UNIVERZA V LJUBLJANI

EKONOMSKA FAKULTETA

MAGISTRSKO DELO

TESTIRANJE INFORMACIJSKEGA SISTEMA V RAZLIČNIH RAZVOJNIH FAZAH

Ljubljana, oktober 2005 **Marija Goltez Poljšak**

IZJAVA

Študentka Marija Goltez Poljšak izjavljam, da sem avtorica tega magistrskega dela, ki sem ga napisala pod mentorstvom izr. prof. dr. Andreja Kovačiča in skladno s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

V Ljubljani, 19. 10. 2005 Podpis:_____

KAZALO VSEBINE

1 UVOD

Načrtovanje in razvoj informacijskih sistemov je načeloma dolgotrajen, dinamičen in drag postopek. Kljub izkušnjam in natančnosti sodelujočih pri razvoju informacijskih sistemov pa tudi tu pogosto prihaja do napak. Ljudje smo zmotljivi in pri delu delamo napake, vsaka napaka pa je draga. Toda če napako odkrijemo v zgodnjih fazah razvoja informacijskega sistema, je njeno odpravljanje cenejše.

Tradicionalni pristop v razvojnem ciklu informacijskega sistema postavlja testiranje kot eno izmed samostojnih faz razvojnega procesa, ki jo začnemo izvajati po integraciji in izvedbi sistema ter preden začnemo z vzdrževanjem. Ker pa kar 64% vseh napak nastane prav v začetnih fazah (faza analize in načrtovanja) razvoja informacijskega sistema (Perry, 2000), novejši pristopi že opozarjajo na to, da je testiranje aktivnost, ki jo izvajamo ves življenjski cikel, znotraj in na koncu vsake faze.

Postopek testiranja izvajamo z namenom, da se zagotovi pravilnost razvoja informacijskega sistema in skladnost z zahtevami uporabnikov, ki so bile podane na začetku. Tako lahko prihranimo čas in denar, saj se s testiranjem ob zaključku vsake faze prepričamo, da gradimo pravi proizvod in da ga gradimo pravilno.

Izvedba testiranja je navadno zamudna in utrujajoča. Odkrivanje in odpravljanje napak se v vsaki fazi izvaja v več ciklih: napako odkrijemo in odpravimo, ponovno testiramo in odkrivamo napake ter jih posredujemo v odpravljanje in postopek spet ponovimo. Izvajamo ga toliko časa, da v njem ni več kritičnih napak, ki smo jih odkrili. Zato pa moramo čas, ki ga imamo na razpolago za testiranje, učinkovito izkoristiti. Prav tu pa pogosto prihaja do nasprotij:

- po eni strani si izvajalci testiranja v pričakovanju novega izdelka in v želji, da bi se projekt izgradnje sistema zaključil čimprej, s čim manjšimi stroški, ne vzamejo dovolj časa za testiranje, kljub temu da je testiranje pomemben integralen del razvoja vsakega sistema,
- po drugi strani pa lahko prihaja tudi do tega, da testiranja ne znamo pravočasno zaključiti in stroški testiranja, z njimi pa tudi stroški izgradnje celotnega sistema, naraščajo.

V prvem primeru tvegamo, da bomo na koncu dobili izdelek, ki ne pokriva ali samo delno pokriva potrebe in želje uporabnikov, v končni fazi tudi tvegamo, da bo do napak prihajalo tudi po uvedbi sistema. V drugem primeru pa nepotrebno izrabljamo dragocene vire za testiranje. Nekega splošnega pravila, kako dolgo in koliko testirati, ne moremo postaviti, saj se postopki testiranja od enega do drugega sistema razlikujejo, najti pa moramo neko optimalno količino.

Prav zaradi tega pa je pomembno, da že v prvi fazi, ko načrtujemo razvoj novega informacijskega sistema, predvidimo dovolj časa za testiranje in preverjanje, ali sistem dosega na začetku zastavljene cilje. Testiranje mora biti v razvoju informacijskih sistemov pomembna dejavnost, ki je tesno povezana z vsem procesom razvoja od samega začetka

do konca. Prisotno mora biti pri samem definiranju ciljev, pri izvajanju planov, preverjanju rezultatov in ukrepanju pri nepravilnih rezultatih. Naloga testiranja je predvsem to, da pokaže, ali produkt ustreza zahtevam stranke, hkrati pa odkrije čimveč nepredvidljivih napak.

Najbolj očitno se pri razvoju informacijskih sistemov napake pokažejo pri testiranju že izdelanega programa. Iz prakse vemo, da imajo vsi programi več ali manj napak, ki se odkrijejo z uporabo izdelane programske opreme. Redki programerji so optimisti, ki upajo, da bodo kar takoj napisali program brez napak. Čeprav je programer še tako izkušen in natančen, obstaja verjetnost, da napravi napako. Pisanje programov je namreč tako zahtevno, da se napake v njih hitro pojavijo. Možnost, da se napakam izognemo, je majhna, zato je testiranje programov ena izmed pomembnejših faz v razvoju programske opreme.

Pogosto se zgodi, da se k testiranju sistema pristopi neplanirano in slabo organizirano. Za kakovostno testiranje je namreč treba pripraviti natančen plan testiranja, določiti prave izvajalce testiranja, izbrati prave tehnike testiranja, imeti urejen in natančen postopek dokumentiranja in sporočanja napak, ponovnega testiranja in merjenja uspešnosti odkrivanja napak, znati pa je treba tudi prenehati s testiranjem v pravem času oz. ugotoviti, do katere mere je testiranje še smiselno.

Razvijalci informacijskih sistemov so v nenehni stiski s časom in denarjem, projekti so praviloma dragi in izvedba traja kar precej časa. Nenehno testiranje v vseh fazah razvoja pa je ena izmed možnosti, kako prihraniti nekaj časa in denarja, saj odkrita napaka pomeni ponovno izvedbo vseh korakov v življenjskem ciklu razvoja sistema. Kasneje v življenjskem ciklu, ko odkrijemo napako, več korakov moramo ponovno izvesti. V povezavi s testiranjem se tako razvijalci vedno znova soočajo z novimi izzivi, ki so z naraščanjem kompleksnosti informacijskih sistemov vse večji.

Zaradi zgoraj navedenih razlogov pa mora biti ena osnovnih nalog vsakega vodje projekta v povezavi s testiranjem ta, da že v samem začetku razvoja formira skupino izvajalcev testiranja, poskrbi za njihovo izobraževanje in po možnosti pomaga pridobiti ustrezna orodja za testiranje. Seveda pa mora biti po izvedbi preverjanja in testiranja tudi pripravljen prisluhniti odkritim napakam, predlogom in pripombam, ki mu jih posredujejo. Če so izvajalci testiranja sami sebi namen, faza testiranja pa je pač samo še ena izmed faz v razvoju informacijskega sistema, potem se lahko izkaže, da je bila s tem storjena ena večjih napak pri razvoju informacijskega sistema (Whittaker, 2000).

Uvedba postopkov testiranja v vse faze razvoja informacijskega sistema nedvomno prinaša veliko koristi. V začetnih fazah razvoja gre tu predvsem za neprestano preverjanje, ali pravi proizvod gradimo pravilno, to je v skladu z zahtevami, ki so bile podane na začetku. V interakciji z bodočimi uporabniki sistema tudi njim damo možnost za ponovno preverjanje in potrditev zahtev. V zadnjih fazah razvoja pa gre za zagotovitev zmanjšanja možnosti prekinitev in nedelovanja sistema v produkcijskem okolju. Skupni imenovalec koristi pri uvedbi postopkov testiranja v vse faze razvoja pa je izgradnja kakovostnega

proizvoda, saj se s tem, ko napake sproti odpravljamo, skrajša tudi čas izgradnje sistema, znižujejo pa se tudi stroški razvoja. Verjetno je želja prav vsakega projektnega vodje, da uporabnikom lahko preda v uporabo kakovosten izdelek, ki je bil narejen s čim nižjimi stroški v čim krajšem času.

1.1 Namen in cilji

Osnovni namen magistrskega dela je prikazati pomembnost vloge testiranja v vseh fazah razvoja informacijskih sistemov in dokazati, da brez natančnega načrtovanja ni dobre izvedbe testiranja. Seveda je težko postaviti stroga pravila, kako testiranje pravilno izvesti, saj se informacijski sistemi med seboj močno razlikujejo. Eden od namenov naloge je tudi dokazati, da se pri izvedbi testiranja ni treba natančno držati vseh navodil, ki so zapisana v literaturi, saj se testiranje lahko izvaja na različne načine, čeprav za nobenega od njih ne moremo trditi, da je najboljši ali najslabši. Merilo pri tem mora biti zadovoljstvo uporabnikov in stabilen, delujoč sistem.

Ob začetku izdelave magistrske naloge sem si postavila naslednje cilje, ki izhajajo iz zgoraj zapisanega namena:

- razložiti nekaj osnovnih pojmov o informacijskih sistemih in njihovem razvoju,
- poiskati definicije testiranja in pojasniti, kaj je testiranje in kaj so napake ter kako se na testiranje sploh pripraviti,
- na osnovi teoretičnih spoznanj in praktičnih izkušenj prikazati razloge za uvedbo testiranja v vse faze razvoja informacijskega sistema,
- prikazati praktično izvedbo testiranja na primeru in
- opraviti analizo opisanega primera ter podati predloge za nadaljnje delo.

1.2 Metoda dela in struktura poglavij

Metoda magistrskega dela bo temeljila na:

- analitično-teoretičnem pregledu slovenske in tuje literature s področja razvoja in testiranja informacijskih sistemov,
- predstavitvi ter analizi izvedbe testiranja na primeru razvoja informacijskega sistema za bančno zavarovalništvo.

Kot dodatne vire bom uporabila tudi tiste, do katerih je možno dostopati preko globalnega omrežja. Glavne ugotovitve in izkušnje avtorjev iz prvega dela magistrske naloge bom skušala povzeti in dodati tudi nekaj svojih pogledov na obravnavano temo.

Magistrsko delo ima poleg uvoda in zaključka še sedem poglavij. V uvodu je predstavljena problematika obravnavanega področja, namen in cilji ter metoda dela. V skladu z zastavljenimi cilji bom v drugem poglavju prikazala teoretične osnove o razvoju informacijskih sistemov. Navedla bom nekaj značilnosti informacijskih sistemov, opredelila, kaj je življenjski cikel razvoja in skušala na kratko razložiti, kaj je treba v

posamezni fazi življenjskega cikla razvoja informacijskega sistema narediti. Predstavila bom tudi nekaj različnih modelov življenjskih ciklov razvoja informacijskih sistemov.

Tretje poglavje bo podalo nekaj splošnih informacij v povezavi s testiranjem. Skušala bom podrobneje opredeliti, kaj sploh je testiranje, kakšne vrste testiranja poznamo, kakšen je proces testiranja, kako se pripravi plan testiranja in izberejo testni primeri. Posebno poglavje namenjam tudi klasičnim napakam, ki se pojavljajo pri testiranju, nekaj besed pa bo zapisanih tudi o izbiri članov testne skupine.

Četrto poglavje je namenjeno metodam in tehnikam testiranja ter avtomatizaciji izvedbe testiranja.

V nadaljevanju bom na osnovi znanja, pridobljenega iz prebrane literature, prikazala, kako naj bi se izvajalo testiranje v posamezni razvojni fazi. Nekaj besed bom namenila tudi stroškom testiranja.

Sedmo poglavje je namenjeno predstavitvi izvedbe testiranja na praktičnem primeru. Na začetku namenjam nekaj strani opisu problematike, nadaljujem pa z natančnim opisom testiranja v posamezni razvojni fazi.

Osmo poglavje je namenjeno analizi opisanega primera. Njegovo dejansko izvedbo bom primerjala s znanjem, ki sem ga med pripravo magistrske naloge pridobila iz prebrane strokovne literature. Prikazala bom tako pozitivne strani naše izvedbe, kot tudi opozorila na njene pomanjkljivosti in priložnosti za izboljšanje.

Magistrsko delo se konča z zaključkom.

2 INFORMACIJSKI SISTEMI

Informatika je dejavnost oblikovanja, uvajanja in izvajanja informacijskih sistemov v organizaciji. Zajema ugotavljanje potreb po podatkih in informacijah, njihovo organiziranost, informacijsko tehnologijo in izdelovanje računalniških rešitev. Informatika pa je tudi teoretična disciplina o sestavi, oblikovanju, delovanju in vzdrževanju informacijskih sistemov (Gričar, 2002, str. 6).

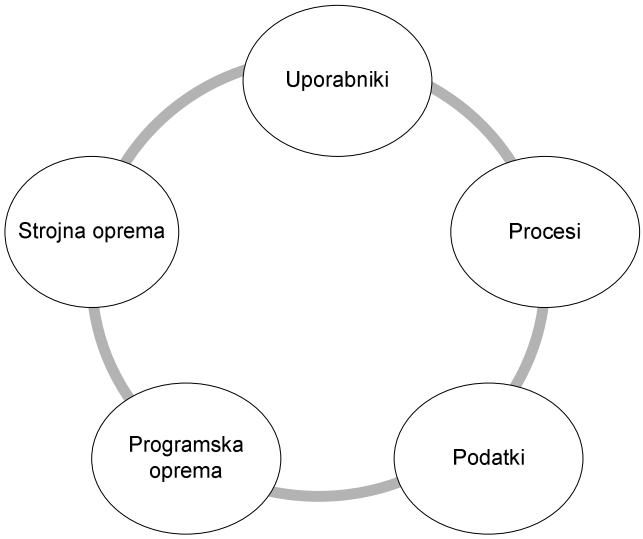
Slika 1 prikazuje pet komponent, ki tvorijo informacijski sistem (Shelly et al., 1998, str. 6):

- strojna oprema,
- programska oprema,
- podatki,
- procesi in
- ljudje oziroma uporabniki sistema.

Informacijski sistem pa lahko označimo tudi kot kombinacijo v podatkovni bazi shranjenih podatkov, človeških sposobnosti in tehničnih pripomočkov, ki skupaj z

ustreznim nizom organizacijskih postopkov proizvajajo informacije za podporo upravljanju, poslovanju in odločanju. Informacijski sistem je model realnega sveta.

Slika 1: Komponente informacijskega sistema



Vir: Shelly et al., 1998

Poglejmo, kakšne definicije informacijskega sistema še najdemo v literaturi. Informacijski sistem je jedro podjetja – če je »živ« in če daje prave informacije pravim ljudem ob pravem času na pravi način (Hajšek, 1999, str. 551).

Tudi Alter definira informacijski sistem kot sistem, ki uporablja informacijsko tehnologijo za hranjenje, posredovanje, obdelavo in prikazovanje informacij, ki se pojavijo v enem poslovnem procesu (Alter, 1996, str. 2).

Avison opredeljuje informacijski sistem kot sistem, ki zbira, hrani, obdeluje in posreduje informacije, potrebne za delovanje podjetja. Informacijski sistem mora biti zgrajen na tak način, da so informacije v njem uporabne in dostopne vsem, ki jih želijo uporabljati. To so zaposleni, kupci, deležniki podjetja in tudi javnost (Avison, 1995, str. 13).

Gričar definira informacijski sistem kot celoto sestavin, ki zagotavlja podatke in informacije ter povezave med temi sestavinami v podjetju in z okoljem podjetja. Sestavine informacijskega sistema so ljudje, podatki in informacijska tehnologija (Možina et al., 1994, str. 711).

Informacijski sistem je tudi zbirka metod, pripomočkov in dejavnosti, ki jih neka organizacija potrebuje za zadovoljevanje potreb po informacijah.

Na splošno lahko informacijske sisteme po značilnosti razdelimo na takšne, pri katerih je osnovni poudarek na:

- zbiranju in hranjenju določenih podatkov,
- obdelavi podatkov in ustvarjanju končnih informacij.

Vsi obstoječi sistemi pa so bolj ali manj kombinacija obeh zgoraj navedenih možnosti.

Poslovni informacijski sistemi so sistemi, ki pokrivajo določeno poslovno področje v podjetju. Razvijalci sistema morajo v ta namen natančno preučiti delovanje podjetja: njihove izdelke, storitve in seveda zahteve, saj se le-te od podjetja do podjetja zelo razlikujejo. V današnjem hitro razvijajočem, dinamičnem okolju morajo biti podjetja sposobna hitrega prilagajanja. Za to potrebujejo informacijski sistem, ki se bo sposoben prilagajati vedno novim zahtevam dinamičnega okolja.

2.1 Značilnosti informacijskih sistemov

Pri razvoju se morajo razvijalci natančno seznaniti s tem kako bo bodoči informacijski sistem deloval in kako bo podpiral poslovne funkcije. Pri definiranju značilnosti morajo razmisliti o tem:

- Ali je sistem kakorkoli povezan s katerimkoli drugim sistemom? Pri tem mislimo na notranje (v podjetju) in zunanje (zunaj podjetja) sisteme. V komunikaciji z zunanjimi podjetji gre za elektronsko izmenjavo podatkov.
- Kje so meje sistema? Meje sistema kažejo, kje se en sistem konča in drugi začne. Navadno jih je težko določiti, saj meje med enim in drugim sistemom niso povsem jasne.
- Kako bo sistem obravnaval posebne poslovne potrebe? V tem primeru gre za posebne poslovne potrebe, ki so značilne za posamezno podjetje, npr. na fakulteti - razpored predavanj, ocenjevanje študentov; v banki – izplačilo denarnih sredstev.
- Kakšna bo velikost podjetja oz. razvoj podjetja v prihodnosti? Mala in velika podjetja v isti dejavnosti imajo običajno povsem različne informacijske sisteme. Za primer lahko vzamemo banko z eno ali dvema poslovalnicama v primerjavi z banko, ki ima poslovalnice po vsej državi in tudi v tujini. S številom komitentov, računov in transakcij se veča tudi kompleksnost informacijskega sistema.

2.2 Razvoj informacijskih sistemov

Informatika igra vedno bolj pomembno vlogo, zato je treba njeno načrtovanje povezati s strateškimi razvojnimi cilji organizacije. Načrtovanje informatike pomeni načrtovanje celotne ali globalne informacijske infrastrukture v obravnavani organizaciji, kar je hkrati tudi predpogoj za uspešen nadaljnji razvoj informacijskih sistemov posameznih poslovnih funkcij (Kovačič, 1994, str. 52). Potreba po novem informacijskem sistemu ali izboljšavi obstoječega sistema izhaja iz problemskega okolja in spremenjenih ali povečanih uporabniških zahtev. Razvoj informacijskega sistema je dolgoročen investicijski poseg, pri katerem je treba upoštevati značilnosti in lastnosti okolja, uporabniške potrebe in zahteve, rezultate analize stroškov, investicijsko politiko,…

Izhodiščna točka za začetek razvoja novega oz. spremembo obstoječega informacijskega sistema je priprava sistemske zahteve oz. dokumenta, ki predstavlja tudi uradno zahtevo po izgradnji novega informacijskega sistema. Glavni razlogi, ki vplivajo na pripravo sistemske zahteve oz. na sprožitev pobude po izgradnji novega informacijskega sistema, pa so (Shelly et al., 1998):

- zmanjšanje stroškov,
- izražena potreba po povečani količini informacij,
- izboljšane kontrole,
- izboljšanje ponudbe in
- izboljšanje storitev.

Zahtev po izgradnji novega sistema je navadno precej več, kot bi jih bili razvijalci sposobni realizirati. Prav zato se v podjetju pogosto določi skupina, ki je zadolžena za pregled zahtev, postavitev prioritet in pripravo ekonomske in tehnične študije upravičenosti izvedbe projekta, zelo pomembna pa je tudi kritična ocena o tem, ali bo nov sistem uporabnikom res prinesel določene prednosti in koristi.

Za razvoj, izvedbo in delovanje informacijskega sistema moramo izvesti zaporedje dejavnosti, ki jih imenujemo življenjski cikel razvoja sistema. Ta zajema tehnološki in postopkovni načrt za razvoj in izvedbo. Postopki vzdrževanja, razširitve in ažuriranja sistema pa so podsistem dejavnosti v celotnem razvojnem ciklu sistema.

2.3 Življenjski cikel razvoja informacijskih sistemov

Kot večina razvojnih procesov tudi razvoj informacijskega sistema sledi določenemu življenjskemu ciklu oz. razvojnemu modelu, ki določa zaporedje faz razvoja, ki jih je treba metodološko izvesti. Življenjski cikel je organizirano in vnaprej določeno ogrodje, znotraj katerega razvijamo programske sisteme oz. preoblikujemo zahteve v delujoč računalniški program.

V slovarju je življenjski cikel razvoja (angl. development life cycle) opredeljen kot opis načrtnega in postopnega razvijanja informacijskega sistema po fazah (Slovensko društvo informatika, 2004, str.16)

Po standardu ANSI/IEEE Std 792-1983 lahko življenjski cikel programske opreme opredelimo kot nabor diskretnih aktivnosti v času razvoja programske opreme in programskih sistemov. Čas izvajanja posameznih aktivnosti, hkrati pa tudi same aktivnosti, imenujemo faze življenjskega cikla. Vsaka od naštetih faz mora dati določene rezultate, katerih kakovost se stalno preverja. Različni avtorji posamezne faze različno imenujejo, vendar v glavnem lahko rečemo, da razvojno življenjski cikel sestavljajo naslednje faze:

- strateško načrtovanje,
- sistemska analiza,
- sistemsko načrtovanje,
- izgradnja sistema ter
- delovanje in vzdrževanje sistema.

2.3.1 Strateško načrtovanje

Osnovni cilj strateškega načrtovanja je zbiranje vseh potrebnih podatkov in njihovo preučevanje na uvodni ravni. Strateško načrtovanje mora podati temelje prihodnjega informacijskega sistema. Rezultat te faze pa mora biti osnovni načrt za poslovni in tehnološki razvoj novega ali dopolnitev obstoječega informacijskega sistema, v katerem mora biti opredeljeno, kaj je dejansko treba narediti ter opredeliti časovni in stroškovni obseg. Strateško modeliranje mora zagotoviti definicijo poslovnih procesov in organizacijske strukture, seznam poslovnih funkcij sistema za določitev njegove operativnosti, pregled vseh sistemskih odgovornosti, seznam potrebnih podatkov za zadovoljevanje večine sedanjih ali predvidenih informacijskih zahtev. Strateško načrtovanje lahko razdelimo na tri faze:

- strateška raziskava, ki pomeni zbiranje zahtev ter ocenjevanje možnosti, njen rezultat pa je ustrezna študija izvedljivosti,
- podrobna strateška analiza, ki predstavlja uvodno sistemsko analizo in modeliranje,
- strateško načrtovanje, ki poda rešitve za arhitekturo informacijskega sistema.

Ključni rezultati faze strateškega načrtovanja so: opredelitev danosti, poslovnih usmeritev in ciljev sistema, analiza primernosti ter analiza stroškov in koristi, opredelitve meja in obsega novega sistema, model arhitekture, uvodna ocena velikosti in vrednosti sistema, fazni razvojni plan in določitve sistemske odgovornosti.

2.3.2 Sistemska analiza

Sistemska analiza mora podati povsem jasne ugotovitve o tem, kaj mora informacijski sistem ponuditi, ne sme pa se spuščati v opis, kako to doseči in izvesti. Zato morajo biti ugotovitve iz strateškega načrta preverjene in prečiščene. V fazi analize popišemo obstoječe stanje in prikažemo, kakšne bodo izboljšave po uvedbi novega informacijskega sistema. Vse zahteve moramo zbrati in pripraviti plan za izboljšavo obstoječega stanja. Proces sistemske analize lahko razdelimo v dve pomembni veji:

- podatkovna analiza, ki jo sestavljajo dejavnosti za določitev uporabniških zahtev po informacijah. Določijo se potrebni podatki za zadovoljevanje uporabniških informacijskih potreb, njihova opredelitev, atributi, relacije in organizacija,
- procesna analiza, ki jo sestavlja podobna razčlenitev z namenom preučevanja sistemskih funkcij in procesov.

Rezultat opravljene analize naj bi bilo popolno razumevanje problemskega področja. Bistveno vprašanje, na katerega skušamo odgovoriti v fazi analize, je, KAJ potrebujemo oz. kaj naj bi grajeni sistem zagotavljal. Faza analize zahtev mora biti opravljena zelo natančno, saj tu nastajajo najhujše napake, ki se vlečejo naprej v naslednje faze.

