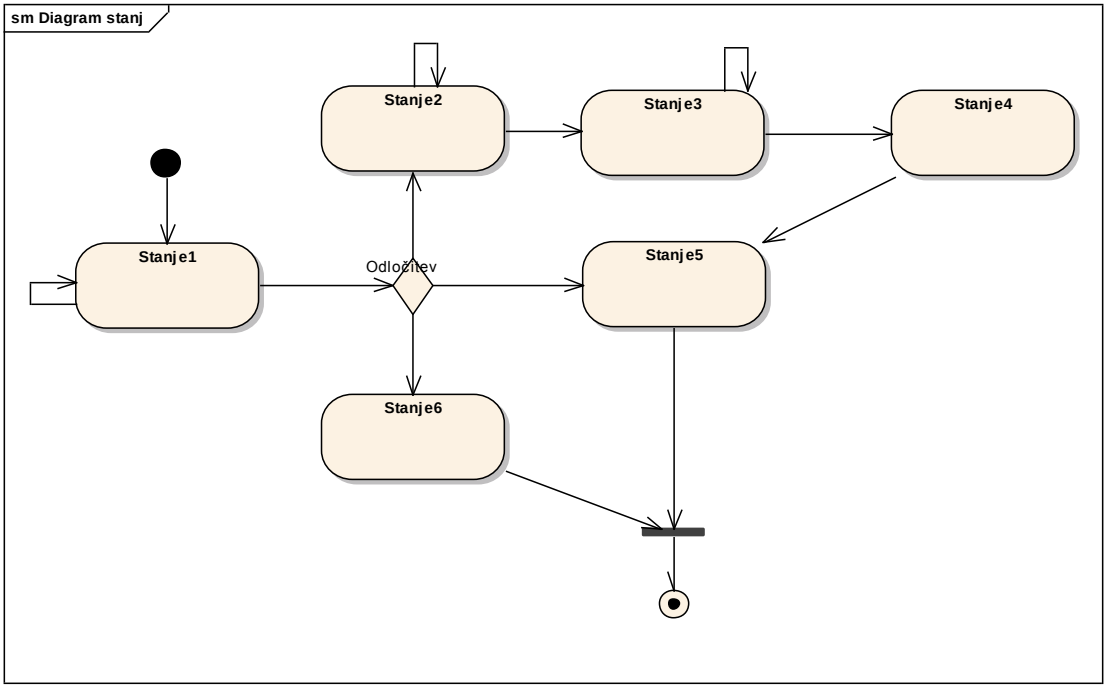
Slika 15: Primer zaporednega diagrama



Slika 16: Primer diagrama sodelovanja



Slika 17: Primer diagrama stanj



Korak zaključimo, ko imajo vsi objekti preproste metode in jih lahko prepustimo za fazo implementacije. Za vse metode, ki jih v arhitekturni fazi nismo opredelili kot sinhrone ali asinhrone, to storimo sedaj.

Pri odločitvi izhajamo iz diagramov sodelovanja objektov, ki smo jih izdelali kot alternativne rešitve. Odločitev mora biti skladna z navodili arhitekturne faze in načrtovanjem vseh ostalih komponent, ki uporabljajo metodo. Posodobimo tudi dokumentacijo arhitekturne faze. Diagrame sodelovanja objektov pregledamo in ocenimo možnost hkratnega dostopa (branje, pisanje) do objektov, kar zahteva popravke pri ustreznih metodah. Hkratnost se lahko pojavi tudi pri objektih, ki nastopajo v več kot enem diagramu sodelovanja ali kot posledica večnitnosti znotraj diagrama sodelovanja. Na koncu preverimo ali so vsi razredi, ki smo jih prenesli iz faze analize še zmeraj potrebni (lahko so redundantni).

6.4.5 Načrtovanje baze sistema

Razredne diagrame lahko rišemo na tri načine: kot konceptualni, specifikacijski ali implementacijski diagram. Kateri način bomo izbrali, je odvisen od perspektive, ki jo želimo opisati.