2.3.3 Sistemsko načrtovanje

Osnovni cilj sistemskega načrtovanja je izdelava učinkovitega načrta za obnovo obstoječega ali izgradnjo novega sistema. V fazi načrtovanja določimo, KAKO bomo identificirane zahteve zadovoljili. Pripravimo vse potrebno za razvoj informacijskega sistema: definiramo vse vhode, izhode, datoteke, izpise, določimo, kako bodo delovali programi, določimo vse kontrole,… Načrtovalski model ali načrt sistema mora biti natančen in strog, saj predstavlja osnovo za izvedbo. Prehod iz faze sistemske analize v fazo sistemskega načrtovanja je osrednja kritična točka v razvoju vsakega informacijskega sistema, saj se v tem delu pogosto naredi največ sistemskih poenostavitev, napak in približkov. Zelo pomembno je, da v tej fazi dobimo potrditev bodočih uporabnikov programov, da tak sistem, kot smo jim ga predstavili, res služi namenu in ga bodo uporabniki lahko s pridom uporabljali. Ob zaključku te faze moramo imeti narejeno podrobno sistemsko analizo arhitekture, podroben načrt razvoja sistema, natančen načrt logične in fizične podatkovne sheme, slike uporabniških vmesnikov, znane morajo biti strukture datotek ter določena strategija zagona in prehod na nov informacijski sistem.

2.3.4 Izgradnja sistema

Cilj faze izgradnje sistema je pretvorba izdelanih načrtov v operativni sistem. V tej fazi zgradimo informacijski sistem: napišejo se programi, izvedejo se ustrezna testiranja, pripravi se dokumentacija in opravi namestitev sistema. Namen te faze je, pripraviti uporabnikom izdelek, ki bo skupaj z ustrezno dokumentacijo primeren za takojšnjo uporabo. Ta faza vključuje tudi pripravo novega okolja, izobraževanje uporabnikov, namestitev strojne in programske opreme, postavitev baze podatkov, migracijo podatkov iz starega v novo okolje. Faza izgradnje sistema se mora zaključiti z oceno zgrajenega sistema, kjer se ugotavlja, ali so bil nameni, definirani na začetku, doseženi in so bili stroški razvoja v pričakovanih okvirih.

2.3.5 Delovanje in vzdrževanje sistema

Po končani izgradnji sistema vedno nastopi še faza delovanja in vzdrževanja sistema. Cilj te faze je zagotoviti tekoče delovanje sistema, zato je potreben nadzor ter stalno opazovanje delovanja in procesnih zmogljivosti informacijskega sistema. Uporabniki lahko kljub testiranju še vedno odkrijejo določene napake, ki jih je treba odpraviti. Ker pa je sistem živ in se stalno spreminja, moramo skrbeti za nenehno dopolnjevanje, izboljševanje, reševanje ter realizacijo dodatnih uporabniških zahtev. Faza vzdrževanja se pravzaprav nikoli ne konča. Dokler sistem deluje oz. ga ne nadomesti nov, bolj primeren sistem, moramo skrbeti za njegovo nemoteno delovanje in prilagajanje spremembam v okolju.

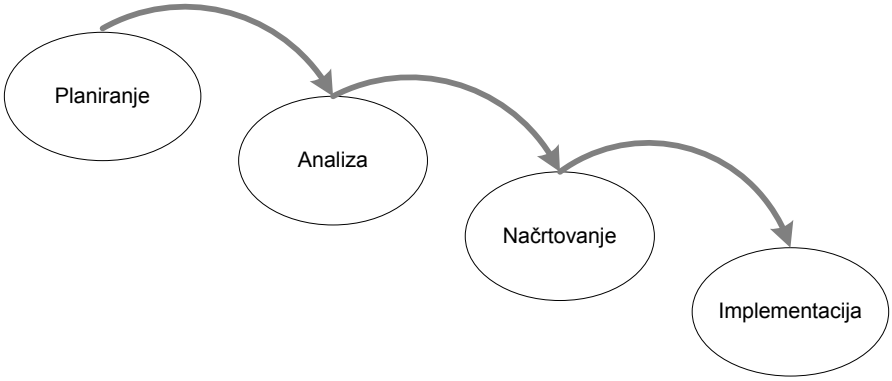
2.4 Različni modeli življenjskih ciklov

Model življenjskega cikla je zaporedje korakov, skozi katerega gre sistem, ki ga razvijamo. Poznamo različne modele razvoja, vsi pa zajemajo fazo planiranja, analize, načrtovanja, implementacije in vzdrževanja. Med seboj se razlikujejo predvsem po

podrobnejši delitvi faz na aktivnosti ter v zaporedju in načinu njihovega izvajanja. V nadaljevanju navajam nekaj najpogosteje uporabljenih modelov:

- **Zaporedni ali kaskadni model**: je najstarejši razvojni model. Zanj je značilno, da se izvede vsaka razvojna faza sistema zaporedno in v celoti, naslednje faze pa ne vplivajo na predhodne. Najprej se izvede faza analize, ki se konča, preden začnemo naslednjo - fazo načrtovanja. Njegova celotna slika spominja na vodne slapove (angl. waterfall model). Slabost tega modela je, da je treba na oprijemljive rezultate čakati dolgo, napake pa se pokažejo šele pri sistemskem testiranju, saj zaporedni model ne dopušča vračanja nazaj. Tveganje, da sistem ne ustreza zahtevam, je visoko vse do zadnje faze razvoja. Opisan razvojni model prikazuje slika 2.

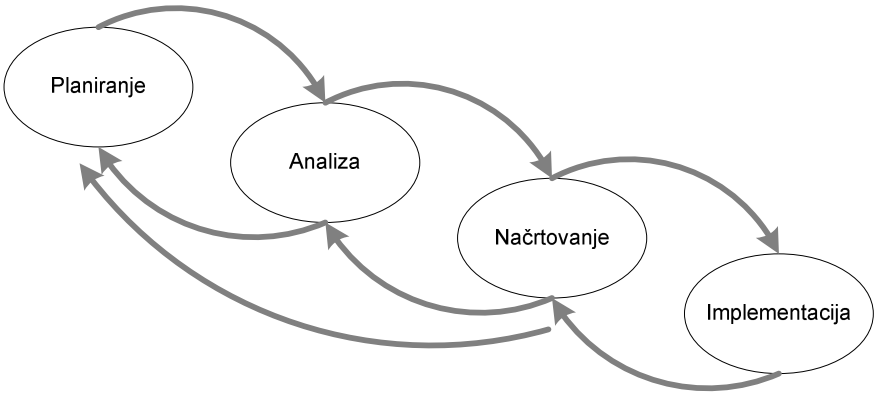
Slika 2: Zaporedni ali kaskadni razvojni model



Vir: Shelly et al., 1998

- **Krožni ali postopni razvojni model** je prikazan na sliki 3: zanj je značilno, da se vse razvojne faze lahko začnejo sočasno in se razvijajo delno ali celo povsem vzporedno. Rezultati, ki jih dobimo v posamezni fazi, lahko pomenijo ponovitev predhodne faze.

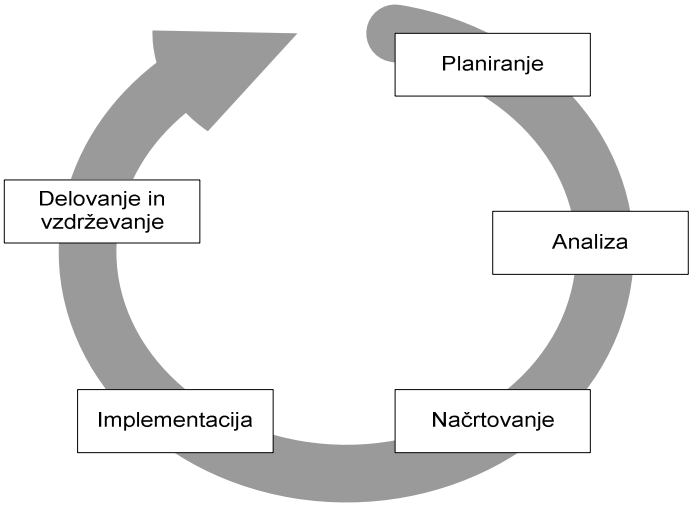
Slika 3: Krožni ali postopni razvojni model



Vir: Shelly et al., 1998

- **Spiralni razvojni model**: posamezne faze se izvedejo le delno in se dokončajo v več ponavljanjih. Postopke ponavljamo, dokler ni dosežen nek končni model informacijskega sistema. Najbolj tvegane so začetne iteracije, zato takrat razvijemo najbolj tvegane dele sistema, vsaka iteracija pa vključuje povezovanje v celoten sistem in preizkušanje. Prednosti tega modela so, da so najbolj tvegani deli sistema razrešeni, še preden postane investicija velika, preizkušanje in povezovanje v sistemu sta nepretrgana, začetne iteracije omogočajo zgodnje povratne informacije s strani uporabnikov, možna je predaja izvedenega dela projekta, še preden je celoten projekt dokončan. Ta model razvoja predstavlja slika 4.

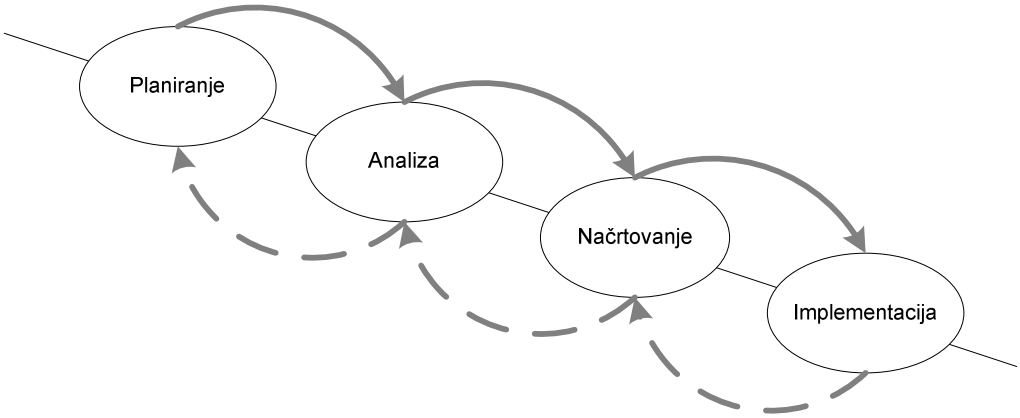
Slika 4: Spiralni razvojni model



Vir: Shelly et al., 1998

- **Kombiniran razvojni model**: je bil zasnovan na osnovi zaporednega modela in omogoča vračanje v predhodne faze. Je blizu naravnemu procesu razvoja, ker nudi osnovno zaporedje in dopušča poljubna prehajanja med fazami. Zaradi naštetih razlogov se tak model, ki je prikazan na sliki 5, v praksi tudi največkrat uporablja.

Slika 5: Kombiniran razvojni model



Vir: Shelly et al., 1998

3 TESTIRANJE

3.1 Kaj je napaka in kakšni so vzroki za nastanek napak?

Napaka se lahko pojavi v katerikoli fazi razvojnega procesa informacijskega sistema: pri planiranju, analizi, oblikovanju sistema, programiranju, v uporabniških navodilih, pri pripravi dokumentacije,… S stališča nastanka napake pa je ključnega pomena prav faza načrtovanja programske opreme.

Če le-ta ne poteka dosledno in v pravilnem zaporedju, je to največji generator napak (Zorko, 2004), kritična pa postane v tistem hipu, ko začne vplivati na pravilnost delovanja sistema.

V grobem lahko trdimo, da je testiranje odpravljanje napak pri razvoju informacijskih sistemov. Pa na začetku poglejmo, kaj sploh je napaka[1] in katere vrste napak poznamo.

Mayers pravi, da je napaka prisotna v vsakem primeru, ko sistem/program ne dela tistega, kar končni uporabniki pričakujejo (Kaner, 1999, str. 60). V tej definiciji je človeški faktor izključen, kljub temu pa je eden pomembnejših faktorjev, ki vplivajo na napake.

V primeru programske opreme o napaki govorimo v naslednjih primerih ko (Zorko, 2004):

- program ne opravlja nečesa, kar navaja specifikacija,
- program izvaja nekaj, kar specifikacija pravi, da ne bi smel,
- program počne nekaj, česar specifikacija ne omenja,

[1] V magistrski nalogi se bom omejila na definicijo napak, ki nastajajo pri razvoju informacijskih sistemov.

- program je težko razumeti, težko ga je uporabljati, je počasen, v očeh uporabnika pa ne bo viden kot ustrezen.

Pri definiciji napake moramo vedno upoštevati specifikacijo izdelka, saj le-ta podrobno opredeli izdelek – kaj bo nastalo, kako bo deloval, kaj bo omogočal,… Napaka je torej vsakršno odstopanje od zahtev, navedenih v specifikacijah, vendar samo v primeru, da te specifikacije v resnici obstajajo in so pravilno napisane. Če je specifikacija nepravilno napisana in ji pri razvoju dosledno sledimo, ne moremo pričakovati korektnih rezultatov ob zaključku razvoja. Ločimo dve vrsti odstopanja delovanja sistema (Perry, 2000, str. 6):

- od zahtev, ki so bile podane v specifikacijah; v tem primeru gre lahko tudi za napačno razumevanje specifikacij,
- od pričakovanj uporabnikov: to pomeni, da so bile nepravilno ali pomanjkljivo napisane že specifikacije.

Lahko pa se tudi zgodi, da zahteva v specifikaciji ni bila zapisana, pa je kljub vsemu vgrajena v sistem. V vsakem primeru gre tudi tu za napako, saj je, glede na zgornjo definicijo, napaka vse kar se razlikuje od zahtev podanih v specifikacijah.

Tudi Zorko (2004) pravi, da je vzrok za napake prav specifikacija, za kar pa obstaja več razlogov:

- za veliko programske opreme specifikacija enostavno ne obstaja,
- specifikacija ni dovolj podrobna in konkretna,
- specifikacija se neprestano spreminja,
- v razvojnem teamu je premalo medsebojnega komuniciranja.

Vsaka napaka pa lahko povzroči veliko škodo organizaciji, v kateri sistem deluje. Nekatere napake so sicer manjše in njihove posledice ne povzročijo velikih težav, druge pa so takšne, da lahko povzročijo tudi milijonsko škodo.

Prav zaradi tega je naloga izvajalcev testiranja v tem, da odkrijejo čimveč napak, da bi tako lahko preprečili težave pri delovanju sistema. Večino napak v sistemu povzroča nepravilen ali napačno izveden proces. Ta na primer lahko povzroči, da končni uporabniki sistema ne bodo dobili ustreznih informacij, ki jih potrebujejo pri vsakdanjem delu.

Napake se pojavljajo v vseh fazah razvoja, torej jih lahko pričakujemo praktično povsod. Moramo pa se zavedati, da do njih ne prihaja zato, ker bi bili razvijalci sistema neodgovorni in neprevidni, pač pa zaradi kompleksnosti sistema, saj imamo ljudje za obvladovanje kompleksnih sistemov omejene možnosti.

3.2 Kaj je testiranje?

Testiranje je aktivnost, ki je tesno povezana z vsem procesom razvoja programske opreme od samega začetka do konca. Prisotno je pri definiranju ciljev, pri izvajanju planov, preverjanju rezultatov in ukrepanju pri nepravilnih rezultatih. Naloga testiranja je

predvsem v tem, da pokaže, da produkt ustreza zahtevam stranke, hkrati pa odkrije čimveč nepredvidljivih napak (Križnik, 2002).

Testiranje je torej proces identifikacije napak oz. vsaka aktivnost z namenom ocenitve, ali sistem ustreza na začetku podanim zahtevam. Pogoste definicije testiranja so naslednje (Drake, 2004):

- testiranje je proces, s katerim skušamo prikazati, da v sistemu, ki ga gradimo, ni napak,
- testiranje je aktivnost ali proces, s katerim lahko prikažemo, da program ali sistem deluje v pričakovanih okvirih,
- testiranje je aktivnost, s katero bomo dokazali in dosegli zaupanje, da sistem dela, kar bi moral delati – v skladu z zahtevami, ki so jih specificirali uporabniki.

Navedene definicije so na prvi pogled pravilne, vendar zavzemajo preveč pozitivno stališče, torej dokazujejo, da sistem deluje. Testiranje programa lahko pokaže na prisotnost napak, ne more pa dokazati, da napak v programu ni (Solina, 1997, str. 176)! Osnovni namen testiranja je torej dokazati prisotnost ene ali več napak. Torej bi bila najprimernejša definicija testiranja naslednja (Myers, 1997, str. 11): »Testiranje je proces izvajanja programa/sistema z namenom odkrivanja napak.«

Glede na zgornjo definicijo lahko trdimo, da je testiranje uspešno samo v primeru, če v sistemu odkrijemo napake. V nasprotnem primeru je bila celotna izvedba dokaj zapletenega postopka prava izguba časa.

Testiranje lahko pojmujemo kot integralni del nadzora kakovosti celotnega razvojnega cikla, pri katerem moramo poskrbeti, da je rezultat vsake razvojne faze kakovosten in v skladu z globalnimi cilji (Solina, 1997, str. 175). V evropskem standardu EN 45020 je testiranje skupaj s certificiranjem in akreditiranjem definirano kot eden od postopkov »ocenjevanja skladnosti«. Gre za opravljanje enega ali več testov, s katerim se po točno določenem tehničnem postopku določi ena ali več karakteristik danega proizvoda, procesa ali storitve – torej ugotavljanje skladnosti med zahtevami in dejanskimi lastnostmi.

Bistvo testiranja je neskončno množico kombinacij prevesti v končno minimalno število najverjetnejših vzorcev, ki bodo nastopili pri uporabniku, in to temeljito preveriti (Zorko, 2004).

Testiranje je torej pomemben del razvoja vsakega informacijskega sistema, ki mora biti prisoten v vseh fazah razvoja. Na splošno velja, da se za testiranje porabi več kot 50% vsega časa, ki je namenjen razvoju informacijskega sistema.

Pravzaprav se testiranje v današnjem času ne razlikuje toliko od tistega, ki so ga izvajali v preteklosti. Tehnike testiranja, ki so jih iznašli pred 20 – 30 leti, se pravzaprav uporabljajo še danes. Testiranje je lahko drago, vendar pa je ne-testiranje lahko še veliko dražje, to velja še toliko bolj, če so v igri človeška življenja. Testiranje je samo po sebi destruktiven proces, vendar pa se moramo zavedati, da so njegovi rezultati vedno zelo konstruktivni.

Pomemben vidik pri testiranju so povratne informacije o testiranju. Brez njih je neprestano izboljševanje testiranja izredno težavno. Način posredovanja povratnih informacij mora določiti vodstvo, ki mora povedati, kakšne tipe povratnih informacij potrebuje za spremljanje testiranja in posledično napredovanje projekta razvoja informacijskega sistema.

Na tem mestu bi rada opozorila še na t.i. kaskadenje napak, do katerega pride, če sistema ne testiramo in napak ne odpravimo. Kaskadenje napak je domino efekt napak, ki v določenem delu sproži naslednjo še nepovezano napako, ta sproži tretjo in tako naprej. Nevarnost kaskadnosti napak je pogosto povezana s spreminjanjem sistema. Sprememba je narejena in testirana v enem sistemu, v katerem se pojavi, ker pa se, kot posledica spremembe, spremenijo tudi pogoji, pa to povzroči napako v drugem sistemu. Primer takšnega kaskadenja napak je npr. sistem za sprejem naročila. Ta je povezan s serijo različnih sistemov. Na videz nepomembna napaka v sistemu za vnos naročil ima lahko zelo hitro kaskadni vpliv na sistem, kar se odraža v hudi motnji sistema za naročanje (Kiger, 1997). Tveganje kaskadnosti pa se povezuje z intenziviranjem integriranosti sistemov.

3.3 Namen testiranja

Zakaj torej izvajati celoten postopek testiranja, če že vnaprej vemo, da vseh napak ne bomo uspeli najti? Osnovni namen testiranja je torej dokazati prisotnost ene ali več napak, ne pa tudi njihovo lokacijo. S testiranjem tudi ne moremo dokazati odsotnost napak, razen za trivialne primere (Dogša, 1993). S testiranjem ocenimo kakovost dela v posameznih fazah razvoja informacijskega sistema in želimo dokazati prisotnost ene ali več napak. Napake v vsakem sistemu lahko povzročijo veliko izgubo, v nekaterih primerih lahko celo katastrofalno (npr. strmoglavljenje letala, eksplozija vesoljske rakete, ustavitev prodaje delnic,…). V današnjem svetu, kjer so računalniki prisotni na vseh področjih, sta življenje ali smrt lahko odvisna od kvalitete in zanesljivosti sistema. Zagotavljanje kakovosti pomeni zagotavljanje, da sistem ustreza minimalnim zahtevam, ki so bile podane na začetku razvoja.

S testiranjem sistema v vseh razvojnih fazah neprestano preverjamo, ali produkt tekoče faze ustreza zahtevam, ki so postavljene v predhodni fazi. Tematiko testiranja je treba pojmovati kot del obsežnega področja, imenovanega verifikacija in validacija. Z verifikacijo ugotavljamo, ali pravilno gradimo proizvod, z validacijo pa ugotavljamo, ali gradimo pravi proizvod (Dogša, 1993).

Poleg zgoraj naštetega pa je tu še poslovni vidik testiranja. Testiranje je neke vrste kontrola, ki jo lahko učinkovito izrabimo tudi za ocenitev in zmanjšanje tveganja. S pomočjo testiranja lahko vodstvo pridobi in koristno uporabi določene informacije. Tu seveda ne gre za konkretne napake, ki nastajajo pri gradnji sistema, temveč za informacije, ki vodstvu pomagajo pri hitrem in pravilnem odzivu na nastale situacije. Pri postopku testiranja lahko ugotovimo, da je nastala zamuda pri izgradnji sistema, glede na to informacijo lahko vodstvo takoj sprejme odločitev o okrepitvi skupine, ki sodeluje pri

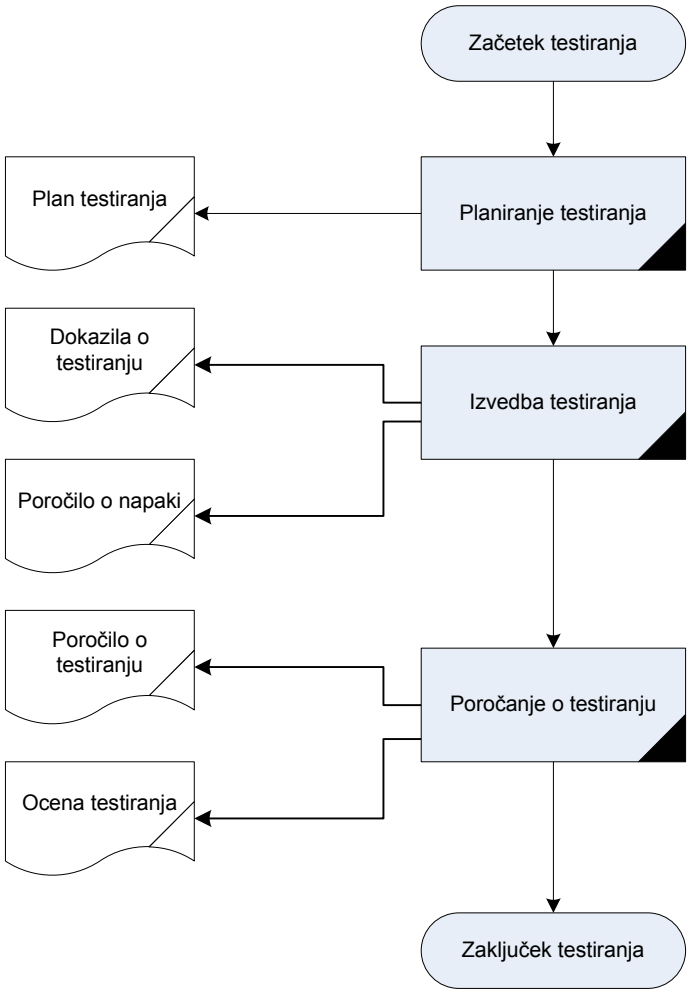
razvoju. Seveda se morajo izvajalci sistema v takšnem primeru zavedati svoje vloge pri testiranju, vedeti morajo, da vodstvo od njih pričakuje informacije, zato jih morajo biti pripravljeni v pravi obliki tudi posredovati.

Namen izvedbe testiranja torej še zdaleč ni v tem, da bi dokazovali, da v sistemu ni nobenih napak več, pač pa s testiranjem skušamo povečati zaupanje v sistem, ki ga gradimo, ne glede na to, v kateri fazi razvoja se nahajamo. Cilj testiranja je zagotoviti, da bo sistem deloval tako, kot je bilo zamišljeno na začetku, z namenom izboljšanja kakovosti, zanesljivosti in vzdržljivosti sistema.

3.4 Proces testiranja

Proces testiranja je kombinacija aktivnosti s področja vodenja in usmerjanja operativnih aktivnosti in aktivnosti upravljanja procesov in nadzora.

Slika 6: Proces testiranja



Vir: Navodilo za testiranje IT rešitve, 2004

V grobem lahko proces testiranja razdelimo na tri osnovne korake (McGregor, 2004), kar prikazuje tudi slika 6:

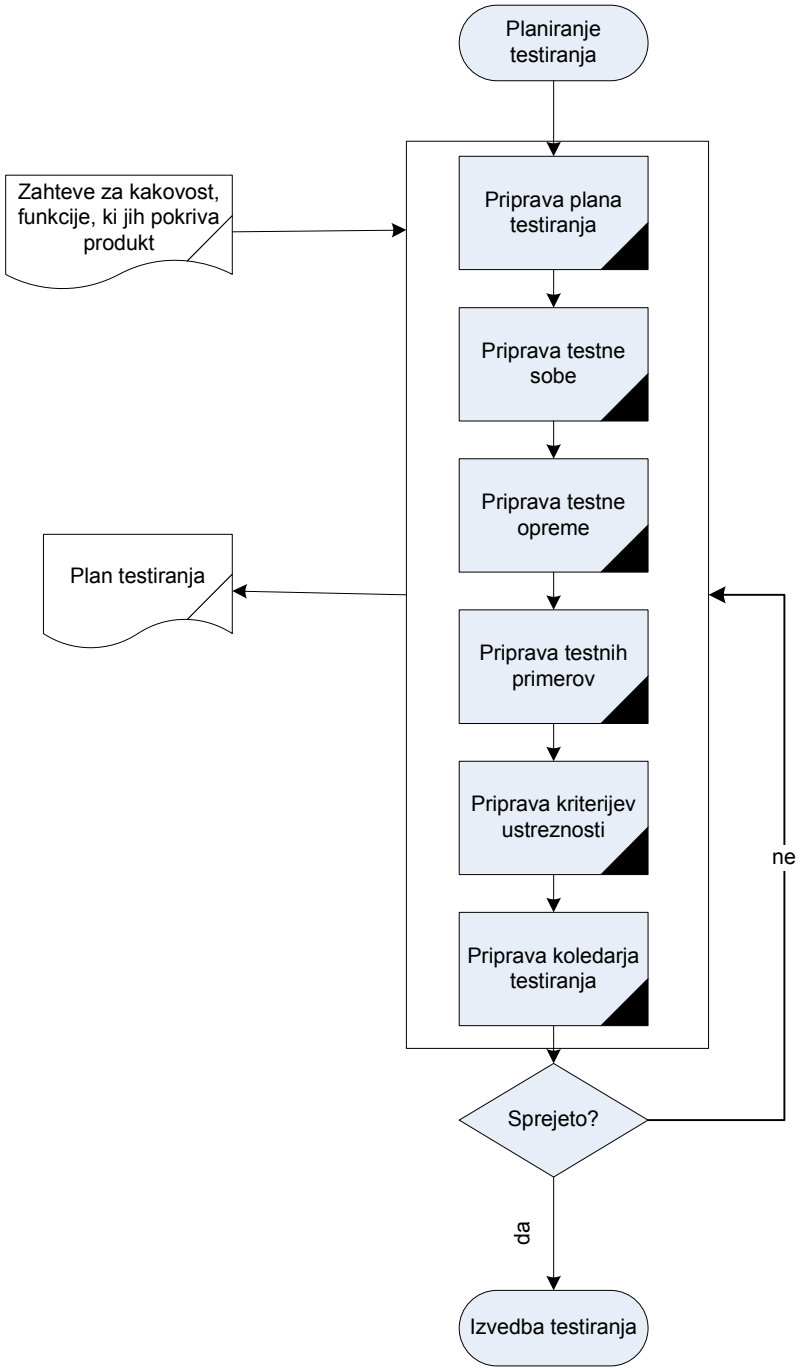
- planiranje testiranja,

- izvedba testiranja,
- priprava poročila o testiranju in njegova ocena.

3.4.1 Planiranje testiranja

V fazi planiranja testiranja moramo odgovoriti na vprašanja: Kaj bomo testirali, kje in s čim, kdaj se bo testiranje izvajalo in kdo ga bo izvajal?

Slika 7: Proces planiranja testiranja



Vir: Navodilo za testiranje IT rešitve, 2004

Določiti moramo kakšni so vhodni podatki, pričakovani rezultati in kako se bo o dobljenih rezultatih testiranja poročalo. Vse odgovore na zgoraj zastavljena vprašanja mora

vsebovati dokument, ki ga imenujemo testni plan. K planiranju testiranja sodi tudi priprava ustrezne testne sobe, testne opreme, orodja za testiranje, priprava ustreznih testnih primerov, določanje kriterijev, po katerih bomo ocenjevali ustreznost sistema in priprava terminskega plana testiranja. Proces planiranja testiranja prikazuje slika 7.

3.4.2 Izvedba testiranja

Po pripravljenem testnem planu sledi faza izvedbe testiranja. Testiranje izvajamo tako, da za izbran testni primer vnašamo vhodne podatke. Ko se testni primer izvede, primerjamo rezultate testiranja s pričakovanimi rezultati. V primeru ustreznega rezultata pripravimo dokazila o izvedbi testiranja in nadaljujemo z novim vnosom podatkov (za isti ali naslednji testni primer). Če pa smo dobili neustrezen rezultat, napako dokumentiramo in jo posredujemo osebi, ki bo poskrbela za njeno odpravo. Ko dobimo obvestilo o odpravi napake, postopek testiranja ponovimo. Proces izvedbe testiranja prikazuje prikazuje slika 8.

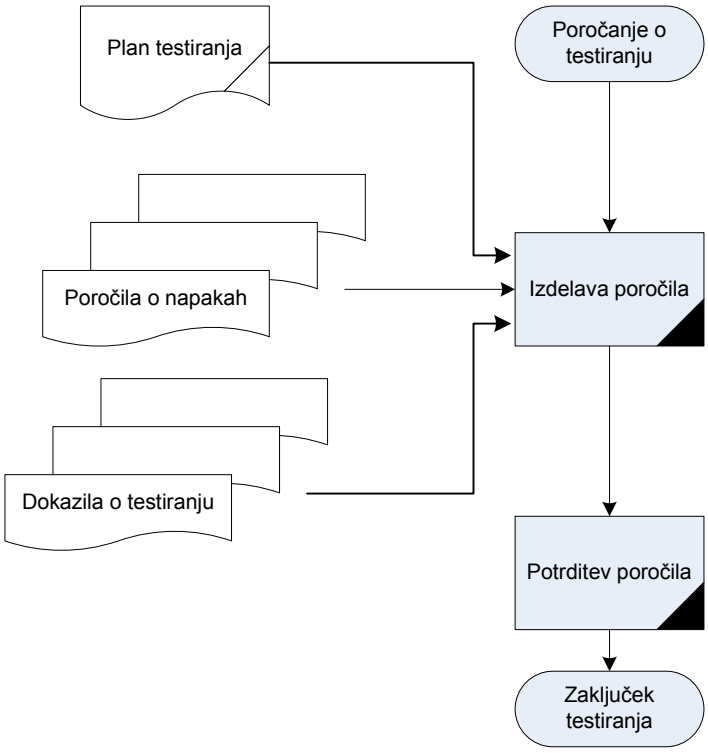
Slika 8: Proces izvedbe testiranja

Vir: Navodilo za testiranje IT rešitve, 2004

3.4.3 Priprava poročila o testiranju in ocena testiranja

Ko zaključimo s testiranjem, na osnovi dokazil o testiranju in poročil o napaki pripravimo poročilo o testiranju, iz katerega je mogoče nedvoumno sklepati, v kolikšni meri produkt ustreza ali ne ustreza zahtevam in pričakovanjem naročnika. Ob zaključku testiranja je smiselno pripraviti tudi oceno izvedbe testiranja, iz katere je razvidno, kako smo zadovoljni z izvedbo testiranja in kje bi se dalo postopek še izboljšati. Grafični prikaz procesa predstavlja slika 9.

Slika 9: Proces poročanja o testiranju



Vir: Navodilo za testiranje IT rešitve, 2004

3.5 Plan testiranja

S testnim planom natančno opredelimo, kako se bo testiranje izvajalo. Pripravljati ga začnemo že v fazi analize uporabniških zahtev. Razvojna skupina skupaj s skupino uporabnikov določi, kakšne bodo zahteve za nov sistem, testna skupina pa te specifikacije uporabi za pripravo plana testiranja sistema. Ta mora zajemati celotno strategijo testiranja sistema, od verifikacij v začetnih fazah razvoja do uporabniškega testiranja aplikacije in testiranja ob predaji sistema v uporabo. Sama priprava testnega plana je precej dolgotrajen postopek, saj po nekaterih navedbah (Perry, 2000, str. 211) zavzema kar tretjino vsega časa, ki ga namenjamo testiranju. Vendar pa nam potrošenega časa ne sme biti žal, saj skrbno pripravljen plan omogoča enostavnejšo izvedbo, analizo in pripravo poročil o testiranju in je torej eden od osnovnih pogojev, ki omogoča učinkovito testiranje.

Testni plan je dokument, ki se stalno spreminja: če se spremenijo zahteve pri razvoju, se mora temu primerno prilagoditi tudi plan testiranja; ves čas mora slediti spremembam in

biti veljaven, saj le tako lahko zagotavljamo, da je testiranje, ki ga izvajamo, ves čas učinkovito.

ANSI/EEEE Standard 829-1983 definira testni plan kot dokument, ki opisuje področje delovanja, namen, vire, urnik in vse testne aktivnosti, ki jih moramo izvesti v postopku testiranja (Kaner, 1999, str. 203).

V grobem lahko rečemo, da moramo v testnem planu navesti (McGregor, 2004):

- **Kaj** bomo testirali: testni plan mora vsebovati jasna navodila, kaj je treba testirati v vsaki posamezni fazi.
- **Kdo** bo testiral: testni plan mora jasno opredeljevati, kakšne so odgovornosti posameznika v različnih fazah razvoja sistema. Več o izbiri testne skupine bom napisala v poglavju o oblikovanju testne skupine.
- **Kdaj** bomo testirali: testni plan mora natančno opredeljevati, kdaj bomo izvajali katero vrsto testiranja.

Testni plan mora biti kratek in enostaven. Lahko je formalen ali neformalen dokument, kar je odvisno od pravil, ki veljajo v določeni organizaciji. Prilagojen mora biti organizacijski strukturi, po drugi strani pa mora biti dovolj fleksibilen, da še dovoljuje kreativnost in določeno prilagajanje v teku projekta. Podrobnejše informacije o tem, kako napišemo testni plan, pa so zapisane v poglavju o pripravi testnega plana.

Poleg zgoraj naštetega pa mora testni plan vsebovati še podrobnejši opis informacijskega sistema, ki ga testiramo, namen testiranja in tveganja, specifična testiranja, ki jih moramo izvesti,… Dobro pripravljen testni plan zagotavlja določen red pri izvajanju postopka, saj je testiranje neke vrste projekt v projektu, ki ga je treba ustrezno upravljati. Upravljanje je toliko zahtevnejše, če je skupina, ki bo izvajala testiranje, velika. Seveda pa to ne pomeni, da se v primeru manjše testne skupine lahko priprava plan testiranja izpusti. Tudi v tem primeru mora biti natančno opredeljena organizacija testiranja. Pripravljen testni plan je namreč za vodje projektov lahko tudi učinkovita pomoč pri upravljanju testiranja.

Priporočljivo je, da plan pregledajo vsi vpleteni v razvoj informacijskega sistema: projektni vodja, programerji, preizkuševalci, kupci oz. vsi, ki so kakorkoli vpleteni v projekt in bi v bodoče lahko imeli določene zahteve glede testiranja. Tako se že pri izdelavi plana rešijo odprta vprašanja glede izvajanja testiranja. Ko ugotovimo, kaj je treba testirati, lahko določimo in ocenimo vse potrebne vire (finančna sredstva, čas, ljudje, oprema), saj testni plan natančno določa, kdo, kako, kje, s kakšnimi viri in zakaj bo izvajal testiranje. Pri identifikaciji nalog navadno ugotovimo, da so nekatere med seboj zelo sorodne. Takrat jih lahko združimo in damo v izvajanje isti osebi oz. manjši skupini.

Proces testiranja zahteva pazljivo razporeditev vseh virov, kar pa moramo upoštevati tudi pri pripravi testnega plana. Vnaprej moramo predvideti, ali bodo izvajalci testiranja za svoje delo potrebovali kakšno posebno programsko ali strojno opremo. Večinoma bodo porabili veliko časa za izvedbo in preverjanje rezultatov testiranja, zato se moramo že pri

pripravi testnega plana zavedati, da (posebno pri kompleksnih sistemih) ne bo možno testirati vseh funkcionalnosti razvijajočega sistema enako natančno.

Primerno pripravljen testni plan je lahko tudi nekakšna pogodba med izvajalci testiranja in projektnim vodjem/uporabniki, ki opredeljuje vlogo testiranja v celotnem projektu razvoja informacijskega sistema. Sama priprava plana pogosto terja veliko časa, vendar se moramo zavedati, da dobro pripravljen testni plan veliko pripomore k lažjemu in hitrejšemu testiranju.

3.6 Priprava testnih primerov

Testni primeri so simulacija poslovnih dogodkov v poslovni praksi. Če bi za testiranje aplikacije imeli neomejeno količino časa, bi teoretično lahko pripravili več milijonov različnih kombinacij testnih primerov. Na žalost pa je čas pri testiranju omejen, z njim pa tudi število testnih primerov na nekaj sto ali tisoč. Izbira testnih primerov je prepuščena preizkuševalcu in dva preizkuševalca tudi ne bosta odkrila istih napak (Zupan, 2004). Prav zato pa je testne primere treba izbrati natančno. Dober testni primer naj bi izpolnjeval naslednje kriterije (Kanner, 1999, str. 124):

- prinaša realne možnosti, da z njim ujamemo napako v programu,
- ni redundanten, npr. dva testna primera iščeta isto napako v programu,
- ni niti preveč enostaven niti preveč kompleksen: veliko časa lahko prihranimo s kombiniranjem več testnih primerov v enega, vendar pa moramo pri tem paziti, da ne postane preveč zahteven, saj je najlažje testirati na enostavnih testnih primerih.

Testne primere zato združujemo v ekvivalentne razrede. V istem razredu se nahajajo testni primeri, ki testirajo npr. isti postopek v programu. Če z določenim testnim primerom odkrijemo napako, jo bodo odkrili tudi ostali primeri v isti skupin, oz. če je ne bo odkril prvi, je verjetno ne bodo odkrili niti ostali primeri v razredu. Pri kreiranju testnih primerov ne smemo pozabiti na napačne vnose v okviru ekvivalentnega razreda, saj je to najpogosteje glavni vzrok za večino napak v programu. Več tipov napačnih vnosov bomo preverili, več napak bomo našli v programu. Kot primer navajamo vnosno polje, kamor naj bi vnesli katerokoli število med 1 in 99. Za ta primer lahko najdemo vsaj štiri ekvivalentne razrede:

- število med 1 in 99,
- število, ki je manjše od 1, kar vključuje 0 in vsa negativna števila,
- števila, večja do 99,
- vsi neštevilčni zanki.

Pri pripravi testnih primerov moramo za vsako polje dobro premisliti, kakšni so ekvivalentni razredi in jih prikazati v primerni obliki. Najlažje to naredimo v obliki tabele. Spodnji primeri prikazujejo, kako izberemo ekvivalentni razred za posamezne primere. Če vnos v polje pripada določeni skupini, mora razred vsebovati člane vse skupine, npr.: vnos države, kjer mora veljaven ekvivalentni razred zajemati vse države, neveljaven razred pa

predstavljajo vsi vnosi, ki niso imena držav. Analizirati moramo tudi vse odzive na izbiro iz padajočih menijev ali vprašanj, ki jih uporabniku zastavlja program, npr. ko program izpiše opozorilo: Ali ste prepričani, da ……? (Da/Ne). Za testiranje predstavlja en ekvivalentni razred odgovor D (tudi d), drugi razred pa predstavlja odgovor N (tudi n). Vsi ostali vnosi so neveljavni (lahko pa se smatrajo kot odgovor N). Za testni primer moramo določiti tudi ekvivalentne razrede, ki so časovno določeni, npr.: pritisni preslednico, v času, ko program bere določene podatke iz baze, diska,… Ekvivalentni razred v tem primeru je pravzaprav vse, kar naredimo v času, ko program bere podatke.

Za vsak ekvivalentni razred naj bi naredili vsaj dva testa. V začetku testiranja je smiselno najprej testirati veljavne vnose iz ekvivalentnega razreda, da preverimo, ali vnosi v polja v grobem delujejo pravilno, ko pa imamo za seboj že nekaj testiranja, pa vključimo tudi neveljavne vnose iz ekvivalentnega razreda. Najboljšo izbiro za testiranje ekvivalentnega razreda predstavljajo mejne vrednosti: največji, najmanjši, najdaljši, najkrajši, najhitrejši,... Poleg tega lahko z mejami ekvivalentnih razredov natančno preverimo neenakost (kot je >, >=, <, <=), saj takšne omejitve prinašajo največ odkritih napak. Program, ki brez težav prestane takšno testiranje, bo verjetno brez težav »preživel« tudi ostala testiranja. Tako na primer: za ekvivalentni razred, kjer veljavni vnos v polje predstavljajo števila od 1 do 99, meje ekvivalentnega razreda za veljavni vnos predstavljajo števila 1 in 99, za neveljavnega pa 0 in 100.

Pri testiranju mej ekvivalentnega razreda ne smemo pozabiti na izvedbo testiranja na računalniku, kjer je disk poln in podatkov ni možno več shranjevati, testiranje s tiskalnikom, ki nima papirja za tiskanje, ima malo spomina,…

Pazljivi moramo biti tudi pri pripravi testnih primerov, kjer se pomikamo po programu s pomočjo menijev in prehajamo do enega vnosnega ekrana preko različnih menijev. Kljub temu da je vnosni ekran isti, moramo testiranja ponoviti tolikokrat, kolikor je različnih dostopov preko menijev do vnosnih ekranov. Ker je kombinacij lahko veliko, poskušamo najti pot, za katero mislimo, da jo bodo uporabniki največkrat uporabljali oz. naključno izberemo več poti, ki nas preko različnih menijev pripeljejo do vnosnih ekranov.

Skrbno pripravo podatkov pa zahteva tudi regresijsko testiranje, saj morajo biti testni primeri izbrani tako, da lahko preverimo, ali je napaka odpravljena in ali v programu niso nastale kakšne nove napake.

3.7 Oblikovanje testne skupine

Učinkovito testiranje zahteva tudi primerno oblikovanje testne skupine. Testiranje informacijskih sistemov lahko opravijo predstavniki različnih vpletenih strani:

- proizvajalec ali dobavitelj,
- kupec, uporabnik ali potrošnik,
- neodvisna organizacija (laboratorij, neodvisni preizkuševalci).

Pri oblikovanju testne skupine je pomembno, da je le-ta del projektnega teama, ki je zadolžen za izgradnjo informacijskega sistema. Brez formalne potrditve izbranih članov testne skupine je težko uvajati testiranje v vse faze razvoja sistema. Prav zato se je treba z oblikovanjem skupine ukvarjati že na samem začetku. Naloge, ki jih mora opraviti testna skupina, so:

- poskrbeti mora, da je testiranje vključeno v projektni plan,
- pripraviti mora testni plan,
- pregledati časovni plan in zagotoviti, da je v tem času sposobna izvesti vsa potrebna testiranja,
- o statusu testiranja mora sproti poročati vodstvu projektnega teama,
- zagotoviti mora testiranje v vseh razvojnih fazah.

Za vsak projekt izgradnje informacijskega sistema je v opisu področja, ki naj bi ga pokrival, opredeljena tudi velikost skupine, ki bo zadolžena za izvajanje testiranja. Načeloma velja, da se velikost skupine od projekta do projekta razlikuje, kar je odvisno od velikosti razvijajočega sistema, časovnega okvira, ki je na razpolago za testiranje, itd.

Kako torej izbrati člane testne skupine?

Naloga preizkuševalca je, da odkrije napako, da jo odkrije čimprej in da poskrbi, da se napako tudi odpravi. Naloga vodje projekta pa je, da v testno skupino izbere člane, ki jim zaupa in verjame, da bodo svojo nalogo uspešno opravili. Izbira članov je odvisna še od ciljev in tehničnega področja projekta, vendar pa se lastnosti, ki jih mora imeti član testne skupine, na splošno ne razlikujejo dosti od lastnosti članov, ki jih izbiramo v katerokoli drugo skupino ali projekt. Pri kandidatu za testno skupino bomo torej iskali naslednje lastnosti (Stare, 2001, str. 18):

- imeti mora delovno zavest in čutiti podrejenost uresničevanju ciljev projekta,
- dobro tehnično znanje in sposobnost prevzemanja odgovornosti, pripravljenost na reševanje problemov in zastojev, biti mora podjeten, a odprt za sugestije,
- razumeti in izpolnjevati mora časovni plan, pripravljen mora biti delati izven delavnega časa, če je treba, biti mora prilagodljiv in sposoben prehajanja iz ene naloge na drugo,
- komunikativnost in tolerantnost, biti mora »teamski igralec« in ne vase zagledan heroj, znati mora pomagati drugim in pustiti drugim, da mu pomagajo,
- sposobnost delovanja pod dvema nadrejenima in mimo formalne strukture podjetja.

Poleg zgoraj naštetega pa gre tu še za vrsto drugih sposobnosti, ki se jih pričakuje od izbranega člana testne skupine: zanima nas npr., kako dobro pozna načine testiranja, kako iznajdljiv je pri uporabi novega sistema, ali pozna orodja za testiranje, pomembna je njegova sposobnost komuniciranja, kako dobro opazuje, je potrpežljiv, zna predvideti (uganiti), kje se lahko pojavi napaka,… Dober preizkuševalec mora biti raziskovalec, biti

mora neusmiljen, nepopustljiv in vztrajen. Iskana lastnost pri preizkuševalcu je tudi kreativnost – testiranje obveznega zanj ni dovolj, izmišljati si mora nove načine, da bi odkril napake, biti mora skoraj perfekcionist, hkrati pa tudi preudaren. Tu ne gre niti brez taktike in diplomacije, saj je preizkuševalec pogosto prinašalec slabih novic. Če ob vseh navedenih lastnostih pozna še osnove programiranja, pa je to lahko še dodatna prednost (Zorko, 2004). Smiselno je razmisliti tudi o vključitvi vsaj enega programerja v testno skupino, h kateremu se bodo lahko ostali obrnili po tehnično pomoč. Zavedati pa se moramo, da programerji niso nujno tudi dobri preizkuševalci, slab programer navadno tudi slabo opravlja testiranje.

V testno skupino bi morali vključiti člane, ki imajo določena znanja in odgovornosti s problemskega področja, ki ga bo pokrival nov informacijski sistem, končne uporabnike sistema, administratorje podatkovnih baz, skrbnika sistema kakovosti, varnostnega inženirja, testne inženirje,… Pri oblikovanju seznama potencialnih kandidatov za testno skupino bi za vsakega od njih morali vedeti, kakšne so njegove sposobnosti in seveda tudi, kakšne sposobnosti pričakujemo od kandidatov.

Testiranje kompleksnega sistema ni nič lažje kot sama izgradnja takega sistema, zato je še kako pomembno, kakšne člane bomo zbrali v testno skupino. Le–ti so namreč bistvenega pomena za uspešno izvedbo testiranja. Uspešna testna skupina lahko nudi veliko več kot samo iskanje in poročanje o napakah: s pomembnimi informacijami o izgradnji sistema oskrbuje celotno organizacijo. To so informacije o napakah in odpovedih, o delovanju sistema, povratne informacije o specificiranih zahtevah in načrtovanju, o statusu projekta,… Več informacij, kot jih je sposobna posredovati testna skupina razvijalcem, načrtovalcem, tehnologom, programerjem, večjo vrednost ima (Rothman, 2003).

3.8 Nevarnosti pri izvedbi testiranja

Pri organizaciji in pripravi testiranja lahko prihaja tudi do določenih odstopanj oz. nevarnosti, da testnega plana ne bo možno izvesti v skladu s pričakovanji. Vzroke za to lahko iščemo v (Pery, 2000, str. 213):

- Pomanjkanju izobraževanja: izvajalci testiranja večinoma nimajo posebnih izobraževanj s področja testiranja, kar je vzrok za nerazumevanje in napačno uporabo različnih tehnik testiranja.
- Miselnosti »mi proti vam«: problem nastane takrat, ko se izvajalci testiranja in razvijalci znajdejo na nasprotnih straneh, namesto da bi združili napore pri odkrivanju napak. Tak način pomeni nepotrebno trošenje energije in postavlja cilje testiranja ter tako celotnega projekta na stranski tir.
- Pomanjkanju testnih orodij: zelo pogosto vodstvo misli, da so testna orodja potrata oz. razkošje, ročno testiranje pa je lahko prevelik zalogaj. Učinkovitost ročnega testiranja pri velikih sistemih je podobna, kot če bi hoteli izkopati velik jarek z žlico.