Pri pripravi logične arhitekture pa konceptualni nivo ne zadošča več. Na vrsti je priprava razrednega diagrama na specifikacijskem nivoju. Najtežja naloga tega koraka je iskanje razredov. Pri tem se nam najbolj obrestuje perspektiva, ki jo poskušamo ohranjati in pa spretnost ter izkušnje, kajti čarobnega recepta, po katerem bi razrede našli, ni. Tehnika, ki jo uporabljamo za iskanje razredov, vključuje pa analizo besedil, intervjuje z uporabniki in domenskimi strokovnjaki, se imenuje Analiza samostalnikov in glagolov. Analiza samostalnikov in glagolov je zelo preprost način, kako z analizo besedil, ki smo si jih pridobili z intervjuji z naročniki, poskušamo najti razrede, atribute in odgovornosti. Osnovno vodilo analize se glasi: samostalniki in samostalniške fraze nakazujejo na razrede ali atribute, medtem ko glagoli in glagolske fraze kažejo na odgovornosti in operacije razredov. Pozorni moramo biti na to, da so besedila, ki jih analiziramo, zapisana v jeziku problemske domene, jezik pa moramo poznati dovolj dobro, da lahko prepoznavamo tudi sinonime in homonime. Poleg opisov primerov uporabe, lahko ugotovimo da so delovni dokumenti bogat vir razredov. Stvari kot so računi, naročilnice in različne druge knjigovodske listine nam pomagajo odkrivati razrede. Pri tem pa moramo biti pazljivi, ker se lahko zgodi, da rano ti delovni dokumenti predstavljajo tisto podvojeno delo, ki bi ga radi z našim sistemom zamenjali.

Omeniti velja še konceptualne entitete, na katerih lahko sloni poslovanje. Primer takšne konceptualne entitete je na primer program zvestobe, ki ga ravno tako vključimo v razredni diagram. Verjetno je najtežji del pri tej analizi iskanje “skritih” razredov. V vsaki domeni obstajajo neki koncepti, ki jih domena smatra za samoumevne. Zato se nam lahko zgodi, da jih mogoče iz samega besedila ne bomo razbrali. V splošnem se zavemo, da smo našli tak skriti razred, takrat, ko naenkrat postane celoten model smiseln. Če se nam torej zdi, da model nima pravega pomena, potem se oziramo za skritimi razredi. To nam je lahko le v pomoč, če ne drugače, pa z dodatnimi raziskavami in razgovori problemsko domeno bolje razumemo.

Pravila, ki se jih poskušamo držati pri kreiranju razredov, so naslednja (Arlow et al., 2002, str. 133 – 134):

Razred naj ima tri do pet odgovornosti. Razred naj bo čim bolj enostaven in omenjeno število odgovornosti nam zagotavlja tak razred.

- Razred ni nikoli samostojen. Razred je vedno povezan z drugimi razredi, saj objekti v sistemu sodelujejo med seboj in s svojim delovanjem vplivajo eden na drugega. Zato lahko pričakujemo, da je vsak razred povezan z manjši številom drugih razredov.
- Bodimo pozorni na večje število zelo majhnih razredov. Kadar ima naš diagram veliko majhnih razredov z eno ali dvema odgovornostma, je dobro diagram proučiti še enkrat in razmisliti o morebitnem združevanju manjših razredov v večje.
- Ravno tako bodimo pozorni na manjše število razredov, ki pa so zelo veliki. V tem primeru imamo ravno obratno stanje kot smo ga opisali v prejšnji točki. Imamo nekaj razredov z večjim številom odgovornosti. To število presega številko pet. V tem primeru naj bo naša strategija naslednja – oglejmo si te razrede in razmislimo o njihovem razdruževanju na manjše razrede, ki bodo nosili ravno pravšnje število odgovornosti.
- Bodimo pozorni na »funkcioide«. »Funkcioid« je v resnici čisto običajna procedura, zakrinkana v razred. Analitike, ki so navajeni uporabljati tehniko funkcionalne dekompozicije s pristopom od zgoraj navzdol, so pa pred nalogo, da prvič pripravijo razredni diagram, rado zanese v tvorbo »funkcioidov«.
- Potem so tu še »vsemogočni« razredi. To so razredi, za katere izgleda, da počnejo vse. Mogoče pa so to razredi, ki prestavljajo sistem? Strategija v tem primeru je

naslednja: poglejmo odgovornosti teh razredov in poiščimo kohezivne, torej dobro povezane podskupine odgovornosti. Če nam to uspe, potem lahko razdružimo razred na manjše, med seboj povezane razrede, ki z medsebojnim sodelovanjem implementirajo obnašanje originalnega, »vsemogočnega« razreda.

- Izogibajmo se globokim dednim hierarhijam. Pri načrtovanju dobre dedne hierarhije je pomembno to, da ima vsak nivo hierarhije dobro definiran namen. Dodajanje nivojev v hierarhijo je pač zelo enostavno in lahko ugotovimo, da v resnici nimajo nekega uporabnega namena. V resnici se dogaja, da načrtovalci uporabljajo dedovanje, da bi z njim popisali funkcionalno dekompozicijo, kjer ima vsak nivo abstrakcije le eno odgovornost. Uporabljajmo torej dednost le v primeru, ko je to jasno razvidno iz problemske domene.