- Pomanjkanju podpore testiranju s strani vodstva: podpora testiranju mora prihajati od vodstva podjetja, sicer ne moremo pričakovati, da bodo izvajalci testiranja svojo nalogo vzeli resno.
- Pomanjkanju vpletenosti končnih uporabnikov: pogosto so končni uporabniki izključeni iz procesa testiranja, lahko se zgodi, da niti ne želijo biti vpleteni. Kljub vsemu pa končni uporabniki predstavljajo eno pomembnejših vlog v procesu testiranja, saj lahko le oni potrdijo, da bo sistem ustrezno podpiral vse poslovne funkcije določenega poslovnega področja.
- Pomanjkanju časa za testiranje: časovna stiska je ena najbolj pogostih pritožb izvajalcev testiranja. Prav zato je priprava testnega plana, ki bo zagotavljal dovolj časa za učinkovito izvedbo, svojevrsten izziv.
- Zanašanju na neodvisne izvajalce testiranja: razvijalci se zavedajo, da bodo neodvisni izvajalci testiranja pregledali njihovo delo, zato se osredotočijo samo na programiranje in dopustijo, da preizkuševalci opravijo svoje delo. Na žalost pa to prinaša še več napak in samo podaljšuje čas testiranja.
- Hitrih spremembah: z nekaterimi tehnologijami (npr. RAD – Rapid Application Development) se sistemi razvijajo in spreminjajo zelo hitro, hitreje kot jih je možno testirati. Od tod izvirajo potrebe po avtomatizaciji: uporabi testnih orodji in seveda tudi uvedbi pazljivega označevanja različnih verzij sistema.
- Nasprotujočih situacijah, v kateri se znajdejo izvajalci testiranja: če prijavijo preveč napak, se jih pogosto krivi, da zavlačujejo z zaključkom testiranja, v nasprotnem primeru pa se jih krivi za slabšo kakovost izdelka.
- Dilemi, kdaj reči: »Ne!« – večna dilema, pred katero so postavljeni preizkuševalci, je kdaj reči, da program še ni pripravljen za prenos v produkcijsko okolje, saj so nenehno izpostavljeni pritiskom, da je treba testiranje zaključiti v predvidenem časovnem in stroškovnem okviru.

3.9 Klasične napake pri testiranju

Pri planiranju in izvajanju testiranja lahko hitro naredimo veliko napak. Nekatere se pojavljajo tako pogosto, da bi jih lahko že označili za klasične napake pri testiranju. V nadaljevanju navajam nekaj najpogostejših (Marick, 2004):

- Prva pogosta napaka je mišljenje, da je testna skupina tista, ki je zadolžena za zagotavljanje kakovosti sistema. To velja še posebno v primeru, ko se sama izvedba testiranja začne prepozno. Testna skupina ne more zagotavljati kakovosti, lahko samo zagotovi minimalno stopnjo le-te. Sistem bo zanesljiv, boljši in cenejši, če bodo vsi vpleteni v vseh fazah skrbeli za kakovost svojega dela in bodo zanjo tudi odgovorni.
- Prepozen začetek testiranja, priprava testiranja se mora namreč začeti že precej prej, preden se začne programiranje.

- Pogosta napaka je tudi prepozna priprava testnega plana, saj se le-ta pogosto pripravlja šele za funkcionalno testiranje.
- Stresno testiranje izvajamo tako rekoč v zadnjem trenutku, ob tem se pojavi vprašanje, kako popraviti napako, če tik pred prenosom v produkcijo ugotovimo, da sistem deluje slabo npr. za več kot 12 uporabnikov. Vsekakor je boljše, da to ugotovimo tik pred prenosom v uporabo, vendar bi se z izvedbo stresnega testiranja morali spopasti precej prej, tako bi se lahko izognili obilici dela ob zaključku projekta in tudi marsikateri stresni situaciji.
- Skoraj praviloma se ne izvaja testiranje dokumentacije in testiranje namestitve sistema.
- Izogibanje testiranju oz. preveliko zaupanje beta-testiranju, se je še dodatno razširilo s pojavom svetovnega spleta, saj se je pri menedžerjih sprožilo prepričanje, da beta-testiranje omogoča zmanjšanje stroškov testiranja. V povezavi s tem se pojavlja vprašanje, ali uporabniki sploh poročajo o vseh odkritih napakah, kako poročajo, koliko časa porabimo, da odkrijemo, kakšno napako so prijavili (neuporabna poročila), ne prijavljajo napak, ker niso prepričani, kakšni so bili vnosni podatki (nesistematično testiranje). Beta testiranje je načeloma zelo uporabno pri testiranju konfiguracije, saj vseh različnih možnih kombinacij konfiguracije, kot jih najdemo pri uporabnikih, nikoli ne moremo simulirati v testnem laboratoriju.
- Preizkuševalci se predolgo ukvarjajo s testiranjem enega področja v sistemu, tako da za ostala lahko zmanjka časa.
- V primeru testiranja izredno velikega sistema se lahko odločimo, da bomo natančneje testirali samo tista področja, ki jih najpogosteje uporablja največ uporabnikov. Hitro pa se lahko zgodi, da takšno področje napačno ocenimo.
- Za testiranje skoraj tradicionalno določimo nove zaposlene programerje.
- Za preizkuševalce določimo slabe programerje.
- Delo preizkuševalcev je fizično ločeno od npr. programerjev, načrtovalcev, čeprav bi morali vse čas delati skupaj in tesno sodelovati.
- Razmišljanje, da razvijalci sami ne morejo testirati svojega dela. Prav tako morajo razvijalci sami opraviti svoj del testiranja, saj so lahko pri svojem delu zelo uspešni, res pa je, da morajo za njimi testirati tudi neodvisni preizkuševalci.
- Premalo pozornosti se namenja pripravi planov za testiranje in preveč sami izvedbi testiranja. Pravi preizkuševalec mora biti sistematičen, tako da pripravi plana namenja veliko pozornosti, pripravi dobre testne primere, ki mu bodo hitro dali potrditev o pravilnosti oz. nepravilnosti delovanja sistema.
- Ni dovolj, da testiramo samo tisto, kar naj bi sistem delal, posebno pozorni moramo biti na tisto, kar naj sistem ne bi delal.
- Testni primeri so pripravljeni tako, da jih razume samo pripravljavec.

- Slabo poročanje o napakah, poročila pogosto ne navajajo natančnih informacij, kako je do napake prišlo, kaj v sistemu deluje narobe. Testno poročilo mora biti točno in natančno. Naloga preizkuševalcev je, da se naučijo pripraviti dobro testno poročilo in da je priprava takšnega poročila njihova glavna odgovornost.
- Slabo določanje prioritet odkritih napak oz. poročanje o napakah na tak način, da se zdi, da so odkrite napake nepomembne.
- Po odpravi napak se ne izvaja regresijsko testiranje.
- Skušamo avtomatizirati vsevprek, čeprav se določeni testi ne izvajajo dovolj pogosto, da bi upravičili avtomatsko testiranje.

4 METODE TESTIRANJA, TEHNIKE TESTIRANJA IN TESTNA ORODJA

4.1 Metode testiranja

Metode testiranja lahko delimo na dva načina. Pri prvi klasifikaciji je pomembno, kako so testi generirani, pri drugi pa je pomembno poznavanje razvijajočega sistema.

4.1.1 Klasifikacija metod, osnovana na načinu generiranja testov

Na način generiranja testov vplivajo različni faktorji: intuicija, predhodno znanje, specifikacija, struktura kode, narava aplikacije, predhodno odkriti problemi in podobno. Glede na to ločimo (Perry, 2000, str. 132):

- Metoda »ad hoc«, ki se nanaša samo na intuicijo in sposobnosti preizkuševalca ter izkušnje s podobnimi programi. Uporabnost metode lahko zelo niha: učinek je lahko zelo velik, če je preizkuševalec resnično strokovnjak ali pa zanemarljiv, če mu izkušenj primanjkuje.
- Metoda pretehtavanja vseh možnosti. Testne primere generiramo na osnovi vseh možnih kombinacij logičnih relacij med pogoji in funkcijami.
- Metoda mejnih vrednost – testni primeri se generirajo na osnovi mejnih vrednosti podatkov.
- Naključno testiranje - s to metodo preizkušamo delovanje funkcij z naključnimi vrednostmi.

4.1.2 Klasifikacija glede na poznavanje testiranega sistema

- Testiranje po metodi črne skrinjice (angl. black box) ne zahteva poznavanja same implementacije in interne logike programa, zahteva le poznavanje funkcionalnosti. Na osnovi tega znanja se generirajo testni primeri: definirajo se vhodni podatki in pričakovani rezultati, vmesno dogajanje pa je črna škatla. Ta metoda vključuje testiranje mejnih vrednosti, da pa bi zmanjšali število testnih primerov, le-te razdelimo v razrede. S to metodo odkrivamo (Solina, 1997, str. 183): napačne ali

manjkajoče funkcije, napake vmesnika, napake pri branju podatkov iz baze, nizko zmogljivost, napake pri inicializaciji,…

- Testiranje po modelu bele skrinjice (angl. white box), ponekod poimenovane tudi steklene skrinjice (angl. glass-box), pa zahteva poznavanje logike programa, na osnovi katere lahko razvijemo hipotetične testne primere. Sem uvrščamo metode ugibanja napak, naključno testiranje, itd. Med metode bele skrinjice sodita testiranje glavnih poti (vsi programski ukazi se izvedejo vsaj enkrat) in testiranje zank.

4.2 Tehnike testiranja

Ob prebiranju literature sem ugotovila, da avtorji uporabljajo različne izraze za načine testiranja. Solina (1997) na primer govori o postopkih testiranja, Perry (2000, str. 103) pa v svojem delu govori o tehnikah testiranja in pravi, da je tehnika testiranja proces, s katerim zagotavljamo, da posamezen sklop sistema ali celotni sistem deluje pravilno. V tem delu se bom zato tudi sama v nadaljevanju omejila na izraz tehnika testiranja. Tehnike testiranja lahko razdelimo na (Perry, 2000, str. 104 - 133):

- **Tehnike testiranja enot** (angl. unit test technique), s katerimi zagotavljamo pravilnosti delovanja posameznih enot sistema v skladu s pripravljenimi specifikacijami. V to skupino sodijo testiranja, ki jih opravi programer pri testiranju posameznega programa: odpravljanje sintaktičnih napak, logičnih napak, kar naredimo s pregledom kode (angl. structured walkthrough, code review) ali pa anonimno recenzijo (angl. peer review) druge osebe. Drugo osebo vključimo zato, ker avtor zaradi pogostega pregledovanja postane skoraj slep za nekatere napake in pomanjkljivosti. Testiranje nadaljujemo s preverjanjem posameznih modulov, za katere pa moramo sestaviti posebno testno konfiguracijo, ki bo nadomestila sistem, v katerega bo modul vgrajen. Testirane module postopno združujemo in testiramo več sklopov med seboj (angl. link testing, integration testing). Testiranje enot nadaljujemo s sistemskim testiranjem (angl. system test), ki vključuje testiranje serije modulov, ki so povezani v celoto.
- **Funkcionalne tehnike testiranja** (angl. system functional techniques) uporabljamo za preverjanje, ali zgrajeni sistem ustreza zahtevam, ki so podane v specifikacijah. Pri testiranju funkcionalnosti nas ne zanima, kako sistem deluje, pač pa le rezultat delovanja. Ta vrsta testiranja je ena izmed lažjih, saj ni potrebno poznavanje logike programa, zahteva pa primerno kreiranje testnih pogojev in listo zahtev funkcionalnosti. V sklop testiranja, ki preverjajo funkcionalnosti sistema, sodijo: uporabniško testiranje, regresijsko testiranje, paralelno testiranje, testiranje odziva sistema ob napaki, preverjanje vgrajenih kontrol v sistem ,…
- **Strukturne tehnike testiranja** (angl. system structural testing techniques) uporabljamo za zagotovitev, da je produkt razvit pravilno in da bo pravilno tudi deloval, da je bila tehnologija uporabljena na pravi način in da so vse komponente

med sabo ustrezno povezane. K strukturnim tehnikam testiranja sodi: stresni test, performančni test, obnovitveni test, operacijski test, varnostni test,…

4.3 Avtomatizacija testiranja in orodja za testiranje

Testiranje lahko izvajamo ročno ali s pomočjo orodij. Poznamo nekaj tehnik in veliko različnih orodij za testiranje. V prejšnjem poglavju smo rekli, da je tehnika testiranja proces, orodje za testiranje pa je sredstvo, ki ga uporabljamo za izvajanje tega procesa. Večina naporov je bila v zadnjih dvajsetih letih usmerjena prav v izboljšavo razvojnega procesa, testiranje pa je ostalo večinoma ročno, v zadnjem času pa je vedno več novih testnih metod, ki so izboljšale učinkovitost testiranja (Zorko, 2004). Ročno testiranje sreča na začetku svoje poti vsak preizkuševalec, pa tudi kasneje, ko je že izkušen, ne gre brez njega. Uporabimo ga povsod tam, kjer:

- ni dovolj časa za avtomatizacijo,
- ni izkušenega kadra, ki bi obvladal postopke avtomatizacije oz. avtomatizacija ni smiselna zaradi prevelikih stroškov in neponovljivosti,
- testiramo uporabniški vmesnik ali
- določamo izvor in ponovitev napake.

Testiranje je težko opravilo, vendar pa obstaja mnogo načinov, da se delo poenostavi in čas testiranja skrajša.

V praksi se pogosto dogaja, da moramo določene teste večkrat ponavljati. Kadar odkrijemo kakšno napako, je ponavadi, potem ko je odpravljena, treba določen del programske opreme testirati večkrat, da lahko potrdimo, da je napaka zares odpravljena in se niso pojavile nove napake. V tem primeru moramo presoditi, ali bomo take teste opravljali ročno ali pa jih bomo avtomatizirali in si pomagali z orodji. Prednosti avtomatizacije testiranja so (Kaner, 1998):

- hitrost – sodobna orodja omogočajo obdelavo ogromne količine podatkov v zelo kratkem času,
- učinkovitost – če testiramo ročno, v tem času ne moremo delati nič drugega. Z avtomatizacijo prihranimo čas, ki ga lahko porabimo npr. za planiranje testiranja,
- točnost in natančnost – ko izvedemo na stotine testov, človekova koncentracija pade, pojavljajo se napake, medtem ko orodje izvaja isti test in preverja rezultate z vedno isto natančnostjo in točnostjo,
- vztrajnost – testno orodje se nikoli ne utrudi, teste lahko opravlja vedno znova.

Če se odločimo za testiranje s pomočjo orodij, moramo posebno pozornost nameniti izbiri pravega orodja. Pravilna izbira je eden od pomembnejših faktorjev v celotnem procesu testiranja, saj je na tržišču moč najti množico orodij, ki pa so namenjena za točno določeno tehniko testiranja. Pozorni moramo biti tudi na to, da izberemo pravo orodje glede na fazo razvojnega procesa informacijskega sistema, v kateri bomo orodje uporabili. Orodja se med seboj razlikujejo tudi po tem, da nekatera zahtevajo veliko programerskega

znanja, druga pa so namenjena praktično vsem, ki imajo vsaj nekaj izkušenj s testiranjem. Vsekakor mora biti orodje prilagojeno testni skupini, njenemu znanju in izkušnjam. Na voljo je ogromno testnih orodij, ki znajo pregledovati kodo, ugotavljati slepe zanke in druge nepravilnosti, ugotavljati skladnost, optimirati kodo, simulirati uporabnikove akcije ter daljše časovno obdobje izvajati teste in zapisovati rezultate (Zorko, 2004).

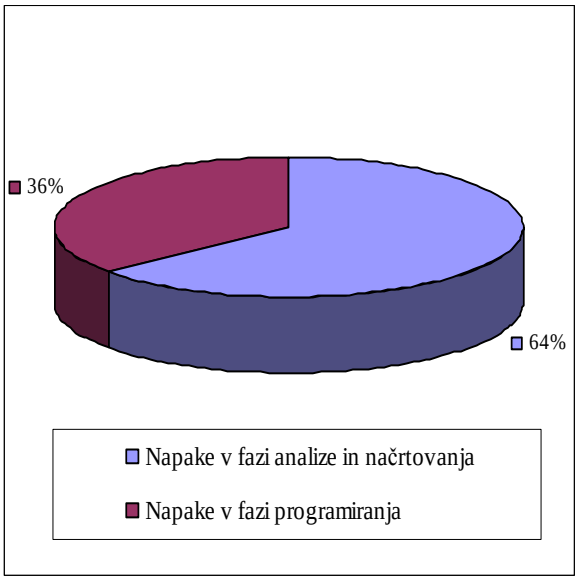
Kljub vsem naštetim dobrim lastnostim orodij za testiranje pa se moramo zavedati, da tudi avtomatizacija ni vedno upravičena. Glede na to, da želimo testiranje opraviti s čim manj stroški, mora biti odločitev za avtomatizacijo testiranja racionalna. Priprava nanj zahteva zares pazljivo planiranje in organiziranje ter predstavlja časovno zelo velik zalogaj (Marick, 2004a). Razvijalci potrebujejo povratne informacije o uspešnosti svojega dela čimprej, prav lahko se zgodi, da bodo takrat, ko bomo z avtomatskim testiranjem našli napake in o njih tudi poročali, razvijalci že delali na drugem projektu. Veliko časa bomo porabili tudi za pripravo testnih primerov, ki pa jih moramo spreminjati in preverjati ob vsaki spremembi, ki jo razvijalci naredijo v programu. Pri odločitvi za avtomatizacijo testiranja moramo razmisliti tudi o tem, kako dolgo bodo pripravljeni avtomatizirani testi še primerni za testiranje in koliko sprememb v programu bodo še prenesli, da bodo dajali še vedno enako dobre rezultate. Sama odločitev za avtomatizacijo testiranja še ne prinaša zagotovila, da bomo zares odkrili vse napake, zavedati pa se moramo, da ravno tako obstaja tveganje, da bomo določeno napako spregledali. Kaner (1999) v svojem delu navaja, da morajo biti tako pripravljavci plana testiranja in organizacije kot tudi izvajalci avtomatskega testiranja naravnost pikolovski. Rezultate testiranja je treba natančno pregledati, saj se v nasprotnem primeru lahko zgodi, da bomo spregledali zares velike napake, ki bi bile pri ročnem testiranju lahko odkrite mimogrede. Zaradi zgoraj naštetega torej ne moremo trditi, da je ročno testiranje slabše od testiranja s pomočjo orodij za testiranje. Kdaj se odločiti za eno in kdaj za drugo možnost, je odvisno od okolja, v katerem testiramo, sistema, ki ga testiramo, časa, ki ga imamo na voljo, razvojne faze, v kateri se nahajamo in še mnogih drugih dejavnikov. Upoštevati moramo, da se človek lahko hitro prilagodi novo nastali situaciji, vendar je utrudljiv in dalj časa ko opravlja testiranje, več napak lahko pričakujemo pri njegovem delu.

Odločitev za ali proti avtomatizaciji bo odvisna tudi od vrste testiranja, ki jo izberemo: v primeru izvajanja npr. obremenilnega testiranja je uporaba orodij nujna, saj bi bilo organizirati testiranje za npr. 300 uporabnikov prevelik strošek, verjetno bi imeli precej težav že pri organizaciji, manj primerno ali neprimerno pa bo uporabiti avtomatsko testiranje za testiranje vseh vrst dokumentacije, namestitve sistema, testiranje mejnih vrednosti, preizkus delovanja funkcionalnosti sistema,… (Nguyen, 2001). Za vsako ceno torej ne bi imelo smisla vztrajati pri ročnem testiranju, niti kategorično odklanjati uporabe orodij. Dobršna mera razmisleka in uporaba kombinacije obojega se bo verjetno tudi v tem primeru izkazala kot prava odločitev.

5 TESTIRANJE V POSAMEZNIH FAZAH RAZVOJA INFORMACIJSKEGA SISTEMA

Ob testiranju informacijskega sistema najprej pomislimo na testiranje programske opreme, preden je ta predana v uporabo končnim uporabnikom. Tudi tradicionalni pristopi k razvoju programske opreme postavljajo testiranje kot eno izmed samostojnih faz razvojnega procesa, ki jo začnemo izvajati po integraciji in izvedbi sistema ter preden začnemo z vzdrževanjem. Novejši pristopi pa uvajajo testiranje v vse faze življenjskega cikla razvoja informacijskega sistema, saj se večina napak pojavlja v zgodnejših fazah razvoja sistema.

Slika 10: Nastanek napak v fazah razvoja informacijskega sistema



Vir: Perry, 2000, str. 45

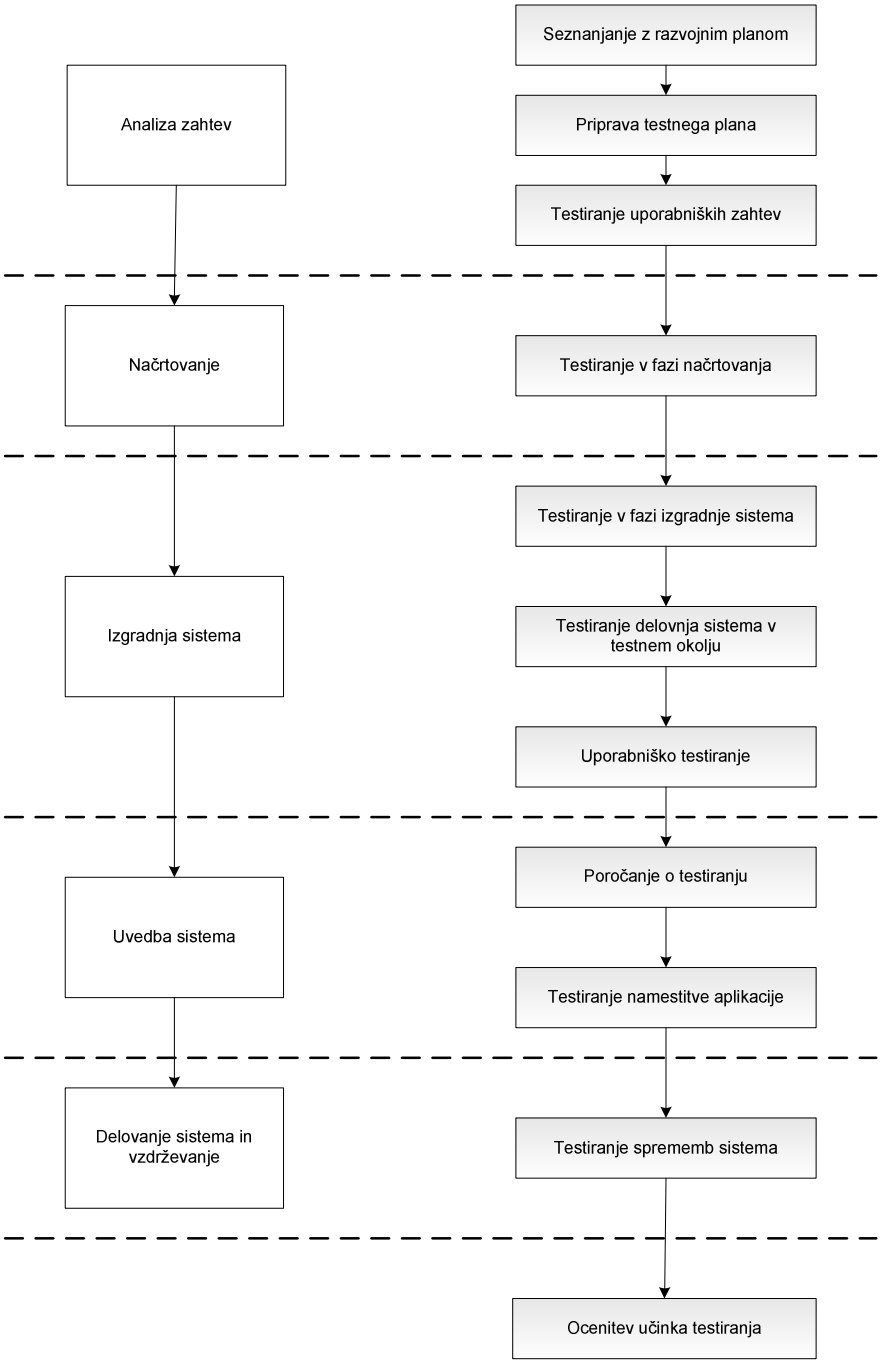
Slika 10 prikazuje odstotek nastanka napak v fazi analize in načrtovanja v primerjavi z odstotkom napak, ki se pojavijo v fazi programiranja novega informacijskega sistema. Kar 64% vseh napak nastane prav v začetnih fazah razvoja informacijskega sistema (faza analize in načrtovanja) (Perry, 2000). Odkrivanje in odpravljanje napak v začetnih fazah razvoja pomeni tudi velik korak pri zniževanje stroškov razvoja sistema.

Uspešnost vsake razvojne faze se preverja s t. i. recenzijami, ki so lahko bolj ali manj formalne narave. V začetnih fazah razvoja v povezavi s testiranjem govorimo predvsem o verifikaciji, kjer se preverja, ali produkt v posamezni fazi ustreza zahtevam, postavljenim v predhodni fazi in preverja tehnično pravilnost. Validacija pa je proces vrednotenja programske opreme na koncu njenega razvoja z namenom, da se ugotovi skladnost z zahtevami (Dogša, 1993). Rezultati obeh, tako varifikacije kot validacije, pa morajo biti dokumentirani v testnem poročilu, ki se spreminja in dopolnjuje skozi celoten proces razvoja sistema.

V nadaljevanju bom prikazala, kako izvajamo testiranje v posameznih fazah razvoja sistema. Testiranje je proces, ki se mora začeti hkrati z začetkom projekta za izgradnjo informacijskega sistema in ga izvajamo v več korakih (Perry, 2000, str. 170).

Na sliki 11 so prikazani posamezni koraki kot si sledijo v procesu testiranja skozi posamezne faze razvoja. Leva stran slike prikazuje korake v razvoju informacijskega sistema, desna stran pa korake v procesu testiranja, ki jih izvajamo v posamezni fazi razvoja sistema. V nadaljevanju bom podrobneje opisala vsako fazo testiranja.

Slika 11: Koraki v procesu testiranja



Vir: Shelly et al., 1998

5.1 Testiranje v fazi zbiranja uporabniških zahtev

5.1.1 Seznanjanje z razvojnim planom

Da bi preizkuševalci lahko pripravili dober testni plan, se morajo seznaniti z razvojnim planom projekta. To jim pomaga (lahko tudi samo vodji preizkuševalcev) pridobiti potrebne informacije o tem, za kakšen projekt gre, kakšni so glavni cilji projekta, kako obširen je projekt in v kakšnem časovnem obdobju se bo izvajal. Preizkuševalci morajo v tem koraku dobiti vse potrebne informacije za pripravo testnega plana: natančno morajo vedeti, koliko preizkuševalcev bo potrebnih za testiranje, kdaj bodo morali biti na razpolago, kakšna oprema za testiranje bo potrebna,… Ob tem bodo preizkuševalci dobili tudi dodatne informacije o tem, kakšen je trenutni status projekta, seznanijo se z aktivnostmi, ki so trenutno v teku na projektu. Razvijalci informacijskega sistema in preizkuševalci morajo začeti z delom istočasno, zato je ta trenutek tudi pravi, da stečejo aktivnosti za oblikovanje testne skupine.

5.1.2 Priprava testnega plana

Testni plan pripravljamo v več korakih. Pri vsakem si lahko pomagamo z vnaprej pripravljenimi dokumenti ali obrazci, ki pripravljavca planov lahko vodijo pri postopni pripravi plana. Prvi korak pri pripravi testnega plana je **definiranje ciljev testiranja**. Cilji morajo biti definirani tako, da so merljivi, hkrati pa ne smejo odstopati od ciljev, ki veljajo za celoten projekt. Pazljivi moramo biti tudi pri definiranju števila ciljev testiranja, saj obstaja nevarnost, da bi v primeru prevelikega števila, preizkuševalci ne dosegli nobenega od njih. Neko priporočljivo število, ki ga zasledimo v literaturi, je 10 ciljev (Perry, 2000, str. 226). Vsakemu cilju določimo prioriteto. Načeloma velja, da ima od vseh ciljev, ki jih definiramo, tretjina nizko prioriteto, tretjina srednjo, preostala tretjina pa visoko prioriteto.

Tabela 1: Testna matrika

Vrsta testiranja / **Modul sistema**	Pregled kode	Testiranje enote	Paralelno testiranje	Funkcionalno testiranje
Iskanje komitenta v registru komitentov	√	√		√
Vnos kredita	√	√		√
Prenos podatkov v glavno knjigo	√		√	√

Vir: Perry, 2000

V drugem koraku naredimo **testno matriko**, ki je ključna komponenta vsakega testnega plana. Testna matrika, ki je predstavljena v tabeli 1, ima na levi strani navedene module sistema, na drugi strani pa vrste testiranja, ki se bodo izvajale. V matriki nato za vsak

modul sistema označimo, katere vrste testiranja bomo uporabljali. Tako dobimo natančen pregled na tem, da ima vsaka funkcionalnost v sistemu, ki ga testiramo, predpisano vsaj eno testiranje in je vsaka vrsta testiranja uporabljena vsaj za testiranje ene funkcionalnosti.

Tretji korak pri pripravi testnega plana pa je določanje t. i. **testne administracije**. Administrativna komponenta v testnem planu prikazuje urnik in vire, ki so potrebni za izvršitev testnega plana, ter ključne dogodke, ki jih lahko uporabimo tudi za nadzorovanje statusa oz. spremljanje napredka testiranja. Da bodo preizkuševalci dobili nekaj splošnih informacij o projektu, je priporočljivo zapisati zgodovino poteka izgradnje informacijskega sistema.

Zadnji korak pri pripravi plana pa je **oblikovanje ustreznega dokumenta.** V večjih organizacijah je pogosto v navadi, da je testni plan nek predpisan dokument, kjer je že navedeno, katere elemente naj vsebuje testni plan. Načeloma naj bi dokument tvorili štirje sklopi (Perry, 2000, str. 232):

1. V prvem delu zapišemo **splošne informacije** o razvijajočem sistemu. To poglavje je namenjeno nekemu splošnemu pregledu funkcij, ki jih bo imel sistem, opišemo, komu je namenjen in v kakšnem okolju bo deloval. Naredimo kratek pregled zgodovine sistema, predstavimo organizacijo, kjer se bo testiranje odvijalo, opišemo morebitna predhodna testiranja in predstavimo rezultate tega testiranja. Navedemo tudi cilje testiranja, pripravimo pričakovano oceno napak (če imamo s podobnim programom že nekaj izkušenj) in navedemo, kateri so še referenčni dokumenti, ki bi bili lahko v pomoč pri izvajanju testiranja.

2. **Drugi del predstavlja načrt testiranja**, ki naj vsebuje natančen opis sistema, ki ga bomo testirali. V tem delu predstavimo testno skupino – navedemo seznam preizkuševalcev in kakšne so posameznikove zadolžitve, popišemo mejnike oz. ključne dogodke, ključne datume, ki jih mora testna skupina upoštevati. Navedemo tudi proračun, s katerim razpolaga testna skupina, izdelamo časovni plan testiranja in natančno definiramo, kdaj se bo izvajala katera od funkcij testnega plana, opredelimo opremo, ki jo bomo potrebovali (npr. ostala programska oprema, dodatna strojna oprema, ljudje). Navedemo tudi dokumentacijo sistema, testne podatke, testno dokumentacijo in testna orodja. V tem poglavju tudi opredelimo, ali bo testna skupina potrebovala kakršnokoli dodatno izobraževanje s področja testiranja.

3. **Opis testiranja in vrednotenje rezultatov**: predstavimo seznam poslovnih funkcionalnosti, ki so bile osnova za pripravo sistemskih specifikacij. Predstavimo testiranje, ki ga bomo izvajali oz. navedemo, kakšna je metodologija testiranja, navedemo testna orodja, ki jih bomo uporabljali pri testiranju, morebitne omejitve v povezavi z opremo, izvajalci testiranja, podatkovno bazo,… Opišemo tudi način, s katerim bomo merili učinek testiranja, npr.: zaloga vrednosti, različne kombinacije.

4. **Izvedba testiranja**: opišemo, kako se bo testiranje izvajalo, navedemo kontrole, ki jih moramo upoštevati, predstavimo, kakšni so vhodni podatki, ki jih bomo uporabljali pri

testiranju, izhodni podatki in vmesna sporočila, ki jih lahko pričakujemo kot rezultat testiranja, po korakih pa opišemo tudi procedure, po katerih izvajamo testiranje.

S pripravo testnega plana moramo začeti dovolj zgodaj, navadno je za njegovo pripravo zadolžen vodja testiranja. Ko je pripravljen, ga predstavi testni skupini. Glede na to, da gre pri razvoju informacijskih sistemov za dinamičen proces in se zahteve hitro spreminjajo, moramo sproti v odvisnosti od sprememb prilagajati tudi testni plan. Napisan mora biti tako fleksibilno, da lahko na enostaven način spreminjamo testne primere, testne podatke, hkrati pa ne sme biti preobsežen in zapleten. Koristno je, da je tudi s stališča kakovosti večkrat pregledan.

Po pripravi testnega plana lahko začnemo z izvedbo testiranja v posameznih fazah. Najprej je na vrsti testiranje zahtev, ki so jih pripravili bodoči uporabniki sistema.

5.1.3 Preverjanje uporabniških zahtev

Preverjanje sistemskih zahtev v fazi analize uporabniških zahtev je ena pomembnejših faz, saj se v njej formirajo najpomembnejše odločitve o tem, kako bo sistem deloval. V tej fazi morajo bodoči uporabniki sistema natančno opredeliti, kakšen sistem želijo uporabljati. Zapisane uporabniške zahteve so osnova za izgradnjo sistema. Uvedba testiranja oz. preverjanje uporabniških zahtev povečuje verjetnost, da bodo pripravljene uporabniške zahteve dobre.

Skozi celoten proces testiranja se pomembnost posameznih vlog pogosto zamenja. Pomembno je, da določimo, kdo je odgovoren za izvedbo testiranja in na čigavi strani je odgovornost za testiranje v posamezni fazi. V nekaterih fazah imajo dominantno vlogo uporabniki, v drugih pa informatiki. Določitev odgovornosti za testiranje ne pomeni nujno tudi odgovornosti za izvajanje testiranja – odgovornost pomeni sprejeti oz. zavrniti izdelek glede na rezultate testiranja. Glede na to, da je uvajanje testiranja v vse faze razvojnega cikla še razmeroma nov koncept, se pri definiranju odgovornosti za testiranje zaenkrat še ne moremo zanašati na standarde oz. izkušnje. Prav zato je najbolje, da so odgovornosti za testiranje opredeljene v testnem planu. Glavno vlogo v fazi analize zahtev imajo uporabniki, informatiki pa so tisti, ki zahteve sprejemajo ali zavračajo oz. predlagajo alternativno rešitev. Tako morajo uporabniki prevzeti tudi vso odgovornost za izvedbo testiranja v fazi analize zahtev, prav tako pa morajo biti vključeni v celoten proces testiranja.

Že večkrat sem omenila, da nepopolne, netočne in neskladne zahteve uporabnikov vodijo v največje napake pri izgradnji sistema. Prav zato je naloga izvajalcev testiranja v tem koraku potrditi, da so uporabniške zahteve jasne, točne, popolne in da ena zahteva ni v nasprotju z drugo. Če bi napako v koraku preverjanja zahtev spregledali, bi to lahko povzročilo serijo napak v naslednjih razvojnih fazah. Na podlagi napačnih specifikacij bi sistem napačno načrtovali, posledično bi programerji napisali napačne programe, ki v resnici ne bi podpirali poslovnih funkcij, ki jih uporabniki izvajajo vsak dan. Da bi takšno napako popravili, bi znova morali začeti z analizo zahtev, spremembo specifikacij, načrtovanja, spremembo programov,… To pa zahteva dodaten čas in denar, ki sta v

razvoju vedno kritični komponenti. Prav zaradi tega moramo težiti k temu, da se napake odpravljajo čimbolj zgodaj.

Preverjanje uporabniških zahtev se začne, ko so le-te definirane v celoti. Najpogostejši način preverjanja je ponovni pregled uporabniških zahtev. Priporočljivo je, da testno skupino sestavljajo ljudje z določeno mero izkušenj in sposobnosti, da na podlagi zapisanih zahtev lahko hitro ugotovijo neskladnost oz. pomanjkljivost v sistemu. Preizkuševalci morajo pri preverjanju ponovno pretehtati vse možne scenarije, tako tiste pravilne možnosti izvajanja postopka kot tudi vsa odstopanja in izredne dogodke. Navadno pri preverjanju niso težava običajni, ustaljeni postopki, problematična so prav odstopanja od njih. Eden od precej zanesljivih načinov, kako zagotoviti, da res ne pozabimo na nobeno od možnosti, je priprava seznama vseh tveganj, ki jih za sistem lahko identificiramo. Če seznama tveganj niso pripravili uporabniki, za to lahko poskrbi testna skupina. Za vsako od identificiranih tveganj tako lahko predvidimo kontrole, za katere pričakujemo, da bodo vgrajene v sistem in morajo biti zapisane v uporabniških zahtevah. Glavna prednost tega načina je, da ugotovimo kaj pričakujemo od sistema v povezavi z ugotovljenimi tveganji. Pri tem upoštevamo tveganja v povezavi z nedovoljenim dostopom do aplikacije, nezadostnimi varnostnimi mehanizmi, proceduralne napake, programske napake, napake operacijskega sistema, napake na komunikacijah,…

Perry (2000, str. 235) v literaturi navaja, da je ponoven pregled zapisanih uporabniških zahtev učinkovit način testiranja v fazi zbiranja uporabniških zahtev, seveda pod pogojem, da člani razvojne skupine sodelujejo in pripravijo odgovore na vsa vprašanja in priporočila, ki jih postavlja testna skupina. Odgovornost testne skupine je navadno omejena na priporočila, pripombe in vprašanja, na uporabnikih pa je, da jih znajo koristno uporabiti. Tako delo testne skupine ne bo samo sebi namen, pač pa bo aktivno pripomoglo k pravilnosti in kakovosti definiranja zahtev.

Po končanem pregledu testna skupina pripravi poročilo o testiranju. Tu so zapisana vsa priporočila in pripombe. Če so bile pri pregledu ugotovljene pomanjkljivosti, razvojna uporabniška skupina popravi napake in preverjanje zahtev se izvede ponovno. Ko je tako testna kot tudi uporabniška skupina zadovoljna z opravljenim delom, se izvrši še formalna potrditev – podpis ustreznega dokumenta, s katerim preizkuševalci potrdijo, da je bilo testiranje opravljeno tako, da:

- napisane specifikacije brezpogojno predstavljajo zahteve uporabnikov,
- so zahteve definirane in dokumentirane,
- je bila izvedena analiza ekonomske upravičenosti (angl. cost/benefit analysis),
- so bili rešeni poslovni problemi,
- so bile zavedene zahteve za izvedbo kontrol,
- je poslovni proces sledljiv in podpira določeno poslovno področje,
- je med morebitnimi različnimi alternativami izbrana najbolj primerna rešitev.

5.2 Testiranje v fazi načrtovanja

V fazi načrtovanja določimo, kako bomo identificirane zahteve iz prejšnje faze pretvorili v strukturo, ki bo implementirana v sistemu. V tej fazi morajo tesno sodelovati uporabniki in načrtovalci sistema. Če smo za fazo analize in zbiranja zahtev zatrjevali, da imajo dominanto vlogo uporabniki, za to fazo ne morem reči, da je vloga katerega od njih prevladujoča, pravzaprav sta obe strani enako odgovorni za to, da se bodo identificirane uporabniške zahteve pravilno odražale v načrtovanem sistemu. Zelo je pomembno, da obe strani sodelujeta z roko v roki in se skupaj trudita, da bo bodoči sistem učinkovit in predvsem sprejemljiv za uporabnika.

Sam sistemski načrt v projektnem teamu naredijo informatiki, saj uporabnikom pogosto manjka poznavanja orodij za načrtovanje, tako da je skoraj logično, da so za izvedbo načrta tudi odgovorni. Prav tako kot samo načrtovanje pa si morata obe strani deliti tudi testiranje v fazi načrtovanja. Odgovornost za testiranje v tej fazi lahko prevzame kar projektni team, še posebno, če ga sestavljajo tako uporabniki kot tudi informatiki. Priporočljivo je, da so v testno skupino vključeni tudi neodvisni izvajalci testiranja. Tu gre predvsem za člane skupine, ki sicer niso tesno povezani s projektom načrtovanja informacijskega sistema: so člani drugih projektov ali pa so to člani skupine za zagotavljanje kakovosti. Zaradi neodvisnosti in izkušenj iz drugih projektov lahko koristno pripomorejo h kritičnemu pregledu načrta sistema. Zaželeno je, da tovrstno testiranje opravijo preizkuševalci, ki imajo znanje s področja načrtovanja, poznajo metodologijo načrtovanja, saj bo samo tak preizkuševalec hitro opazil morebitne pomanjkljivosti v načrtu. V nasprotnem primeru bo verjetno bolj obremenjen z notacijo, ki jo uporabljamo za načrtovanje, kot pa z vsebino in preverjanjem pravilnosti nastalega načrta.

V fazi načrtovanja moramo pripraviti t. i. načrt bodočega sistema. Vsak načrt je najlažje spreminjati in dopolnjevati v začetnih fazah, zato moramo vključiti testno skupino že zelo zgodaj oz. čimprej, ko se začne faza načrtovanja. Zgodaj odkrite napake pomenijo zniževanje stroškov pri njihovem odpravljanju. Če bi s testiranjem čakali do zaključka faze načrtovanja, bi se ujeli v enako past kot tisti projekti, ki s testiranjem čakajo do zaključka programiranja, ko je napaka odkrita, to pa pomeni časovno in stroškovno velik zalogaj za izvedbo projekta izgradnje informacijskega sistema.

Testiranje v fazi načrtovanja poteka s ponovnim pregledom pripravljenega načrta. Ena od možnosti izvedbe pregleda je tudi s pripravo predstavitve, ki jo pripravi načrtovalec sistema oz. oseba iz razvojne skupine, ki je sodelovala pri načrtovanju. S predstavitvijo je način komunikacije med izvajalci testiranja in projektnim teamom najlažja, saj omogoča sprotno razčiščevanje problemov, hkrati pa preizkuševalci niso obremenjeni s poznavanjem notacije in metodologije načrtovanja, saj predstavitev poteka z opisovanjem nastalega načrta.

Glavni cilji testiranja v fazi načrtovanja so (Perry, 2000, str. 320):

- potrditi, da je načrt informacijskega sistema pripravljen v skladu z uporabniškimi zahtevami,
- potrditi, da so definirane specifikacije in postopki za ročne in avtomatizirane dele sistema,
- potrditi, da načrt rešuje določen poslovni problem na najboljši možen način: upoštevane morajo biti vse kontrole, ki bodo bistveno zmanjšale tveganja v sistemu, podprta ročna in/ali avtomatska avtorizacija uporabnikov, sistem mora biti enostaven za uporabo, določena mora biti sledljivost sprememb podatkov (angl. audit trail), načrt mora biti narejen tako, da bo sistem možno vzdrževati, nadgraditi, določene morajo biti procedure za dostop do informacij (pooblastila) in povezave z drugimi sistemi, definirani morajo biti tudi ročni postopki v primeru nepredvidenih nesreč in podobno,
- potrditi skladnost načrta z (internimi) predpisi, standardi in veljavno metodologijo.

Testna skupina v skladu z zgoraj navedenimi cilji preveri korake in izdelke, ki nastajajo v fazi načrtovanja. Ti so navadno predpisani z metodologijo in internimi standardi. Pregled v fazi načrtovanja vedno izvajamo v več iteracijah, število le teh pa je odvisno od pomembnosti projekta, časa, ki ga imamo po planu na voljo za posamezno fazo in seveda števila napak, ki so bile odkrite v prejšnjem pregledu. Po opravljenih spremembah v načrtu morajo preizkuševalci vedno opraviti še en pregled, kjer naj bi odkrivali morebitne nove napake.

Ob zaključku pregleda mora testna skupina pripraviti priporočila in dopolnitve, ki jih nato pregleda skupaj s projektno skupino, s katero se tudi dogovori, ali bodo predlagane dopolnitve in spremembe upoštevane ali ne. Ob koncu se izda uradno poročilo, kjer so zavedene vse ugotovitve in priporočila. Odvisno od internih predpisov v podjetju je, ali poročilo v pregled dobijo tudi vodje, vsekakor pa je poročilo del projektne dokumentacije.

Načinov, kako pripraviti načrt sistema, je verjetno toliko, kolikor je tudi načrtovalcev sistema. Gre za ustvarjalno delo, ki ga vsak opravi na nekoliko drugačen način. Prav zaradi tega je tudi testiranje v tej fazi pogosto precej naporno, še zlasti zaradi tega, ker je vedno treba doseči nek kompromis; najprej med načrtovalci in uporabniki ter potem še med člani testne skupine. Prav zato je že pri pripravi plana treba misliti na to, da bo časa za testiranje v tej fazi dovolj, saj samo z večkratnimi ponovitvami testiranja lahko pridemo do dobrega načrta sistema.

Po opravljeni verifikaciji in pridobljenih potrditvah v fazi načrtovanja lahko delo pri razvoju informacijskega sistema preide v naslednjo fazo – fazo izgradnje sistema.

5.3 Testiranje v fazi izgradnje sistema

Programiranje je naloga, ki jo izvajajo izključno informatiki. Vpletenost uporabnikov je minimalna, razen v primeru, ko se pojavijo določena vprašanja, ki jih je treba razrešiti z njihovo pomočjo. Pogosto se zgodi, da uporabniki še v fazi izgradnje sistema poskušajo spreminjati svoje zahteve. Če je le mogoče, takšne spremembe zahtev zavrnemo oz. jih

preložimo na čas, ko bo sistem že deloval v produkcijskem okolju. Če te možnosti nimamo oz. bi bilo delovanje sistema brez teh sprememb nepopolno in nepravilno, moramo zahteve implementirati takoj, seveda po predpisanem razvojnem procesu: dopolnitev specifikacij uporabnikov in testiranje zahtev, spremembe načrta sistema in izvedbe testiranja v fazi načrtovanja.

Kako zapletena bo faza programiranja, je precej odvisno od izdelkov, ki jih pripravimo v fazi načrtovanja. Dobro definirane programske specifikacije lahko zelo poenostavijo programiranje, zavedati pa se moramo, da bodo napake, ki smo jih spregledali v prejšnjih fazah, v tej fazi vgrajene v sistem. Računati moramo tudi na napake, ki jih bo pri svojem delu naredil še tako pazljiv programer. Dela je torej za izvajalce testiranja dovolj tudi v tej fazi, začetek testiranja pa naj bi se čimbolj skladal z začetkom programiranja sistema. Odkrivanje in odpravljanje napak med programiranjem sistema bo prav gotovo cenejše kot odpravljanje napak med testiranjem končnega programa.

Fazo izgradnje sistema lahko razdelimo na tri aktivnosti (Perry, 2000, str. 370):

- napišejo se programske specifikacije na podlagi specifikacij, ki so nastale v fazi načrtovanja,
- na podlagi pripravljenih programskih specifikacij programerji napišejo ustrezne programe,
- programer preveri, ali se program izvaja v skladu z zahtevami in daje zahtevane rezultate.

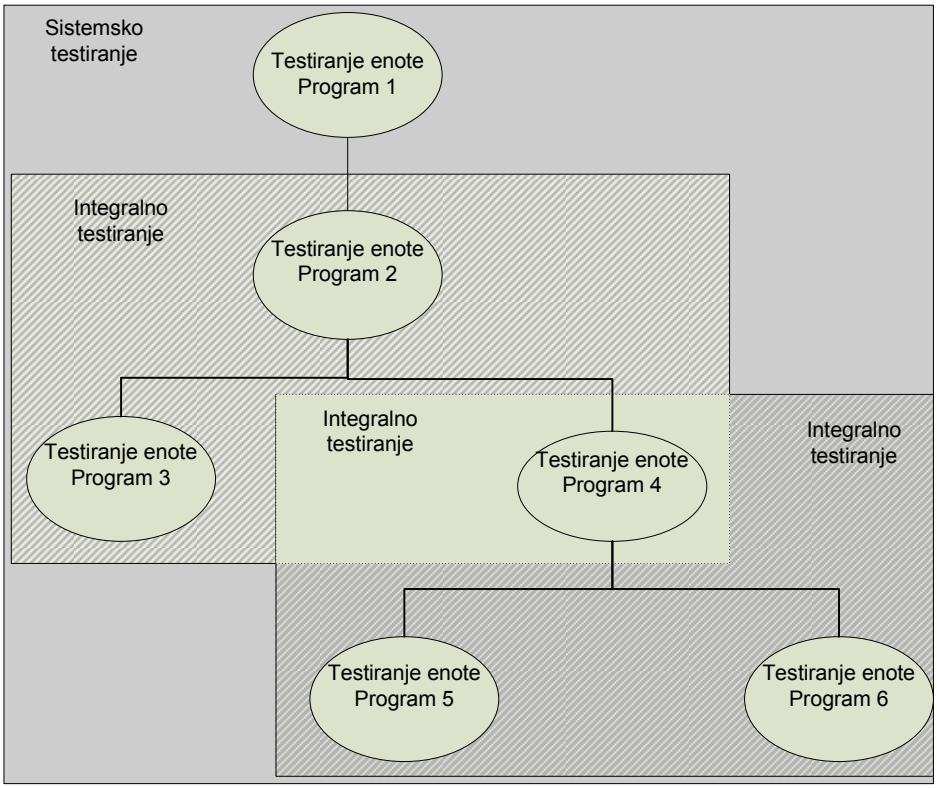
Odgovornost za testiranje mora prevzeti vodja informatikov v okviru projekta. Glavni cilj testiranja je zagotoviti, da so specifikacije, ki so bile izdelane v fazi načrtovanja, pravilno implementirane v sistem. Ker gre za tehnično zahtevno delo, večino testiranja v tej fazi opravi programer sam oz. nekdo, ki ima s programiranjem veliko izkušenj. Uporabniki so v tem delu vključeni v toliko, da preverijo izgled uporabniškega vmesnika. Testiranje programov se mora izvesti, preden se celoten sistem preda v testiranje testni skupini. Takšno testiranje je navadno poceni in zagotavlja učinkovite rezultate pri odkrivanju napak, hkrati pa se lahko izvaja kadarkoli med programiranjem.