Razredni diagram na specifikacijskem nivoju je rezultat analize zahtev. Poleg analize zahtev in priprave razrednega diagrama, pa pripravimo še definicijo arhitekture in načrt komponent programske opreme. Tokrat upoštevamo tudi nefunkcionalne zahteve in omejitve. Naš načrt arhitekture opišemo z namestitvenim diagramom, medtem ko za opis komponent uporabimo komponentni diagram. Oba diagrama lahko združimo v en diagram, s katerega lahko razberemo tako fizična vozlišča in fizične povezave med vozlišči kot tudi komponente, ki jih pričakujemo na vsakem od vozlišč ter odnose med komponentami.

Tako kot pri ugotavljanju zahtev, velja tudi v tej fazi, da elemente modeliranja, ki jih želimo podrobneje opazovati, opišemo z dodatnimi diagrami. Tako uporabimo na primer diagram prehajanja stanj pri opisovanju življenjskega cikla objekta. Diagram aktivnosti uporabimo na primer pri razlagi neke kompleksne metode nekega razreda. Kadar želimo podrobno opisati sodelovanje med več objekti, uporabimo tehniko risanja diagramov sodelovanja. Ves čas pa se držimo načela, da ne rišemo diagramov za čisto vsako malenkost. Osredotočamo se raje na ključna področja. Dosedanje izkušnje kažejo, da je bolje delati z nekaj diagrami, ki jih redno vzdržujemo kot z množico pozabljeni in zastarelih diagramov.

6.4.6 Kritične točke sistema

Identificiramo kritične točke sistema. Osnovne informacije dobimo iz arhitekturnih prototipov, ki natančno opisujejo izbrano arhitekturo sistema in predstavljajo temelj analize zmogljivosti.

V skladu z ugotovljenimi kritičnimi točkami pripravimo načrt analize zmogljivosti. Izdelamo kriterije sprejemljivosti.

6.4.7 Razvoj kode

Implementiramo razrede, ki so pogojeni z izbrano tehnološko osnovo.

Razvijemo potrebno kodo tehnoloških razredov. Pri tem upoštevamo pravila programiranja (npr. dogovore o poimenovanju, strukturiranju, dokumentiranju).

Opis razredov iz načrtovanja algoritmov predstavlja osnovo za pripravo kode komponente, ki implementira razred v izbranem programskem jeziku. V primerih, ko izbran programski jezik ali njegove knjižnice že vsebujejo primeren razred ga lahko uporabimo (razmislimo o zmogljivosti in vzdrževanju). Vsak atribut postane spremenljivka tega razreda. Enako velja za povezave, ki omogočajo navigacijo med objekti (implementacija odvisna od izbire podatkovne baze). Vsaka operacija iz faze načrtovanja logične arhitekture postane metoda ciljnega razreda. Vsako metodo implementiramo na podlagi specifikacij za operacije.

6.4.8 Dopolnjevanje projektnega slovarja

Projektni slovar sproti dopolnjujemo s pojmi, katerih definicije želimo zabeležiti.

6.4.9 Mejnik in dokumentacija iz faze izdelave

Ob zaključku faze izdelave se sodelujoči na projektu sporazumejo o naslednjih točkah:

- stabilnost sistema (stabilna in zrela programska oprema ter dokumentacija o sistemu),
- pripravljenost sodelujočih na naslednjo fazo (sodelujoči na projektu ter bodoči poslovni uporabniki so pripravljeni na namestitev sistema),
- dejavniki tveganja (sodelujoči so seznanjeni z dejavniki tveganja v naslednji fazi in se strinjajo s strategijami za upravljanje z ugotovljenimi tveganji),
- stroški (sodelujoči se strinjajo z dosedanjimi stroški in oceno stroškov za naslednjo fazo),
- rokovnik za fazo prehoda (sprejem rokovnika),
- ujemanje s standardi podjetja (soglasje o ustreznosti izdelka).