Prva aktivnost, ki jo opravi programer po tem, ko napiše program, je preverjanje sintaktičnih napak. To so napake v zvezi s sintakso uporabljenega programskega jezika. Navadno je že prevajalnik orodja, v katerem je napisan program, v pomoč pri njihovi odpravi, saj pokaže napake v programu. Po odpravljenih sintaktičnih napakah je program pripravljen za izvajanje. Programer opravi ponoven pregled programske kode z namenom odpravljanja logičnih napak, kjer se preveri skladnost napisanega programa s programskimi specifikacijami. Za preverjanje napisanih programov pa v fazi izgradnje sistema izvajamo še tri vrste testiranja (Shelly at al., 2001):

- testiranje enote proizvoda,
- integralno testiranje,
- sistemsko testiranje.

Slika 12 prikazuje, kako te vrste testiranja izvajamo na sistemu, ki ga razvijamo.

Slika 12: Testiraje v fazi izgradnje sistema



Vir: Shelly, 2001

Testiranje enote (angl. unit testing) zajema testiranje posameznega programa, običajno je takšno testiranje zahtevnejše v primeru algoritmov za izračunavanje ali ažuriranje podatkov, manj zahtevno pa je v primerih vgrajenih sprememb algoritma za prikazovanje podatkov. Cilj testiranja enote proizvoda je identificirati in odpraviti napake, ki bi lahko povzročile nenormalno obnašanje programa in zgrajenega sistema. Izvajamo ga na vsakem programu, ne glede na to, da se povezuje z drugimi programi. Pri tem programer simulira vhod podatkov v program, v programu pa pripravimo sporočilo ali izpis izhodnih podatkov, ki jih program vrne. Testiranje enote je zaključeno, ko le-ta deluje pravilno. Takrat lahko preidemo na integralno testiranje.

Integralno testiranje (angl. integration testing, link testing) zajema testiranje programov, ki so med seboj povezani. Delovati morajo usklajeno in poskrbeti, da morebitna napaka v npr. tretjem programu povzroči ustrezne korake v prvih dveh programih, da se vzpostavi stanje, kot je bilo pred izvajanjem programa. Ob ugotovitvi napake je treba ponoviti test sklopa podatkov, če pa se ugotovi napaka na že testiranem elementu (v sklopu testiranja enote), se mora za popravljen element najprej izvesti testiranje enote in nato ponovno integralno testiranje.

Testiranje enote proizvoda in integralno testiranje opravi programer sam oz. skupaj s preizkuševalcem, ki dobro pozna programsko orodje, njegove možnosti in omejitve ter ima na splošno precej programerskih izkušenj. Načeloma je pregled kode, ki jo opravita programer in preizkuševalec, boljša izbira. Programer sam namreč zelo težko opazi svoje napake, saj napisan program zelo dobro pozna, druga oseba pa je pri odkrivanju pomanjkljivosti lahko zelo uspešna.

Sistemsko testiranje (angl. system testing) zajema testiranje celotnega informacijskega sistema, to je testiranje vseh programov skupaj. Glavni cilj sistemskega testiranja je ponovno preverjanje pravilnosti delovanja vseh programov in potrditev, da so vse komponente sistema pravilno povezane in omogočajo enostavno delo uporabnikov. Sistemski test je pomemben še posebno takrat, ko je za programiranje enega sistema zadolženo več programerjev, ki gradijo vsak svoj del programa. Sistemski test opravi preizkuševalec, ki je pri razvoju sistema prisoten od samega začetka in pozna vse zahteve glede izgradnje informacijskega sistema. S sistemskim testom tudi potrdimo, da bo informacijski sistem nemoteno deloval s količino podatkov, ki je bila za delovanje sistema predvidena.

Uspešno zaključen sistemski test zaključuje tudi korak testiranja v fazi izgradnje sistema. Vsako od zgoraj navedenih testiranj zahteva tudi ustrezno dokumentiranje odkritih in odpravljenih napak ter ob koncu testiranja izdano poročilo, kjer je jasno vidno, da je sistemsko testiranje uspešno zaključeno.

5.4 Testiranje delovanja sistema v testnem okolju

Vsa testiranja sistema do tega trenutka se opravijo v razvojnem okolju oz. v okolju, ki ga za razvijanje uporabljajo razvijalci sistema. V tej fazi pa se sistem prenese v testno okolje. Testiranje sistema v testnem okolju tako ponuja priložnost preveriti in oceniti delovanje sistema v natančno takem okolju, v kakršnem bo deloval ob uvedbi in uporabi. Vse to seveda velja ob predpostavki, da sta testno in produkcijsko okolje identična. Dosedanja uspešno izvedena testiranja, še posebno sistemsko testiranje, ki smo ga izvajali v prejšnji fazi, zagotavljajo, da bo sistem tudi tu deloval korektno. Kljub vsemu pa so v njem lahko še vedno napake, ki so posledica nerazumevanja ali napačnega razumevanja specifikacij in programiranja. Odvisno od tega, kako resna je napaka, se odločimo za spremembo oz. dopolnitve v sistemu pred predajo v uporabo. Če gre za obsežnejše napake, moramo razmisliti o prekinitvi testiranja in vrnitvi v prejšnje fazo, saj glede na stroške, ki jih testiranje povzroča, ni smiselno nadaljevati s testiranjem tako pomanjkljivega sistema.

Testna skupina ima v tej fazi tri naloge:

- priprava testnih primerov za testiranje,
- izvajanje testiranja,
- priprava poročila.

Priprava testnih primerov je ena pomembnejših nalog, saj se mora testna skupina od vsega začetka zavedati, da ne bo mogla preveriti čisto vseh možnosti, ki jih sistema ponuja. Prav

zato je priprava dobrih in smiselnih testnih primerov odločilen kriterij vsakega uspešnega testiranja. Testna skupina se pri pripravi testnih podatkov naslanja na testni plan oz. pripravlja testne primere v skladu s priporočili, ki so v njem zapisani. Njena odgovornost je, da za vsako vrsto testiranja izbere pravilno tehniko testiranja, v primeru avtomatskega testiranja pa tudi testno orodje. Pri pripravi testnih primerov se vedno držimo preprostega pravila, da se najprej pripravijo primeri za testiranje postopkov, ki se izvajajo na pravilen način, nato pa še za postopke, ki jih namerno izvajamo napačno oz. z neveljavnimi podatki.

Veliko je načinov in metod, kako preveriti delovanje sistema, izbira je odvisna od samega sistema. Kako bo potekalo testiranje, mora biti zapisano v testnem planu, saj bi bilo testiranje brez plana neekonomično in neučinkovito. Testiranje zahteva skrbno zamišljen proces izvedbe testiranja. V skladu s priporočili, zapisanimi v testnem planu, testna skupina izvaja naslednje vrste testiranja: testiranje avtorizacije uporabnika, testiranje funkcionalnosti celotnega sistema, regresijsko testiranje, stresno testiranje, varnostno testiranje, paralelno testiranje, testiranje sledljivosti nastalih sprememb v sistemu, testiranje delovanja sistema na rezervni lokaciji, performančno testiranje,…

Sama izvedba testiranja zajema vnašanje pripravljenih testnih primerov in ob njihovem vnosu spremljanje obnašanja sistema oz. primerjanje dobljenih rezultatov s pričakovanimi rezultati. Vsako odstopanje od njih pomeni, da smo naleteli na napako. Prvi korak k uspešnemu razreševanju in odpravljanju napak je ustrezno dokumentiranje napak in odstopanj od pravilnosti delovanja sistema (Perry, 2000, str. 407). Prav zato moramo poskrbeti, da so vse ugotovljene napake na razumljiv in pregleden način dokumentirane. Vsak zapis o ugotovljeni napaki mora vsebovati podatke o tem, kaj testiramo, kakšen rezultat smo pri testiranju dobili, kakšnega smo pričakovali, obrazložiti moramo, zakaj je razlika med dejanskimi in pričakovanimi rezultati pomembna za obravnavo, če so poznani razlogi za odstopanje od pričakovanega rezultata, jih tudi navedemo. Vse odkrite napake zahtevajo dodatno analizo, ki jo opravi razvojna skupina. V odvisnosti od tega, kakšna napaka je bila odkrita, se določi nadaljnji postopek pri odpravljanju napak. Odkrita napaka lahko zahteva vrnitev v katero od začetnih faz razvoja in nato spremembo v vseh fazah, ki si sledijo, lahko zahteva samo aktivnosti v fazi izgradnje sistema – torej gre za t. i. napako razvijalca, ali pa je napaka takšna, da se njeno odpravljanje prestavi na poznejši čas, ko je sistem prenesen in že deluje v produkcijskem okolju. Seveda takšna napaka ne sme bistveno vplivati na uspešnost delovanja sistema oz. mora biti manjša napaka, ki uporabnikov bistveno ne ovira pri delu. V primeru da napaka zahteva vrnitev v prejšnje faze razvoja, se celoten postopek testiranja v vsaki fazi ponovi.

Testiranje sistema v testnem okolju je torej zadnja priložnost za odpravo napak pred uvedbo sistema v produkcijsko okolje. Pri razvoju informacijskih sistemov se pogosto zgodi, da prav za testiranje sistema v testnem okolju zmanjkuje časa oz. je omejen na zelo kratek časovni termin zaradi zamud v prejšnjih fazah. Tako se lahko zgodi, da bodo preizkuševalci zaradi pritiska spregledali kakšno resno napako, ki bo povzročala težave v produkcijskem okolju. Neupravičeno bi bilo pričakovati, da bomo s tem testiranjem

odpravili vse napake, še posebno če v prejšnjih fazah nismo opravili predpisanega testiranja. Izkušnje kažejo, da učinkovitost pri odpravljanju napak v tej fazi težko preseže 80%. Prav zato se moramo truditi, da v to fazo »prinesemo« čimmanj napak iz prejšnjih faz – tako bomo manj napak prenesli tudi v produkcijsko okolje (Perry, 2000, str. 438). Testiranje v tej fazi je tudi zadnja priložnost, da uporabniki zagotovijo pravilno delovanje sistema in prav zato je pomembno, da so vključeni v testiranje. Informatiki so imeli priložnost opraviti testiranje v fazi izgradnje sistema, v tej fazi pa naj imajo odločilno vlogo uporabniki. Če bi testiranje v obeh fazah opravili isti preizkuševalci, pravzaprav delitev v dve fazi sploh ne bi imela pravega smisla. V nasprotnem primeru pa lahko trdimo, da je delo preizkuševalcev v obeh fazah komplementarno.

5.5 Uporabniško testiranje

Sprejemno testiranje, pogosto imenovano tudi uporabniško testiranje, predstavlja testiranje ustreznosti programske podpore, ki jo izvaja končni uporabnik (kupec) z namenom preveriti, ali deluje v skladu z definiranimi zahtevami. Sprejemno testiranje se ne sme začeti, preden so zaključena vsa testiranja v prejšnjih fazah, ki jih izvaja testna skupina. Biti morajo ustrezno dokumentirana in morajo izkazovati dokaze o uspešnosti testiranja. Sprejemno testiranje je pravzaprav zadnja priložnost končnih uporabnikov/kupcev, da pregledajo in zahtevajo morebitne popravke v aplikaciji. Potrditev v fazi sprejemnega testiranja pomeni verifikacijo, da sta programska oprema in pripadajoča dokumentacija primerni in skladni ter da programska oprema podpira vse zahteve končnih uporabnikov. Na ustreznost programske opreme vplivajo štiri komponente (Perry, 2000, str. 464):

- **podatki**: zanesljivost, pravočasnost, doslednost in uporabnost podatkov,
- **ljudje**: spretnosti, izobraževanje, zmožnost in želja po pravilni uporabi programske opreme,
- **struktura**: primeren razvoj sistema, ki omogoča optimalno uporabo tehnologije in tako uresničuje zahteve uporabnikov,
- **pravila**: postopki, ki jih moramo upoštevati pri procesiranju podatkov.

Če je katera od zgoraj navedenih komponent zastopana v manjši meri, je uspešnost in ustreznost razvitega sistema precej zmanjšana. S testiranjem preverimo in zagotovimo, da so vse štiri komponente primerno zastopane in skupaj omogočajo najboljšo možno rešitev poslovnega problema.

Formalni sprejemni test se mora izvesti ob koncu razvojnega procesa, pomembno pa je, da se sprejemni testi oz. potrditve uporabnikov odvijajo skozi vse faze razvoja. Le na ta način se lahko izognemo nepotrebni porabi časa za uvajanje sprememb v sistem, za katere bi lahko ob koncu ugotovili, da so za uporabnike nesprejemljivi. Tako mora biti sprejemno testiranje vključeno v projektni plan, v nasprotnem primeru se možnost, da bo končni izdelek za uporabnika nesprejemljiv, močno poveča.

Naloga preizkuševalcev v fazi sprejemnega testiranja je:

- priprava sprejemnih kriterijev,
- priprava plana uporabniškega testiranja,
- priprava testnih primerov,
- izvedba testiranja in
- priprava poročila o uporabniškem testiranju.

Uspešno sprejemno testiranje zahteva skrbno pripravo sprejemnih kriterijev, po katerih bodo uporabniki preverjali primernost razvitega sistema. Vsi kriteriji, po katerih bodo uporabniki izvedli ocenjevanje, morajo biti postavljeni tako, da so merljivi, lahko pa jih razdelimo v naslednje skupine (Perry, 2000, str. 468):

- funkcionalni kriteriji, ki se nanašajo na poslovna pravila, ki jih mora sistem podpirati,
- performančni kriteriji, ki so povezani z delovanjem sistema glede na časovne omejitve,
- splošni kriteriji, ki specificirajo zanesljivost, uporabnost, možnost sprememb.

Plan uporabniškega testiranja je lahko del testnega plana ali pa samostojen dokument. Vsebuje enake elemente kot testni plan, ki smo ga pripravili za testiranje v ostalih fazah razvoja, natančneje opredelimo le še kriterije, po katerih bodo bodoči uporabniki ocenjevali sistem. Vsebuje naj tudi terminski plan aktivnosti, odgovornosti preizkuševalcev ter priporočljivo uporabo tehnik testiranja in testnih orodij.

Ob zaključku testiranja morajo končni uporabniki pripraviti poročilo oz. sprejeti odločitev o neustreznosti oz. ustreznosti razvitega sistema. Taka odločitev jasno izraža, da so bile opravljene vse zahtevane spremembe in da se lahko nadaljuje zaključni del razvoja sistema – to je prenos sistema v produkcijsko okolje.

5.6 Testiranje ob namestitvi sistema

Testiranje ob namestitvi sistema vključuje preizkus namestitve sistema v produkcijsko okolje in preverjanje delovanja osnovnih funkcij sistema. Takšno testiranje izvajamo v primeru prenosa novega sistema v produkcijsko okolje in tudi v primeru prenosa nove verzije že delujočega sistema v produkcijsko okolje. Sam proces testiranja je v obeh primerih podoben, v primeru prenosa nove verzije sistema pa zahteva še nekaj več pazljivosti. Izvedba testiranja v tej fazi je odgovornost informatikov, uporabniki morajo sodelovati pri pripravi podatkov in pri preverjanju delovanja funkcionalnosti sistema. Glavni cilj testiranja je torej preveriti, ali so:

- v produkcijsko okolje preneseni vsi potrebni programi, moduli in podatkovni objekti,
- pripravljeni vsi potrebni podatki,
- pripravljena in ustrezno napisana navodila za namestitev.

Naloge preizkuševalcev v fazi namestitve sistema so:

- namestitev sistema,
- preverjanje delovanja osnovnih funkcij in spremljanje delovanja sistema v produkcijskem okolju,
- priprava poročila o testiranju.

Način, kako poteka namestitev sistema v produkcijsko okolje, je odvisen od tipa aplikacije. V nekaterih primerih se namestitev opravi na delovno postajo uporabnika, v drugih primerih gre za namestitev sistema na strežnik. V vsakem primeru izvedemo namestitev sistema po vnaprej pripravljenih navodilih, ki jih pripravi razvijalec sistema. Poskus namestitve novega sistema je navadno neproblematičen, razen če vemo, da bo namestitev novega sistema vplivala na delovanje ostalih sistemov. Bolj previdni pa moramo biti pri namestitvi nove verzije sistema, saj le-ta prekrije trenutno veljavno verzijo. V takšnem primeru moramo vedno imeti pripravljen plan, kako bomo ob morebitni neuspešni namestitvi sistema čimprej lahko vzpostavili prejšnje stanje.

Po uspešno izvedeni namestitvi preizkuševalci začnejo preverjati delovanje osnovnih funkcionalnosti sistema v produkcijskem okolju. Vsako takšno preverjanje moramo natančno predvideti, še posebno, če nov sistem nadomešča starega in smo naredili migracijo obstoječih produkcijskih podatkov iz starega v novo okolje. Preizkuševalec se mora zavedati, da bo testiranje osnovnih funkcij opravljal na produkcijskih podatkih, zato takšno testiranje navadno zajema proženje pregledov, izpisov,… oz. izbiro takšnih funkcij, pri katerih podatkov ni mogoče spreminjati. Pred namestitvijo nove verzije se moramo dogovoriti, kaj bomo storili s staro verzijo. Iz povsem varnostnih razlogov, pa tudi zaradi potrebe po ohranjanju arhiva, navadno starih verzij ne brišemo, pač pa jih shranimo na dogovorjeno mesto, kjer hranimo arhive vseh verzij, ki so bile predane v uporabo.

Morebitne napake oz. odstopanje od napisanih navodil za namestitev, izvajalec testiranja zapiše in sporoči osebi, ki je odgovorna za izvedbo dopolnitev oz. spremembo.

Ob zaključku testiranja mora biti izdana verifikacija, da vse procedure delujejo pravilno, da so postopki namestitve sistema zapisani pravilno, se pravilno izvajajo in da osnovne funkcije, katerih delovanje je bilo preverjeno, delujejo v skladu s pričakovanji. Z zaključkom testiranja v tej fazi in pripravo ustreznih poročil se lahko začnejo priprave na dejansko namestitev in uporabo aplikacije. Ko je sistem predan v produkcijsko okolje in ga uporabniki začnejo uporabljati, je smiselno njegovo delovanje še nekaj časa spremljati.

5.7 Testiranje sprememb sistema

Spremembe v sistemu navadno upravljajo tako uporabniki kot tudi skupina, ki skrbi za vzdrževanje sistema. Testiranje sprememb, ki nastanejo v času vzdrževanja sistema, je prav tako pomembno kot testiranje v času razvoja sistema. Napake, ki smo jih odkrili v procesu testiranja ali pa tudi že ob delovanju sistema v produkcijskem okolju, je treba odpraviti. To pa zahteva ponovno testiranje, saj obstaja bojazen, da bi s spremembami vnesli nove napake ali pa bi bila tudi sprememba implementirana na napačen način. Možno je tudi, da je sprememba tako velika in vpliva na velik del delovanja aplikacije, da

bo treba večino do sedaj opravljenega testiranja ponoviti. Tudi najmanjše spremembe zahtevajo veliko testiranja in nič neobičajnega ni, da za to porabimo precej časa. Spremembe sistema se pojavljajo tako v razvoju kot pri vzdrževanju sistema. Glede na to ločimo:

- testiranje sprememb v produkcijskem okolju delujočega sistema – nova verzija delujočega sistema,
- testiranje sprememb, ki nastanejo med razvojem sistema – nova verzija razvijajočega sistema.

Izvedba testiranja, ki jo opisujem v nadaljevanju, se lahko uporabi v obeh primerih.

Cilj testiranja sprememb je zagotoviti, da so spremembe pravilno implementirane in pravilno delujejo. Spremembe zajemajo tako morebitne ročne postopke kot vse avtomatizirane postopke in navodila za delo. Izvajalci testiranja morajo skupaj s spremembami prejeti tudi novo verzijo dokumentacije in rezultate testiranja v prejšnjih fazah, poskrbeti pa morajo, da se preveri delovanje nove verzije sistema skupaj z vso pripadajočo dokumentacijo.

V koraku testiranja sprememb morajo izvajalci v skladu z opravljenimi spremembami testiranja ustrezno spremeniti ali dopolniti:

- testni plan, ki je praviloma krajši od tistega za testiranje nove aplikacije,
- prilagoditi testne primere. V večini primerov za testiranje zadoščajo že obstoječi testni primeri, v nekaterih primerih, če je npr. sprememba tako velika, da podpira povsem novo funkcionalnost, pa je treba pripraviti nove,
- testno in sistemsko dokumentacijo.

Testiranje nove aplikacije navadno traja več tednov, testiranje sistema, ki je že operativen, pa je moč zaključiti v enem dnevu, v nekaterih primerih celo v nekaj urah. Način testiranja je odvisen od implementiranih sprememb. Če na primer spremenimo izpis poročila, potem res ni nikakršne potrebe, da bi izvajali ponovno varnostno ali npr. obnovitveno testiranje in nasprotno, če spreminjamo procedure za obdelavo podatkov ali se npr. kreirajo nove datoteke, pa morda samo funkcionalno testiranje tudi ne bo zadoščalo. Prav zaradi tega moramo dopolniti/pripraviti testni plan, v katerem glede na spremembe opredelimo, kaj in kako bomo testirali in kakšen je pričakovani rezultat. Seveda so prav tako pomembne časovne omejitve testiranja in odgovornosti. Za testiranje sprememb, ki nastanejo med razvojem sistema, je še posebno pomembno regresijsko testiranje, kar pomeni, da spremenjeno verzijo aplikacije ponovno testiramo na takšnih testnih primerih, kot smo jo testirali pred spremembo, preveriti pa moramo tudi, da sprememba ni povzročila novih in nepredvidenih napak.

5.8 Testiranje primernosti sistemske dokumentacije

Testiranje primernosti sistemske dokumentacije sicer navajam ob koncu procesa testiranja, vendar se pojavlja prav v vseh fazah razvojnega in testnega cikla. Izvaja se periodično

skozi vse korake testiranja: testiranje dokumentacije, ki je pripravljena v fazi zbiranja zahtev, testiranje dokumentacije v fazi načrtovanja, testiranje programske dokumentacije,…

Priprava razvojne dokumentacije zahteva 10 – 25% časa med razvojem in vzdrževanjem sistema. Ob zaključku mora biti pripravljena popolna in veljavna dokumentacija. Tako kot pri razvoju samem pa se tudi pri pripravi sistemske dokumentacije pojavljajo določene napake. Razlika je v tem, da napake v programu vodijo k napačnim rezultatom, napake v dokumentaciji pa lahko povzročijo napačno spremembo, ki jo naknadno opravimo v sistemu. Glavni cilj testiranja sistemske dokumentacije je zagotoviti, da pripravljena dokumentacija natančno ustreza sistemu, ki ga razvijamo.

Skozi razvojne faze informacijskega sistema nastajajo različni dokumenti: zagonska dokumentacija, dokument z uporabniškimi zahtevami, programske specifikacije, uporabniška navodila, navodila za namestitev, programska dokumentacija,… Kakšno dokumentacijo bomo pripravljali v posamezni fazi, je navadno odvisno od standardov, ki veljajo v določeni organizaciji, zagotoviti pa moramo, da je le-ta ves čas ažurna in da odraža dejansko stanje sistema. Testiranje primernosti sistemske dokumentacije zahteva, da preverimo njeno popolnost in veljavnost. V obeh primerih je delo precej zahtevno, saj je težko preveriti, ali vsak delček dokumentacije odraža dejansko stanje sistema, zato ga mora opraviti izkušen preizkuševalec, za katerega pa je najbolje, da ni član projektnega teama.

Preverjanje dokumentacije se lahko izvaja na različne načine:

- S simulacijo izvedbe spremembe na podlagi napisane dokumentacije. Po opravljeni spremembi mora nekdo iz projektnega teama oceniti, ali je bila sprememba opravljena pravilno. Takšna vrsta testiranja pove, ali je pripravljena dokumentacija razumljiva in ali na njeni osnovi neodvisna oseba lahko učinkovito opravi spremembo v sistemu.
- S primerjavo napisane dokumentacije in programske kode. Preizkuševalec v tem primeru izbere samo posamezne naključno izbrane dele dokumentacije, v pogovoru s prejemniki oz. porabniki dokumentacije pa nato skupaj preverijo in potrdijo, da pripravljena dokumentacija zares ustreza sistemu.