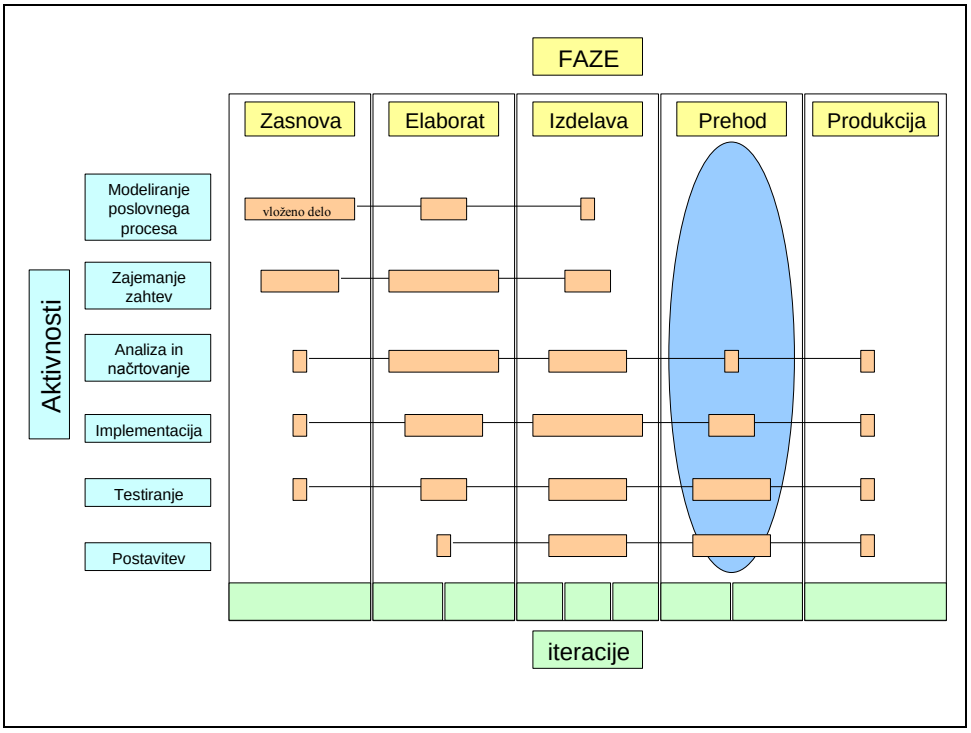
V fazi izdelave zabeležimo naslednje dokumente:

- ➔ zapisniki sestankov projektne skupine;
- ➔ model primerov uporabe;
- ➔ razredni diagram, ki predstavlja konceptualni pogled na informacijski sistem in se ga poslužujemo pri projektih, kjer domene ne poznamo dovolj dobro;
- ➔ razredni diagram na specifikacijskem nivoju;
- ➔ logična arhitektura sistema;
- ➔ fizična arhitektura sistema;
- ➔ projektni slovar.

6.5 Prehod

V fazi prehoda se osredotočimo na prenos razvitega sistema v produkcijo. V tem času izvajamo testiranja strojne in programske opreme ter se mogoče odločimo za izdajo beta verzije izdelka. Informacijski sistem uglašujemo in odpravljamo morebitne pomanjkljivosti.

Slika 18: Faza prehoda



6.5.1 Testiranje sistema

Zmogljivost je pogosto ena izmed zahtev in meril kakovosti sistema. Z zmogljivostjo se moramo ukvarjati že v času analize in načrtovanja, ne pa šele ob zaključku implementacije. Razumljivo je, da lahko zmogljivost natančno določimo šele, ko je celoten sistem izdelan. V primerih, ko se sistem pokaže kot premalo zmogljiv, moramo poiskati razloge za odstopanja in izvesti optimizacijo delovanja. Optimizacije ne izvajamo nad metodami, ki se redko uporabljajo, temveč nad tistimi, ki jih sistem pogosto proži. Za optimizacijo kode obstajajo tudi posebna orodja, ki beležijo klice posameznih metod in čas izvajanja.

6.5.2 Testiranje uporabniških vmesnikov

Pripravimo si načrt testiranja, na podlagi katerega bomo testiranje nato izvedli. Za testne primere določimo cilje, ki jih želimo doseči. Določimo jih na podlagi pogovorov s sodelujočimi na projektu (projektni vodja, naročnik, končni uporabnik) in primeri uporabe. Pričakovanja naj izrazijo v obliki kakovosti in produktivnosti, ki jih poskušamo preoblikovati v cilje testiranja. Cilji testiranja morajo biti realni in merljivi, zato jih izražamo v številčno preverljivi obliki. V načrtu testiranja moramo določiti minimalni nivo sprejemljivosti znotraj nekega časovnega okvirja. Če je možno, vpletemo v določevanje ciljev tudi končnega uporabnika. Kadar to ni možno, naj izdelane kriterije vsaj pregleda in odobri. Oblikujemo načrt testiranja in določimo postopke testiranja.

Določimo cilje testiranja. Pri tem se lahko osredotočimo na kateregakoli izmed parametrov kakovosti. Na podlagi ciljev testiranja določimo načrt testiranja ter testne postopke.

Testni primeri

Na podlagi projektnega načrta in uvrstitve primerov uporabe v različne iteracije izberemo primere uporabe, katerih funkcionalnost želimo testirati. Kot osnovo za pripravo testnih primerov (TP) uporabimo opis primerov uporabe iz začetne faze (strukturiran opis, poslovno

naravnan) in realizacijo primera uporabe s pomočjo sodelovanja množice razredov (diagrami sodelovanja, diagrami zaporedja). Testne primere pripravimo v skladu s postavljenimi kriteriji, nefunkcionalnimi zahtevami in scenariji primerov uporabe. Pri oblikovanju testnih primerov upoštevamo tudi uporabljeno arhitekturo. Ker sistem gradimo v več iteracijah, je potrebno izvajati integracijsko (med iteracijo) in sistemsko (na koncu vsake iteracije) testiranje.

Na podlagi izbranih scenarijev primerov uporabe določimo testne vzorce. Pri tem je potrebno zajeti vse razrede podatkovnih vrednosti (pravilne, mejne, nepravilne).

Vzpostavitev testnega okolja

Z vzpostavitvijo ustreznega testnega okolja zagotovimo ponovljivost. Definiramo konfiguracijo sistema, ki je potrebna za izvajanje posameznega testa.

Izvajanje testiranja

Teste delimo na več vrst. V vsakem primeru pa moramo testiranje izvajati v skladu s postavljenimi testnimi postopki in zahtevi po dokumentiranju izvajanja testov. V primeru sistemskega testiranja gledamo na sistem kot na celoto vključno z dokumentacijo in drugimi nastalimi proizvodi. Izvedemo niz testov, ki preveri skladnost funkcionalnosti sistema s funkcionalnimi in nefunkcionalnimi zahtevami. Naloga tega koraka je ugotoviti ali sistem kot celota deluje v skladu z postavljenimi zahtevami. V primeru prevzemnega testiranja je poudarek na izpolnjevanju prevzemnih kriterijev.

Teste izvedemo v vnaprej določenem testnem okolju. Izvedemo jih v skladu s testnimi postopki. Avtomatizirani testi olajšajo izvajanje testov. Rezultate, dobljene pri testiranju, je potrebno natančno dokumentirati.

6.5.3 Opredelitev korekcij sistema in uporabniških vmesnikov

Na podlagi dobljenih testnih rezultatov ocenimo skladnost z predvidenimi rezultati. Vsako odstopanje dokumentiramo v testni dokumentaciji, identificiramo razlog odstopanja in pripravimo korekcijske ukrepe, ki bodo privedli do odstranitve razloga za odstopanje. Korekcijski ukrepi lahko vplivajo na spremembo zaporedja aktivnosti in vlog, ki jih posamezniki imajo v projektu.

Za vsako izvajanje testnega primera je potrebno ugotoviti skladnost dobljenih in pričakovanih rezultatov.

Za vsako odpoved je potrebno določiti razlog in pripraviti korekcijski ukrep. Določiti je potrebno odgovornosti in časovne termine za odpravo napake.

6.5.4 Tehnična dokumentacija

Tehnično dokumentacijo sistema ustvarjamo v bistvu že ves čas razvoja informacijskega sistema, saj sem spadajo:

- model primerov uporabe,
- modeli, ki opisujejo arhitekturo sistema,
- model podatkovne baze,
- modeli, ki opisujejo algoritme,
- navodila za namestitev sistema.

6.5.5 Namestitev sistema

Ko je zaključeno zunanje testiranje in odpravljene vse zaznane pomanjkljivosti, pripravimo namestitvene pakete, ki morajo vsebovati tudi navodila za namestitev.

Glede na arhitekturo sistema ločimo namestitve za odjemalce, aplikacijske strežnike ter podatkovno bazo.

Namestitveni paketi za odjemalce morajo biti enostavni, saj je spekter potencialnih uporabnikov namestitvenega paketa zelo širok. Najprimernejši so t.i. čarovniki za namestitev.

Namestitveni paketi za aplikacijski strežnik morajo vsebovati detajlne specifikacije namestitve, saj je delovanje aplikacijskega strežnika ključnega pomena za učinkovitost celotnega sistema.

Namestitveni paketi za podatkovni strežnik vsebujejo navodila in skripte za postavitev baze podatkov ter potrebnih objektov.

Pri namestitvah v okviru projekta eProstor je potrebno smiselno upoštevati Generične in Posebne tehnološke zahteve in druge specifične dokumente za posamezna področja.

Navodila za namestitev

Navodila za namestitev morajo biti napisana razumljivo, posamezni koraki namestitve morajo biti v navodilih za namestitev dovolj detajlni in natančno razloženi.

Navodila za namestitev aplikacijskega strežnika morajo vsebovati detajlne specifikacije namestitve, saj je delovanje aplikacijskega strežnika ključnega pomena za učinkovitost celotnega sistema. V navodilih je potrebno tudi specificirati, katera predhodna znanja mora imeti izvajalec namestitve.

V navodilih za namestitev podatkovnega strežnika morajo biti določene možne različice baz podatkov, ki se lahko uporabljajo, ter njihove verzije.