V celotnem ciklu razvoja sistema se pripravi veliko raznovrstne dokumentacije. Preverjanje primernosti sistemske dokumentacije je navadno kar naporno delo, saj se ni prav lahko prebiti čez množico napisane dokumentacije. Zato je smiselno razmisliti o postavitvi standardov, ki natančno določajo, kakšna dokumentacija mora biti v posamezni fazi pripravljena in kaj mora biti v njej zapisano (navadno se definirajo že kar poglavja, ki jih mora dokumentacija vsebovati). Za vsako fazo se predvidi tudi priprava dokumentacije, še posebno med vzdrževanja sistema, ko je dokumentacijo treba prilagajati in dopolnjevati v skladu s spremembami. Uvedba ISO standarda je lahko eden od korakov k zagotovitvi kakovosti pri pripravi dokumentacije, seveda pa se na ta način še vedno ne moremo izogniti preverjanju primernosti pripravljene dokumentacije.

5.9 Ocenitev učinka testiranja

»Je vredno vložiti toliko denarja v proces testiranja?« je razumljivo vprašanje, ki si ga zastavlja vodstvo, saj je navadno v proces testiranja vključenih veliko virov. Na takšno vprašanje pa ne moremo odgovoriti, če ne izmerimo učinka testiranja. Merjenje učinka testiranja ima dva namena:

- ocenjuje, kakšen je učinek posameznika,
- rezultati ocenjevanja se uporabijo za prilagoditev, spremembe in izboljšave izvedbe testnega procesa, ki ga bomo izvajali v prihodnje.

Oba namena sta med seboj tesno povezana in se zanju lahko uporablja iste ocenjevalne kriterije.

Cilj ocenjevanja učinka testiranja je torej učiti se iz svojih napak, kar pomeni, da moramo identificirati napake in težave, na katere smo naleteli pri testiranju, z namenom, da bi testiranja v prihodnje opravljali bolj pravilno. Tudi testiranje samo je proces, pri katerem se naredi veliko napak in izkušnje v mnogih podjetjih kažejo, da preizkuševalci naredijo več napak kot razvijalci (Perry, 2000, str. 599). Prav zato moramo testni proces stalno izboljševati. Brez izboljšav bodo preizkuševalci ponavljali iste napake in bo testiranje vedno znova neučinkovito.

Za izvršitev ocenjevanja učinka testiranja moramo določiti skupino, ki bo odgovorna za zbiranje podatkov in ocenjevanje izvedbe testiranja. Na začetku mora skupina najprej ugotoviti, kdo vse je bil vpleten v testiranje in v kakšnem obsegu, katera področja so bila pokrita, koliko virov je bilo porabljenih v procesu testiranja in koliko časa je bilo porabljeno za odpravljanje napak. Nato pa mora postaviti še cilje ocenitve učinka testiranja, ki morajo biti jasno definirani, sicer ne morejo biti učinkoviti. Naloga testne skupine je, da:

- definira vse slabosti, na katere so preizkuševalci naleteli v procesu testiranja,
- ugotovi potrebo po novih testnih orodjih,
- pretehta vse dobre in slabe testne postopke in primere,
- določi, katere karakteristike so tiste, ki naredijo testiranje bolj ekonomično ter
- ugotovi, kje je bila metodologija pri odkrivanju napak neučinkovita.

Za ocenjevanje učinkov testiranja je na voljo več različnih pristopov (Perry, 2000, str. 605):

- Razsodba, presoja – testni proces se ocenjuje na podlagi mnenja posameznika, gre za poljubno oceno posameznika, ki jo je težko zagovarjati, vendar pa je tak način lahko zelo učinkovit, če ocenjevalec zna realno gledati na rezultate testiranja in podati oceno, ki je realna in skladna s testnimi rezultati.
- Usklajenost z metodologijo – če so postavljeni testni standardi in navodila, lahko naredimo oceno na podlagi usklajenosti s standardi.

- Napake po testiranju – učinkovitost testiranja lahko merimo tudi s številom napak, ki so se pojavile po zaključku testiranja. Če je teh težav malo, potem lahko zaključimo, da je bilo testiranje dobro izvedeno, v nasprotnem primeru pa slabo.
- Odziv uporabnikov – učinek testiranja lahko izmerimo tudi z zadovoljstvom uporabnikov z razvitim informacijskim sistemom.
- Merske metode – z njimi ugotavljamo korelacijo med dvema testnima spremenljivkama. Rezultate moramo primerjati s postavljenimi standardi in normami, sicer dobljeni rezultat nima pravega pomena. Tak način je zelo priporočljiv za uporabo, saj ga je, ko je enkrat zgrajen, enostavno uporabljati. Glavna prednost pa je v tem, da je proces ocenjevanja jasno definiran, poznan je ljudem, ki jih ocenjujejo in je dovolj specifičen, da se zlahka določi, kje bo treba testni proces izboljšati: na področju učinkovitosti, zmožnosti, ali/in na ekonomskem področju.

Za kateregakoli od uporabljenih pristopov bomo morali zbrati potrebne informacije o pogostosti, velikosti in tipu sprememb v sistemu, stroških razvoja sistema, stroških testiranja, številu odkritih napak v posamezni fazi razvoja sistema, stroških testiranja posamezne funkcije.

Ocenjevanje učinka testiranja je v procesu testiranja pogosto tisti korak, ki ga izpustimo. Sicer nima direktnega učinka na sistem, ki smo ga pravkar preizkušali, vendar je zelo pomemben, saj prinaša večjo učinkovitost pri testiranju bodočih sistemov. Prav preizkuševalci bi morali razumeti, da so pri testiranju in odkrivanju napak drugih tudi oni tisti, ki pri svojem delu delajo napake. Tu ne gre za direktno ocenjevanje njihovega dela z namenom, da bi določili boljše in slabše preizkuševalce, pač pa z namenom, da se vsak preizkuševalec nekaj nauči iz lastnih napak in to znanje uporabi v prihodnosti za večjo učinkovitost pri testiranju informacijskih sistemov.

Z oceno učinka testiranja smo proces testiranja pripeljali do konca. Zadnji korak v njem je tako že osnova za uspešen začetek dela na razvoju novega informacijskega sistema. Proces izvedbe testiranja moramo seveda sproti prilagoditi vsakemu sistemu, ki ga razvijamo. Pomembno pa je, da se zavedamo, da je testiranje pomemben del razvoja sistema in da si moramo ves čas prizadevati, da so preizkuševalci vključeni v razvoj sistema že od samega začetka. Samo na tak način lahko zagotovimo najnižje stroške pri razvoju sistema, kar pa je vsekakor cilj vsakega podjetja.

6 STROŠKI TESTIRANJA

Izvedba testiranja je navadno zamudna in utrujajoča, pogosto se tudi zgodi, da prav testiranje, seveda če je pristop k njemu resen, zajema okoli 50% vseh stroškov razvoja informacijskega sistema. Odkrivanje in odpravljanje napak se v vsaki fazi izvaja v več ciklih: napako odkrijemo in odpravimo, ponovno testiramo, odkrijemo nove napake ter jih posredujemo v odpravljanje in postopek spet ponovimo, dokler s končnim izdelkom, ne

glede na to, v kateri fazi razvoja se nahajamo, nismo povsem zadovoljni. Z vidika stroškov je testiranje upravičeno, dokler njegovi stroški ne presegajo tistih, ki bi jih prinesle neodkrite napake in posledično nedelovanje ali nepravilno delovanje sistema v produkcijskem okolju.

Testiranje informacijskega sistema lahko delimo na:

- testiranje v času razvoja, pred implementacijo sistema. Cilj testiranja v času razvoja je potrditi, da sistem deluje v skladu s pripravljenimi specifikacijami in brez napak. Stroške odpravljanja napak, ki nastanejo v tem času, lahko razdelimo na stroške (Perry, 2000, str. 161 - 162):
  1. vgrajevanja napak v sistem,
  2. iskanja napak,
  3. odpravljanje napak,
  4. testiranja (potrjevanja), da so napake odpravljene,
- testiranje po uvedbi sistema v produkcijsko okolje. Na stroške, ki jih povzročajo neodkrite napake v sistemu med vzdrževanjem, pa vplivajo (Perry, 2000, str. 162):
  1. vgrajevanje napak v sistem,
  2. odkrivanje napak,
  3. poročanje o napakah razvijalcem sistema,
  4. reševanje težav, ki jih je povzročila napaka,
  5. delovanje sistema, dokler napaka ni odpravljena,
  6. popravljanje napak,
  7. testiranje (potrjevanje), da napake v sistemu ni več,
  8. prenos popravljenega/ih programa/ov v produkcijsko okolje.

Stroški testiranja torej vključujejo tako stroške testiranja odkritih napak kot tudi stroške, ki jih povzročajo neodkrite napake. Prav zato jih je težko določiti: če že lahko določimo stroške testiranja, pa je praktično nemogoče določiti stroške, ki so povezani z neodkritimi napakami v sistemu. Bolj praktičen način za oceno stroškov pa opisuje Perry (2000, str. 163), ki pravi, da je treba popisati vse napake, ki so bile odkrite v različnih razvojnih fazah. Pomembno, je da vemo, v kateri fazi je bila določena napaka odkrita, saj se stroški odkrivanja napak skozi razvojne faze zvišujejo. Napake, ki so odkrite v fazi zbiranja in popisovanja zahtev, prinašajo najnižje stroške, ki so povezani s popravljanjem napak. Za faktor 10 so višji stroški napak, ki jih odkrijemo šele v fazi sistemskega testiranja in za faktor 1000 so višji stroški odpravljanja napak, ki jih odkrijemo v že delujočem sistemu v produkcijskem okolju.

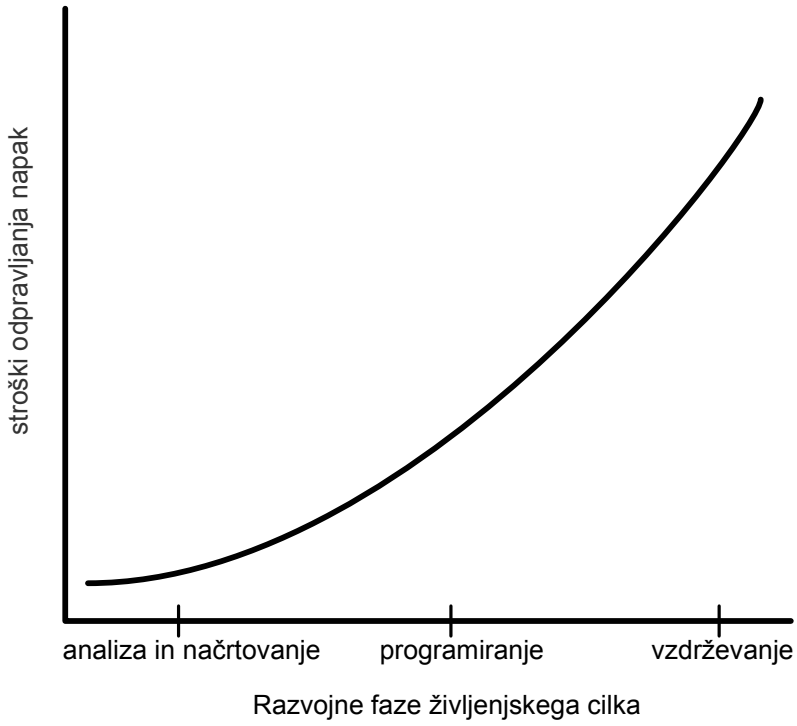
Martin & McClure (1983) sta v svojem delu razporedila deleže relativnih stroškov, ki nastajajo v procesu razvoja informacijskega sistema po posameznih fazah:

- analiza zahtev 3%
- priprava specifikacij 3%

- načrtovanje 5%
- programiranje 7%
- testiranje 15%
- delovanje in vzdrževanje 67%

Glede na zgoraj navedeno je največji strošek vzdrževanje že razvitega sistema, testiranje, ki se pojavlja v vsaki od navedenih faz razvoja informacijskega sistema, pa je druga najdražja aktivnost. Torej stroški iskanja in popravljanja napak v razvojnem procesu eksponentno naraščajo, kar predstavlja tudi slika 13.

Slika 13: Naraščanje stroškov odpravljanja napak v različnih razvojnih fazah



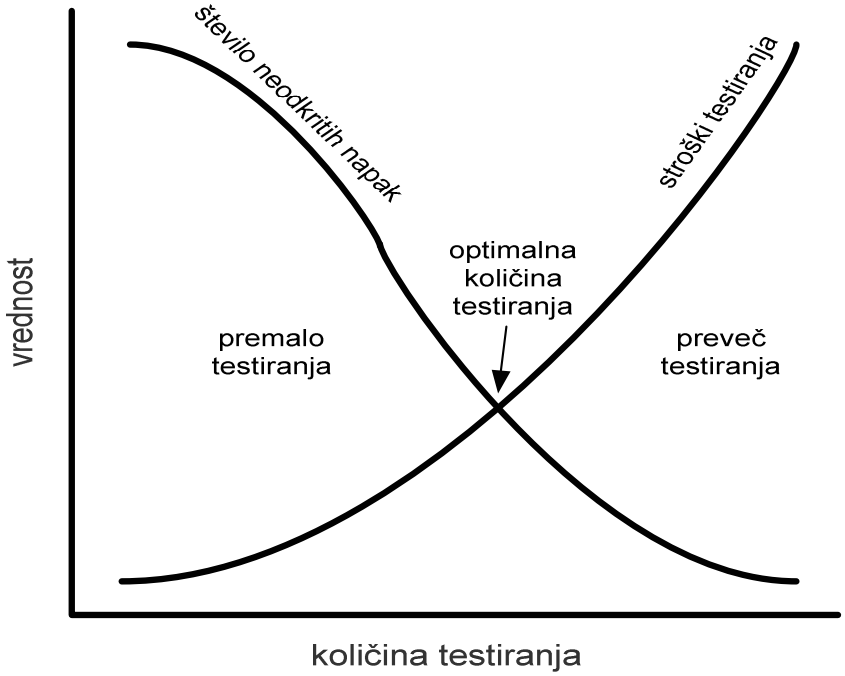
Vir: Kanner, 1999

Spreminjanje zahtev v fazi zbiranja zahtev prinaša minimalne stroške, sprememba zahtev v fazi izdelave (programiranja) sistema pa prinaša že precej velike stroške, saj se je treba vrniti v začetne faze razvoja, dopolniti ali spremeniti obstoječe izdelke, opraviti potrebna testiranja in prilagoditi že napisano programsko kodo. Popravljanje napak je precej cenejše, če svoje napake odkrijejo programerji sami: nikomur namreč ni treba pojasnjevati napak, ni jih treba beležiti v sistem za beleženje napak, če je le-ta vzpostavljen, prav tako pa napake oz. nedelovanje ne zmotijo uporabnika sistema.

Seveda testiranja ne moremo in ne smemo izvajati v nedogled, saj to nima pravega smisla. Na neki točki je testiranje vendarle treba ustaviti in predati sistem v uporabo. Pravilno bi bilo končati s testiranjem takrat, ko zanesljivost sistema doseže takšno stopnjo, kot je zapisano v zahtevah za izgradnjo informacijskega sistema ali ko koristi, ki jih prinaša

testiraje še lahko upravičimo s stroški, ki pri testiranju nastanejo (Pan, 1999). Krivulja na sliki 14 predstavlja razmerje med količino in stroški testiranja. Medtem, ko stroški testiranja naraščajo, se zmanjšuje število neodkritih napak, stična toča obeh krivulj pa kaže na optimalno količino testiranja. Ko dosežemo stično točko obeh krivulj, smo dosegli t. i. optimalno količino testiranja. Testiranje od tega trenutka dalje pa že kaže na prekoračitev meje. Leva stran v grafu predstavlja premalo testiranja: stroški testiranja so nižji, kot bodo stroški, ki jih povzročajo neodkrite napake. Desna stran pa predstavlja preveliko količino testiranja, kar pomeni, da so stroški testiranja večji, kot pa bodo stroški, ki bi jih povzročile neodkrite napake.

Slika 14: Razmerje med količino in stroški testiranja



Vir: Perry, 2000, str. 38

6.1 Stroški zagotavljanja kakovosti

Na proces testiranja lahko gledamo kot na vrsto naložbe (Black, 2000). Nekateri vidijo testiranje kot neko nujno zlo, ki se pojavi ob koncu vsakega projekta in seveda razmišljajo o tem, da je testiranje predrago, da traja predolgo in ne prinaša nobenih prednosti. Tisti drugi, ki pa so bolj naklonjeni testiranju, pa se zavedajo, da je to naložba v kakovost.

Stroški kakovosti so stroški, ki so povezani z iskanjem, popravljanjem in preprečevanjem napak, ki nastajajo pri razvoju informacijskega sistema. Ti stroški so navadno visoki, vendar je večino njih možno občutno zmanjšati ali pa se jim skoraj povsem izogniti. Stroške kakovosti tvori vsota stroškov, ki jih porabimo za (Kaner, 1996):

- izobraževanje testne skupine, analizo zahtev, pregledovanje zahtev uporabnikov, izdelavo prototipa, oceno (pred morebitnim nakupom) orodij za testiranje,
- planiranje testnih aktivnosti, pripravo testnih podatkov in izvajanje testov,
- stroške iskanja in odpravljanja napak v programu, ponovno testiranje, direktne stroške, ki jih prinaša zamuda pri končanju sistema,
- stroške, ki jih povzročijo napake, ki jih v sistemu odkrijejo končni uporabniki (odkrivanje in odpravljanje napak, testiranje sistema, ponoven prenos v produkcijsko okolje,… Upoštevati moramo tudi škodo, ki je povzročena podjetju zaradi napake, izgubljenih poslov,…).

Dejavnikov, ki vplivajo na zviševanje stroškov, je veliko. Tako kot velja za večino stroškov, želimo zmanjšati tudi stroške kakovosti. Pri tem velja pravilo, da je lažje preprečevati kot zdraviti in testiranje je eden od načinov, kako lahko stroške znižamo. V nadaljevanju, v tabeli 2, navajam hipotetičen primer, ki na preprost način kaže, kako je z uvedbo testiranja možno znižati stroške in povečati donosnost naložbe (angl. return of investment (ROI)).

Za primer najprej navajam nekaj predpostavk: gre za informacijski sistem, ki vsako trimesečje zahteva novo verzijo programa. Predpostavimo, da je v sistem z vsako verzijo vgrajenih 1000 napak, ki morajo biti popravljene. Razvijalci sami jih odkrijejo 250, 350 jih odkrije skupina, ki izvaja testiranje, ostale napake pa odkrijejo uporabniki. Stroški odpravljanja napak razvijalcev predstavljajo 10 enot, strošek iskanja in odpravljanja napak, ki jih odkrijejo v testni skupini, je 100 enot in strošek napake, ki so jih odkrili uporabniki, znaša 1000 enot. Primer prikazuje, kakšni so stroški v primeru, da testiranja ne izvajamo oz. ga izvajamo ročno ali z orodji. V zadnji vrsti je v odstotkih prikazano tudi, kako se povrne naložba v testiranje.

Ključni dejavniki pri zmanjševanju stroškov razvoja informacijskega sistema je pravilno razmišljanje in podpora vodstvenih delavcev v podjetju. Prav ti so namreč tisti, ki morajo o testiranju razmišljati kot o naložbi. Pripravljeni se morajo biti soočiti s stroški kakovosti, ki nastajajo skozi celotni cikel razvoja sistema. Uvedba testiranja v vse faze razvoja informacijskega sistema je edini način, ki prinaša zmanjševanje stroškov in tako še povečuje kakovost samega proizvoda. Velja namreč pravilo, da so zgodnja testiranja cenejša in bolj učinkovita. Z izvajanjem procesa testiranja sproti preverjamo skladnost z zahtevami uporabnikov oz. skladnost z zahtevami, postavljenimi v prejšnjih fazah, hkrati pa poskušamo čim manj napak prenesti v zadnje faze razvoja oz. skupaj z delujočim sistemom v produkcijsko okolje, kjer so stroški odpravljanja napak še neprimerno višji.

Brez podpore vodstva je testiranje samo še ena v vrsti aktivnosti, ki jo je treba opraviti čimprej, da se zadosti nekim predpisanim pravilom.

Tabela 2: Donosnost naložbe

	Brez testiranja	Ročno testiranje	Testiranje z orodji
VIRI			
Izvajalci testiranja	0,00	60.000,00	60.000,00
Infrastruktura	0,00	10.000,00	10.000,00
Orodja za testiranje	0,00	0,00	12.500,00
Skupaj	**0,00**	**70.000,00**	**82.500,00**
Razvoj			
Odpravljanje napak	250,00	250,00	250,00
Stroški skupaj	**2.500,00**	**2.500,00**	**2.500,00**
Testiranje			
Odprava napak	0,00	350,00	500,00
Stroški skupaj	**0,00**	**35.000,00**	**50.000,00**
Uporabniki			
Odprava napak	750,00	400,00	250,00
Stroški skupaj	**750.000,00**	**400.000,00**	**250.000,00**
Stroški			
Stroški virov	**0,00**	70.000,00	82.500,00
Odpravljanje napak skupaj	**752.500,00**	437.500,00	302.500,00
SKUPAJ stroški kakovosti	**752.500,00**	507.500,00	385.000,00
ROI v %	**N/A**	**350**	**445**

Vir: Black, 2000

7 IZVEDBA TESTIRANJA RAČUNALNIŠKE REŠITVE NA PRIMERU SISTEMA ZA PODPORO BANČNEMU ZAVAROVALNIŠTVU

7.1 Bančno zavarovalništvo in bančno-zavarovalniške storitve

Kljub lepemu številu zavarovalnic v Sloveniji še vedno obstajajo možnosti za prodajo zavarovalniških storitev: 80% obsega zavarovalniškega posla pri nas predstavljajo neživljenjska, predvsem premoženjska zavarovanja, le 20% vseh sklenjenih zavarovanj pa je življenjskih zavarovanj. Prodaja zavarovanj se vrši pretežno preko mreže agentov in posrednikov, ki so bodisi zaposleni v zavarovalnicah, v posredniških podjetjih ali pa so samostojni podjetniki, ki za zavarovalnico po pogodbi posredujejo zavarovanja. Prav to pa je bil tudi eden od razlogov, da sta Nova Ljubljanska banka d. d. (NLB d. d.) in KBC v letu 2002 sprejeli odločitev, da skupaj preučita možnosti za uvedbo prodaje življenjskih zavarovanj v banki. Banka ima za prodajo tovrstnih storitev veliko možnosti prav zaradi velikega števila svojih komitentov, razvitih tržnih poti in nenazadnje zaradi ugleda močne, stabilne in zaupanja vredne institucije.

Pojem bančnega zavarovalništva razumemo kot prodajo bančnih in zavarovalniških storitev ter njihovih kombinacij, preko skupnih tržnih poti oz. preko iste baze komitentov. Prednost trženja bančno-zavarovalniških storitev je v tem, da so take storitve, ki so naprodaj v poslovalnicah, bolj razumljive, fleksibilne in donosne za komitente kot obstoječa zavarovanja, ki jih ponujajo zavarovalniški agentje. Tako sta se razvili dve osnovni skupini ponudb:

- **Varčevanja in naložbe z zavarovanjem življenja** sestavljajo različne storitve, odvisne od oblike upravljanja s prihranki glede na komitentovo pripravljenost za prevzem tveganja – prek bančnih depozitov, naložb z garantiranim donosom ali naložb v vzajemne sklade s prevzemom tveganja.
- **V bančne storitve** (kot so krediti, kartice in osebni računi) **vgrajena čista življenjska zavarovanja**, ki predstavljajo dodano vrednost za povečanje zadovoljstva in zvestobe najboljših komitentov ali kot kritja v primeru smrti.