Načrt prehoda na nov sistem

V primeru, ko naročnik že ima nek sistem, ki pokriva vsaj del funkcionalnosti novega sistema, je potrebno narediti načrt prehoda iz starega na nov sistem.
Glede na značilnosti starega in novega sistema, okolja, kjer delujemo, ter zahtev naročnika se odločimo, kakšen tip prehoda bomo uporabili: direktni, vzporedni, fazni ali pilotski.

- direktni:
  Star sistem se opusti in takoj začne uporabljati nov sistem. Ta način je najhitrejši, vendar najbolj tvegan. Uporabimo ga le, ko smo prepričani, da je tak način smiseln in bo uspešen.
- vzporedni:
  V času uvajanja uporabljamo star in nov sistem. Je najvarnejši način, vendar tudi najbolj naporen zaradi dvojnega dela. Ta način uporabimo, ko nismo prepričani o primernosti ali zanesljivosti razvitega sistema.
- fazni:
  Postopoma aktiviramo posamezne module novega sistema. Pogoj pa je, da sta si star in nov sistem podobna. Je varnejši kot direktni prehod, ker tvegamo le pri enem modulu, ampak dlje traja.
- pilotski:
  Manjša skupina ljudi začne uporabljati nov sistem. Po nekem času pokažemo rezultate še ostalim in jih lažje prepričamo v uporabo novega sistema.

6.5.6 Uvajanje uporabnikov

Uvajanje uporabnikov poteka v 5-ih korakih:

1. Začetno uvajanje uporabnikov, ki bodo program testirali: to usposabljanje se izvaja po notranjem testiranju in po izdelavi prvih navodil za uporabo.
2. Splošno usposabljanje uporabnikov: izvaja se po zaključeni implementaciji končne izdaje produkta in izdelavi celotnih uporabniških navodil. Usposablja se vse potencialne uporabnike. Predstavi se jim koncepte delovanja in celotno sliko sistema.
3. Usposabljanje uporabnikov po področjih: po skupinah se usposablja uporabnike določenih modulov.
4. Usposabljanje specifičnih uporabnikov: usposablja se uporabnike, ki imajo specifične naloge, npr. skrbnike šifrantov, uporabnike, ki opravljajo izključno namenska opravila, itd.
5. Dodatna izobraževanja zaradi novih izdaj: pri večjih spremembah ali dodanih novih modulih je potrebno dodatno usposabljanje uporabnikov.

Uvajanje poteka na testnem sistemu na naročnikovi ali razvojni lokaciji.

Če je sistem, ki ga razvijamo, dovolj enostaven, lahko po potrebi združimo ali celo opustimo katerega od naštetih korakov.

Navodila za uporabo

Navodila za uporabo moramo izdelati pred prvim usposabljanjem uporabnikov.

Jasno moramo zapisati, za katero izdajo uporabniškega vmesnika smo napisali navodila, saj moramo s spremembami uporabniškega vmesnika tudi navodila dopolnjevati in spreminjati. Navodila za uporabo moramo pripraviti tako, da bodo enostavna in razumljiva. Operacije razložimo po korakih ter zelo detajlno in natančno. V navodilih se poslužujemo zaslonskih slik, saj je s tem uporaba in razumevanje navodil na bistveno višji ravni.

6.5.7 Dokumentacija za naročnika

Dokumentacija, ki jo izročimo naročniku ob zaključku faze izdelave, zajema:

- navodila za namestitev,
- uporabniška navodila,
- navodila za sistemske skrbnike,
- modele, ki opisujejo arhitekturo sistema,
- logične modele podatkovne baze in izvedbene skripte.

6.5.8 Zaključek in dokumentacija iz faze izdelave

Osnovni cilj faze prehoda je prenos razvitega sistema v produkcijo. Prehod pa sodelujoči na projektu potrdijo skozi naslednje točke:

- potrditev sistema (sistem potrdijo sodelujoči na projektu, ki predstavljajo poslovne uporabnike),
- operativna potrditev (sistem in dokumentacijo, ki se nanaša na sistem, potrdijo tisti, ki bodo zadolženi za operativno vzdrževanje sistema v produkciji),
- potrditev podpore sistema (sistem in dokumentacijo, ki se nanaša na sistem, potrdijo tisti, ki bodo zadolženi za podporo sistema v produkciji(v primeru, ko naročnik v celoti prevzema sistem)),
- stroški (pregled in potrditev tekočih stroškov ter ocene stroškov prihodnje faze).

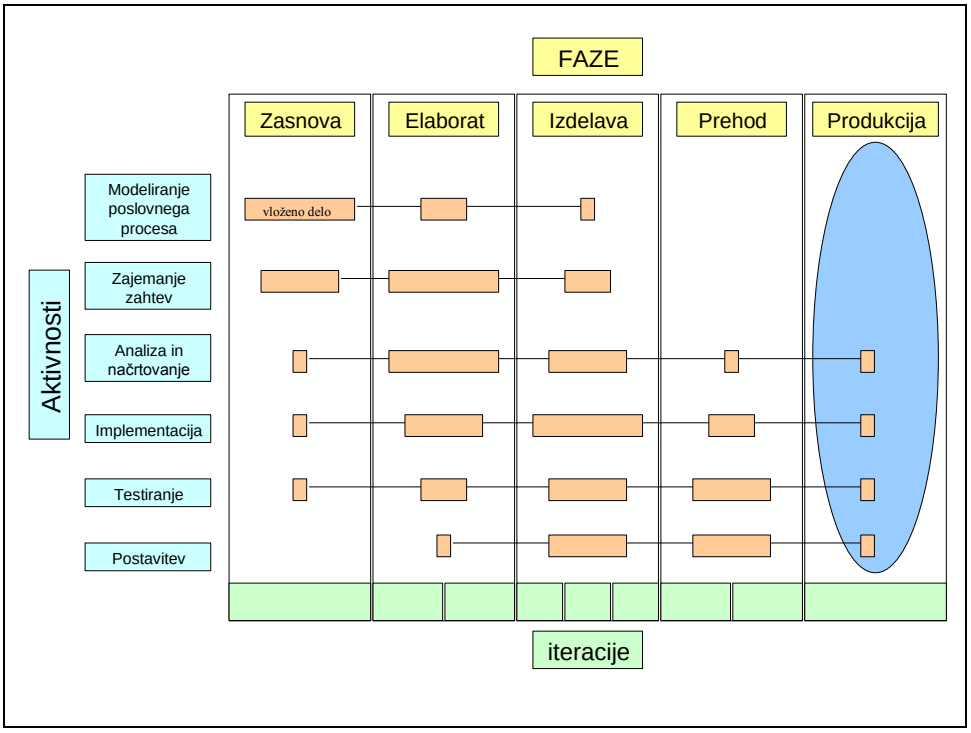
Iz faze izdelave pridobimo naslednje dokumente:

- zapisniki sestankov projektne skupine;
- namestitveni paket;
- poročilo o analizi zmogljivosti;
- poročilo o testiranju sistema in uporabniškega vmesnika;
- seznam zahtevanih popravkov na sistemu in uporabniškem vmesniku;
- navodila za namestitev sistema;
- navodila za uporabo.

6.6 Produkcija

V fazi produkcije skrbimo za operativno delovanje sistema, sprotna vzdrževalna dela na sistemu in upravljamo z zahtevami po spremembah sistema.

Slika 19: Faza produkcije



6.6.1 Vzdrževanje

V operativnem smislu skrbimo za to, da je sistem na razpolago končnim uporabnikom. Med vzdrževalna dela lahko uštejemo popravila manjših napak, ki se pojavijo med delovanjem sistema ter redna vzdrževalna dela na sistemu.

6.6.2 Upravljanje z zahtevami po spremembah sistema

Spremembe sistema lahko zahteva naročnik ali izvajalec. Glede na obseg sprememb, ki jih izvedemo na sistemu, izdamo lahko novo verzijo sistema ali pa izvedeno nadgradnjo obstoječe verzije. V vsakem primeru damo naročniku na voljo vzporedno razvojno okolje, kjer omogočimo uporabnikom testiranje spremenjenega sistema.

Označevanje verzij oziroma nadgradenj

Označevanje novih izdaj oziroma nadgradenj mora biti v skladu z dogovorjenimi pravili, ki so opisana v nadaljevanju, sicer lahko pride do nepotrebnih zapletov pri komunikaciji z naročnikom ali uporabniki ali pa pri samem razvoju.

Sistemi, ki se razvijajo, so vedno kompleksnejši in takšen je tudi njihov razvoj, ki poteka od zasnove do končne implementacije skozi več faz. Za lažje obvladovanje razvoja izvajamo verzioniranje izdelka, ki ga dopolnimo še s statusom izdelka. Tako dobi vsak izdelek (oziroma vsaka izdaja izdelka) svoje razpoznavno ime, ki je hkrati lahko tudi njegovo komercialno ime. Izdelek ima lahko enega od treh statusov:

Razvojni
Tu je vključeno celotno definiranje izdelka. Sistem je v fazi razvoja, testira ga le razvijalec sam ali ožja ekipa razvijalcev. Zagotovi se izdelek, ki odgovarja razvojni specifikaciji,

izpolnjuje vse tehnične pogoje ter zagotavlja funkcionalnost, za katero je namenjen. Izdelek vsebuje programsko opremo in pripadajočo dokumentacijo (tehnična dokumentacija in navodila za uporabo).

Testni
Razvoj je zaključen. Sledi notranje in zunanje testiranje. Testira se pravilnost delovanja sistema ter skladnost z dokumentacijo. Napake, ki se odkrijejo, se sporočijo razvijalcu. Produkt ostane v tem statusu, dokler se ne odpravijo vse zaznane napake in je notranje in zunanje testiranje uspešno.

Končni
Razvoj in testiranje sta zaključena. Uporabnik dobi končno izdajo produkta.

Verzije označujemo na naslednji način: **x.xx.xxx [R ali T]**

Prva številka pomeni glavno verzijo produkta. Spreminja se pri večjih spremembah oziroma pomembnejših novih funkcionalnostih sistema. Druga številka je podverzija produkta. Številka se veča pri spremembah in novih funkcionalnostih sistema. Korak, s katerim se podverzija poveča, določi vodja projekta glede na velikost in pomembnost spremembe funkcionalnosti. Vsako podverzijo spremlja dokument, kjer je zapisan seznam sprememb glede na prejšnjo podverzijo. Zadnja številka se poveča za 1 z vsako novo izdajo produkta, ki je poslana v notranje ali zunanje testiranje in ne vsebuje vsebinskih sprememb glede na prejšnjo izdajo, ampak le popravke odkritih napak. Vsaka taka izdaja vsebuje tudi dokument, kjer je seznam napak testiranja z označitvijo njihovega reševanja. To označevanje je sicer v nasprotju z Borlandovimi priporočili, da tretja številka označuje število napak, ki so bile odpravljene za trenutno verzijo in podverzijo. Razvojna izdaja ima zadnjo številko vedno 000. Znak R pomeni razvojno izdajo, T pa testno. Končna izdaja nima dodanega znaka.

Primer imena izdaje produkta: PRODUKT 1.12.027
Zgornje ime označuje končno izdajo produkta PRODUKT za verzijo 1 in podverzijo 12. Do te izdaje smo prišli po 27-ih odpravljanjih napak in testnih verzijah.

Pri spremljanju razvoja produkta preko vseh faz in izdaj je zelo pomembno tudi shranjevanje izvorne kode. Shranjevati je potrebno vse izdaje, ki so trenutno v uporabi pri naročniku, vse končne izdaje ter trenutno razvojno in testno izdajo. Po potrebi vodja projekta shranjuje tudi predhodne razvojne in testne izdaje. Datoteke shranjujemo v drevesni strukturi oblike:
\ime produkta\izdaja\datoteke izvorne kode

V času razvoja se uporablja sistem za verzioniranje in revidiranje programske kode (CVS – Concurrent Versions System), kjer se shranjuje celotna izvorna koda izdelka z vsemi spremembami. Dnevno se izdelujejo varnostne kopije, kar zagotavlja le izgubo dnevnih sprememb izvorne kode pri morebitni odpovedi celotnega sistema arhiviranja. To izgubo pa preprečujemo s hranitvijo začasnih razvojnih kopij na računalnikih razvijalcev. Varnostne kopije se hranijo ciklično s periodo enega tedna, kar v praksi pomeni, da se hranijo vse spremembe, ki so nastale v enem tednu.

6.6.3 Mejniki in dokumentacija iz faze produkcije

V fazi produkcije so mejniki tiste točke, ko se odgovorni za sistem na naročnikov strani odločajo za zahteve po spremembah v sistemu, ki zahtevajo novo verzijo sistema. V takšnih primerih se izvajalec in naročnik dogovorita o:

- umiku oziroma zamenjavi verzije sistema (sistema ne bodo uporabljali več, oziroma v primeru zamenjave, gre za dogovor o rokovniku izvedbe zamenjave).

Zelo pomembno je ob vsaki spremembi oziroma novi izdaji osvežiti celotno dokumentacijo, ki mora biti jasno označena, za katero izdajo velja.

Literatura in viri

1. Arlow Jim et al.: UML and the Unified Process. London: Addison Wesley, 2002. 395 str.
2. Fowler M. et al.: UML Distilled Second Edition: A Brieff Guide to the Standard Object Modeling Language. Addison-Wesley. Boston [etc.], 2001. 186 str.
3. Krisper Marjan et al.: Enotna metodologija razvoja informacijskih sistemov: Strateško planiranje. Ljubljana, Vlada Republike Slovenije, Center Vlade RS za informatiko., 2002, 421 str.
4. Krisper Marjan et al.:Enotna metodologija razvoja informacijskih sistemov: Objektni razvoj IS. Ljubljana, Vlada Republike Slovenije, Center Vlade RS za informatiko., 2000, 198 str.
5. Shelly G. Bet al.: Systems analysis and design. Cambridge [etc.], Course Technology, cop., 1998, 485 str.