V skladu z veljavno zakonodajo v Sloveniji banke ne morejo opravljati zavarovalniških dejavnosti, ampak mora biti le-ta organizirana v univerzalni zavarovalnici (velja za obstoječe zavarovalniške družbe) oziroma v specializirani zavarovalnici za življenjsko oz. premoženjsko skupino zavarovalnih storitev. Glede na to je bila ustanovljena nova zavarovalniška družba NLB Vita d. d., ki je v popolni lasti NLB d. d. in belgijske banke KBC. NLB Vita d. d. v začetku ne bo imela komercialne funkcije in ne bo izvajala trženjskih aktivnosti. Delovala bo kot »produktna tovarna« s pripravo podlag za storitve, spremljanje poslovanja, izplačila škod,… Novo razvite bančno-zavarovalniške storitve so tako sestavni del bančne ponudbe, ki se trži izključno v poslovalnicah in preko sodobnih tržnih poti. Bančni komercialisti, ki tržijo bančno-zavarovalniške storitve, morajo pred začetkom trženja opraviti ustrezna izobraževanja oz. morajo pridobiti licenco zavarovalnega posrednika. Licenco dodeli Agencija za zavarovalni nadzor na podlagi uspešno opravljenega izpita. Poleg pridobljene licence pa morajo vsi komercialisti opraviti tudi interno izobraževanje, ki je vsebinsko vezano na konkretne storitve, ki se tržijo v poslovalnicah. Trenutno se v okviru bančnega zavarovalništva tržijo naslednje storitve:

- NLB varčevanje Vita Plus: predstavlja tradicionalno varčevanje in naložbe z zajamčenim donosom ter zavarovalnim kritjem. Izplačilo: v primeru doživetja – privarčevana vsota v enkratnem znesku, v primeru kritične bolezni ali smrti – privarčevana ali zavarovalna vsota (kjer je znesek višji), v primeru nezgodne smrti ali trajne invalidnosti – dvojna privarčevana vsota ali zavarovalna vsota (kar predstavlja višji znesek).
- NLB Naložba Vita Plus: je moderno varčevanje in naložba z zavarovalnim kritjem. Donos je odvisen od izbranih skladov, možna je delna prekinitev plačila premije ob pogoju, da po dvigu vrednost police ostane 250.000 SIT. Izplačilo premije je možno ob izteku police – privarčevana vsota v enkratnem znesku, v primeru kritične bolezni ali smrti je privarčevana ali zavarovalna vsota (kar je višji znesek), v primeru nezgodne smrti ali 100% invalidnosti pa je izplačilo dvojna privarčevana vsota ali zavarovalna vsota (kar je višji znesek).

- Življenjsko zavarovanje kreditojemalcev: je zavarovanje kreditojemalcev, ki najamejo stanovanjski hipotekarni in osebni kredit.
- NLB Naložba Vita: predstavlja enkratno naložbo v vzajemni sklad, ki ima omejen čas vplačil (običajno en mesec). Storitev zagotavlja garantirano glavnico in garantiran donos na ravni sklada ter možnost dodatnega dobička (zajamčen odstotek) na podlagi tržne vrednosti sklada ter ima osnovno življenjsko zavarovanje.

Nabor storitev, ki se tržijo, se stalno povečuje, saj se storitve sproti razvijajo in tako prilagajajo slovenskemu trgu.

7.2 Računalniška rešitev za podporo bančnemu zavarovalništvu

Ker je trženje bančno-zavarovalniških storitev v banki novost, je bilo za trženje omenjenih storitev treba pridobiti tudi ustrezno računalniško podporo. Za to sta obstajali dve možnosti:

- lastni razvoj računalniške rešitve, ki bo trženju bančno-zavarovalniških storitev v NLB d.d. pisana na kožo ali
- nakup obstoječega sistema angleškega dobavitelja, ki ga strokovnjaki s področja zavarovalništva v KBC dobro poznajo in so se zanj odločili tudi za nakup v podobnih zavarovalnicah na Madžarskem in v Češki republiki.

Po temeljitem pregledu informacijskega sistema angleškega dobavitelja je bila v banki sprejeta odločitev za prvo možnost, torej za lastni razvoj informacijskega sistema. Natančna analiza poslovnega področja je pokazala, da bomo potrebovali informacijski sistem, ki bo omogočal sklepanje ponudb in spremljavo poslovanja, podatke o sklenjenih ponudbah pa naj bi enkrat dnevno pošiljali na zavarovalnico NLB Vita d. d., kjer imajo za vodenje sklenjenih polic svoj informacijski sistem IGAS. Podatki morajo biti zavarovalnici posredovani v datoteki, katere struktura je natančno določena in bo omogočala enostaven uvoz podatkov v informacijski sistem zavarovalnice.

Informacijski sistem za podporo bančnemu zavarovalništvu bosta tako tvorili dve aplikaciji:

- aplikacija Evita, ki bo namenjena sklepanju ponudb, ažuriranju podatkov v podatkovni bazi, pregledovanju in spremljanju prodaje in
- aplikacija Evita-vmesnik, ki bo skrbela za pripravo ustrezne datoteke s podatki o sklenjenih policah v poslovalnicah in predstavlja t. i. vmesnik med aplikacijo Evita in informacijskim sistemom IGAS.

Aplikacija Evita je spletna aplikacija, ki deluje v lokalnem omrežju banke. Podatki so shranjeni v DB2 relacijski podatkovni bazi, uporablja se za trženje bančno zavarovalniških storitev v poslovni mreži NLB. Razvita je v skladu s pravili razvoja tovrstnih aplikacij v banki in omogoča: prijavo uporabnika in preverjanje pooblastil, izdelavo informativnega izračuna s projekcijo in grafom za poljubno obdobje, izdelavo ponudbe, popravljanje in

pregledovanje informativnih izračunov in ponudbe, zaključevanje ponudb in priprava ponudb za prenos v informacijski sistem IGAS, izdelavo poročil na nivoju poslovalnice, podružnice in banke, skrbniški modul za postavitev parametrov, ki jih potrebuje aplikacija za svoje delovanje in nastavitev ustreznih šifrantov. Obe aplikaciji bosta uporabljali že obstoječe podatke iz registra komitentov banke, kataloga bančnih storitev, skupnih šifrantov in organizacijskih enot banke.

V projektno skupino za razvoj informacijskega sistema so bili vključeni: vodja projekta, IT vodja projekta, tehnološki skrbnik, procesni skrbnik, trije razvijalci za razvoj spletne aplikacije, razvijalec paketnih obdelav in administrator podatkovne baze, več delavcev s področja trženja, skrbnik strežnika, skrbnik sistema kakovosti ter razvijalec storitev in vodja informacijske tehnologije iz NLB Vita.

7.3 Organizacija in izvedba testiranja informacijskega sistema

V NLB d. d poteka razvoj informacijskih sistemov v skladu z veljavno metodologijo razvoja in sprememb informacijskih sistemov. Namen metodologije razvoja je, da ureja odnose med uporabniki (naročniki) in izvajalci (dobavitelji) razvojne naloge, definira in poenoti postopke in delo pri razvoju in spremembah informacijskega sistema, izboljša kakovost izdelkov, omogoča sprotno pripravo dokumentacije, izboljša spremljanje dela, nadzor nad potekom dela in porabo virov, definira odgovornosti posameznih udeležencev pri razvoju, spremembah in uporabi informacijskega sistema (Metodologija razvoja in sprememb informacijskega sistema v NLB, 2004, str. 3). Trenutno veljavna metodologija je usklajena tudi z zahtevami standarda ISO 9001 in za razvoj novega informacijskega sistema zahteva naslednje faze razvoja (Metodologija razvoja in sprememb informacijskega sistema v NLB, 2004, str. 8):

- zagon projekta,
- načrtovanje (priprava logičnega in fizičnega modela sistema),
- realizacija sistema,
- sprejemni test uporabnikov,
- uvedba informacijskega sistema v produkcijsko okolje,
- zaključek projekta.

Celoten cikel razvoja je oblikovan kot zaporedje faz. Za vsako od zgoraj navedenih faz je po metodologiji razvoja informacijskih sistemov določeno, kakšno dokumentacijo je treba pripraviti, definirani so izvajalci, vhodi in izhodi. Vsaka naslednja faza se ne more začeti, če izdelki predhodne faze niso bili pregledani in potrjeni s strani vodje projekta in skrbnika sistema kakovosti.

Za potrebe izvedbe testiranja je bila na novo formirana testna skupina. V banki ni zaposlenih delavcev, ki bi se ukvarjali izključno s testiranjem informacijskih sistemov, pač pa se testna skupina formira po potrebi v odvisnosti od sistema, ki ga razvijamo. Tako so testno skupino za testiranje sestavljali: več končnih uporabnikov, zaposleni s področja trženja v NLB d. d., ki se ukvarjajo z bančnim zavarovalništvom, procesni tehnolog,

tehnološki skrbnik ter aktuar, razvijalec storitev in vodja informacijske tehnologije iz NLB Vita. Za svoj del testiranja so bili seveda zadolženi tudi vsi razvijalci, v zaključnih fazah pa so se vključili tudi: administrator podatkovnih baz, skrbnik strežnika, kjer je nameščena spletna aplikacija in udeleženci izobraževanja v prvih dveh skupinah.

Vlogo vodje testiranja je prevzel tehnološki skrbnik, ki je tudi povezovalni člen med vsemi udeleženci testiranja in razvijalci. Zadolžen je bil tudi za pripravo plana testiranja, hkrati pa je poskrbel za organizacijo testiranja in obveščanje o spremembah in dopolnitvah v razvijajočem sistemu. Ob prvem testiranju je bila njegova skrb predstavitev testnega plana testni skupini in seznanjanje s funkcionalnostmi novega sistema. Ob testiranju informacijskega sistema za bančno zavarovalništvo smo s formiranjem testne skupine začeli v fazi realizacije sistema, takrat je tehnološki skrbnik tudi pripravil testni plan za izvedbo testiranja.

Imenovana testna skupina se je ukvarjala samo s testiranjem spletne aplikacije. Testiranje paketnih obdelav pa se je izvajalo v zavarovalnici NLB Vita d.d., kjer se tudi dejansko vrši je prevzemanje in preverjanje pravilnosti pripravljene datoteke. Preizkuševalci iz testne skupine nimajo s sistemom IGAS nikakršne povezave, niti ne poznajo njegovih funkcionalnosti. Celoten opis testiranja, ki ga navajam v nadaljevanju, se tako nanaša samo na testiranje spletne aplikacije Evita.

7.3.1 Izvedba testiranja v fazi zbiranja uporabniških zahtev

V prvi fazi razvoja informacijskega sistema je bila določena skupina, ki je bila zadolžena za definiranje uporabniških zahtev. Običajno takšno skupino tvorijo uporabniki, ki dobro poznajo problemsko področje, ki naj bi ga bodoči sistem pokrival. Ker je bil takrat bančno zavarovalništvo povsem nov posel v banki, s katerim do tedaj ni nihče v banki imel pravih izkušenj, so skupino tvorili sodelujoči na projektu iz Sektorja za marketing in razvijalci storitev iz NLB Vite d. d., vključenih je bilo tudi nekaj bodočih uporabnikov sistema, ki so že opravili ustrezno izobraževanje in izpit za opravljanje zavarovalniških poslov. Člani izbrane skupine so bili zadolženi za izvedbo aktivnosti v prvi fazi razvojnega procesa. Torej je bila njihova naloga: natančno zbiranje zahtev in želja, katerih realizacijo končni uporabniki pričakujejo v novem informacijskem sistemu in priprava ustrezne dokumentacije, iz katere so te zahteve jasno razvidne. Ko se bile glavne zahteve definirane, se je uporabniški skupini priključil tudi eden od informatikov, ki so sodelovali pri projektu razvoja sistema. Zbrane uporabniške zahteve je potrjeval v smislu, da je realizacija s stališča informatike sprejemljiva. Izkušen informatik je tako uporabniški skupini pomagal z nasveti, predlogi, alternativnimi rešitvami in tako poskrbel, da je realizacija zapisanih zahtev izvedljiva. Pod njegovim vodstvom je uporabniška skupina pripravila primere uporabe (angl. use-case) za vse identificirane poslovne dogodke.

Delo skupine je ves čas potekalo v teamu, končni izdelek, ki so ga pripravili, pa je poleg popisanih zahtev in primerov uporabe vseboval tudi opis scenarija za izvedbo vsakega dogodka. Za lažjo ponazoritev izvedbe so bile pripravljene tudi slike uporabniških vmesnikov. Ko je uporabniška skupina zaključila s svojim delom in pripravo

dokumentacije, so bili k sodelovanju povabljeni še trije bodoči končni uporabniki informacijskega sistema, ki so pregledali pripravljeno dokumentacijo. Na podlagi pripravljenih scenarijev in s pomočjo vnosnih ekranov so si poskušali zamisliti izvedbo posameznega poslovnega dogodka. Skupina uporabnikov se je sestala z vsakim od njih ločeno in si pozorno zapisovala vse njihove predloge in pripombe. Zbrane predloge je skupina še enkrat pregledala in v skladu z njimi dopolnila obstoječo dokumentacijo. Za pravilnost zbranih uporabniških zahtev pa s podpisom jamči predstavnik uporabnikov.

Podpisan dokument z zahtevami dobi v pregled še skrbnik sistema kakovosti, ki preveri, ali so bili izdelani vsi v tej fazi predvideni dokumenti, hkrati pa preveri še ostalo zagonsko dokumentacijo, npr. plan projekta in plan kakovosti ter preveri, ali je razvojna naloga stroškovno in rokovno sprejemljiva in hkrati da zeleno luč za nadaljevanje projekta.

Testna skupina v tej fazi posebnega testiranja še ni opravljala, saj takrat še niti ni bila formirana.

7.3.2 Izvedba testiranja v fazi načrtovanja

Po potrditvi narejenih izdelkov v prvi fazi razvojna skupina lahko nadaljuje s fazo načrtovanja. V tej fazi je na podlagi informacijskih potreb poslovnega področja, ki so zapisani v dokumentih, pripravljenih in zapisanih v prejšnji fazi razvojne naloge, treba izdelati logični model sistema. Logični model mora vsebovati: entitetni diagram, opise atributov in entitet, seznam poslovnih dogodkov, dekompozicijski diagram in povezave modelov. V našem primeru so pri načrtovanju sistema sodelovali samo informatiki, saj se pri tem uporabljajo predpisana orodja, uporabniki pa pogosto nimajo ustreznih znanj za izdelavo logičnega podatkovnega modela niti za uporabo orodja za načrtovanje. Ob zaključku izdelave logičnega modela se znotraj razvojne skupine model še enkrat pregleda in uskladi, hkrati pa člani skupine poskrbijo še za uskladitev s podatkovnim modelom banke, saj mora biti načrt sistema tak, da je novo rešitev možno integrirati v obstoječ sistem.

Načrtovalci so poskrbeli, da je bil verificiran logični model nazorno in natančno predstavljen uporabnikom in tudi ostalim tehničnim strokovnjakom. Ob zaključku morajo uporabnik, tako kot v fazi zbiranja zahtev, s podpisom potrditi izdelan načrt sistema. Končno verifikacijo opravi še skrbnik sistema kakovosti v banki, ki preveri, ali so bili izdelani vsi predvideni dokumenti, ali obstaja potrditev, da je logični model usklajen s podatkovnim modelom banke, ali sta podatkovni in procesni del logičnega modela sistema skladna s poslovanjem, ki bo računalniško podprto.

Na podlagi logičnega modela se izdela še fizični model, ki podaja način realizacije sistema na podlagi poslovnih zahtev iz logičnega modela in tehnoloških standardov. V fizičnem modelu se pripravi fizična shema baze podatkov, strukture zapisov v bazi, slike uporabniških vmesnikov in oblika poročil. Člani razvojne skupine ob koncu izdelave fizični model še enkrat pregledajo in ga verificirajo, predstavljen pa je tudi skupini uporabnikov in skupini tehničnih strokovnjakov.

Tudi v fazi izdelave fizičnega modela skrbnik sistema kakovosti opravi kontrolni pregled. Z verifikacijo potrdi, da je bila izdelana vsa predvidena dokumentacija, če so specifikacije izdelane v skladu z logičnim modelom sistema in če ustrezajo internim standardom in priporočilom, ki veljajo za posamezno razvojno programsko opremo.

7.3.3 Izvedba testiranja v fazi realizacije sistema

Z izdelavo fizičnega modela je pravzaprav vse pripravljeno za začetek dela v fazi realizacije sistema. Administratorji podatkovne baze po načrtu kreirajo ustrezne tabele v podatkovni bazi, razvijalci pripravijo potrebno okolje za začetek razvoja sistema, pripravljene pa morajo biti tudi programske specifikacije. Hkrati z razvojem pa so se začele tudi priprave na testiranje sistema. Zanj so zadolženi razvijalci, tehnološki skrbnik aplikacije in testna skupina.

7.3.3.1 Testiranje enote, integralno in sistemsko testiranje

V skladu z metodologijo razvoja informacijskih sistemov v banki morajo biti pred predajo sistema v testiranje uporabnikom opravljena vsa t. i. interna testiranja. Interno testiranje tvorijo testiranje enote proizvoda in integralno testiranje, ki ga opravijo razvijalci in sistemsko testiranje, ki ga opravi tehnološki skrbnik. Integralno in sistemsko testiranje sta bila v našem primeru še posebno pomembna, saj je pri delu na projektu sodelovalo več razvijalcev, ki so bili zadolženi vsak za svoj del razvoja aplikacije. Tako je vsak zadolžen za preverjanje pravilnosti delovanja svojega dela, čemur pa v nadaljevanju sledi še preverjanje delovanja povezanih modulov. Sama izvedba testirana enote ali integralno testiranje zahteva kar nekaj priprav na izvedbo: razvijalci so morali pripraviti potrebno testno konfiguracijo, ki v tistem trenutku nadomešča sistem, v katerega bo modul vgrajen, na podlagi poznane logike programa pa so pripravili ustrezne testne primere. Samo testiranje se izvaja z vnosom testnih podatkov in pregledovanjem pričakovanih in dobljenih rezultatov testiranja. Pri testiranju oz. odkritih napakah razvijalec pregleduje kodo in jo dopolnjuje oz. spreminja.

Preizkuševalci morajo za vsako testiranje pripraviti ustrezne zapise o testiranju, ki nedvoumno dokazujejo, da je bilo testiranje opravljeno. Vse odkrite napake pri internem testiranju se natančno opišejo, če je možno, se priložijo slike vnosnih ekranov, kjer se pojavi napaka. Ko so odpravljene napake, se proces testiranja ponavlja, dokler razvijalci niso zadovoljni z rezultati testiranja. Po opravljenem integralnem testu se sistem preda v sistemsko testiranje.

Sistemsko testiranje opravi tehnološki skrbnik sistema, ki ima pregled nad celotnim sistemom, ki se razvija. Preveriti mora delovanje aplikacije, ali so vsi programi med seboj pravilno povezani in ne prihaja do kakšnih velikih napak, ki bi že takoj na začetku onemogočile izvajanje nadaljnjega testiranja. S sistemskim testiranjem preverimo, ali so vsi programi vključeni, ali se kličejo na pravem metu, hkrati pa se preverja tudi, ali je sistem zgrajen v skladu s narejenimi specifikacijami. Tehnološki skrbnik preveri tudi delovanje posamezne funkcionalnosti v aplikaciji, vendar se ne spušča podrobneje v pravilnost delovanja funkcionalnosti - to prepusti testni skupini. Tudi za izvedbo

sistemskega testiranja tehnološki skrbnik pripravi ustrezne zapise o testiranju, kjer je razvidno, da je bilo le-to izvedeno in sistem ustreza do te mere, da ga je možno predati v testiranje testni skupini ali pa je bilo v sistemu ugotovljeno določeno število napak, ki jih morajo razvijalci odpraviti.

Po zaključenem sistemskem testiranju se tehnološki skrbnik v primeru ugotovljenih napak sestane z razvijalci in jim pojasni, kakšne so bile odkrite napake. Skupaj pregledajo specifikacije in se dogovorijo o tem, kakšno je pričakovano delovanje aplikacije in kakšni so popravki oz. spremembe, ki jih je treba v sistemu narediti. Vsaka takšna sprememba zahteva ponoven cikel izvajanja testiranja: ponovno se opravijo testiranja enote, integralno in sistemsko testiranje. Ko večjih napak ni več, pa lahko začne s testiranjem tudi testna skupina.

7.3.3.2 Priprava testnega plana

Pripravo testnega plana smo izvajali vzporedno z ostalimi aktivnostmi, ki so potekale v fazi realizacije rešitve. Testni plan je izdelal tehnološki skrbnik aplikacije, ki skrbi tudi za organizacijo testiranja.

Testni plan sestavljajo štiri glavna poglavja:

1. Splošne informacije: kjer je zapisano ime in verzija aplikacije, ki se bo testirala. Opisane so glavne značilnosti, kakšna je povezanost z ostalimi sistemi in komu je uporaba sistema sploh namenjena. Predstavljen je sistem pooblastil in podrobno so vpisane vse funkcionalnosti sistema. Opisani so cilji in pričakovani rezultati testiranja ter referenčni dokumenti, ki so na voljo in v pomoč vsem izvajalcem testiranja.
2. Organizacija testiranja: v tem poglavju je naveden imenski seznam preizkuševalcev. Tu so zapisana tudi pravila in postopki za izvedbo testiranja, našteta pa je tudi dokumentacija, ki jo morajo preizkuševalci pripraviti pri testiranju.
3. Tretje poglavje navaja priporočila za pripravo testnih primerov. Ločeno po posameznih funkcionalnostih so za vsa polja na uporabniškem vmesniku prikazani veljavni in neveljavni vnosi ter vgrajene kontrole. Hkrati pa je v tem poglavju tudi navedeno kje in koliko časa se morajo testni primeri hraniti.
4. Časovni plan testiranja, s katerim določimo čas izvedbe testiranja posamezne aktivnosti v okviru testiranja celotnega sistema.

Pripravljen terminski plan so dobili v pregled vsi, ki so bili kakorkoli povezani s testiranjem. Vsi člani testne skupine so pred začetkom testiranja pregledali svoje zadolžitve in podali svoje morebitne pripombe, v skladu s pripombami pa se testni plan po potrebi korigira.

Primer testnega plana, ki smo ga pripravili za testiranje spletne aplikacije Evita za podporo bančnemu zavarovalništvu, se nahaja v Prilogi 1.

7.3.3.3 Sprejemno testiranje

Sprejemno testiranje opravi testna skupina, ki preveri, ali je razvojni team res izdelal tisto, kar je naročnik naročil. Za preizkuševalce, ki so bili izbrani v testno skupino za testiranje aplikacije Evita, je bilo značilno, da niso imeli praktično nobenih večjih izkušenj ali kakšnih dodatnih znanj s področja testiranja. Nekateri med njimi so pred časom že sodelovali pri razvoju in testiranju drugih informacijskih sistemov, večina pa je bila pred tako nalogo prvič. Prav zato smo pred začetkom testiranja precej pozornosti posvetili sestanku tehnološkega skrbnika s testno skupino. Namen sestanka je bil natančneje seznaniti testno skupino s planom testiranja in njihovimi zadolžitvami v tem procesu, jih seznaniti z različnimi tehnikami testiranja, z uporabo aplikacije in vsemi funkcionalnostmi v sistemu, katerih delovanje je bilo treba preveriti. Opozorjeni so bili, na kaj morajo biti še posebno pozorni, kako mora biti izpolnjena dokumentacija o testiranju, dogovorili pa so se tudi, kakšen bo način poročanja o ugotovljenih napakah in nepravilnostih v sistemu.

Vsi člani testne skupine so testiranje opravljali poleg svojih rednih delovnih obveznosti, kar pomeni, da so testiranje izvajali na svojem delovnem mestu. V odvisnosti od potreb se je testna skupina med sprejemnim testiranjem sestajala na skupnih sestankih, ki so bila namenjena razčiščevanju morebitnih vprašanj in težav pri testiranju ter izmenjavi izkušenj. Na začetku so takšna srečanja pogostejša, kasneje vedno bolj redka.

7.3.3.3.1 Ugotavljanje napak in poročanje o napakah

Testna skupina je izvajala sprejemno testiranje v skladu s testnim planom. To pomeni, da se je držala predpisanih rokov in da je vsak od preizkuševalcev testiral natanko tisti del, za katerega je bil zadolžen. Vsak preizkuševalec je poskrbel za pripravo testnih primerov in se držal v testnem planu predpisanih pravili pri izbiri načinov testiranja. Prav tako so bili v skladu s priporočili pripravljeni tudi testni primeri, ki so jih za testiranje pripravili preizkuševalci, shranjeni pa so bili v taki obliki, da jih je bilo možno arhivirati in jih tako kadarkoli spet uporabiti za testiranje aplikacije.

Preizkuševalci so preverjali delovanje programa tako, da so vnašali pripravljene vhodne podatke in preverjali, kako se program po vnosu podatkov obnaša. Vsak rezultat so si skrbno zapisali oz. so pripravljali zapise o testiranju. Iz zapisa o testiranju mora biti razvidno, katera verzija aplikacije se testira, katera funkcionalnost v aplikaciji se testira, kakšne rezultate pričakujemo in kakšni so rezultati, ki jih pri testiranju dejansko dobimo. Primer tako izpolnjenega zapisa z vsemi opisanimi elementi se nahaja na naslednji strani.

Zapis o testiranju

Ime produkta: aplikacija Evita

Oznaka verzije: 01.00.00

Datum testiranja: 17. 9. 2003 **Preizkuševalec**: Preizkuševalec 1

Opis funkcije ter vsebina testiranja: izdelava informativnega izračuna za storitev NLB Varčevanje Vita plus.

Testni podatki:

- Ime in priimek: Marija Novak
- Telefon: 02 856 933
- Spol: ženski
- Datum rojstva: 19. 1. 1969
- Frekvenca plačevanja: letno
- Znesek premije: 500 €
- Dodatni vprašalnik: /
- Kritje za smrt: da zavarovalna vsota: 20.000 €
- Kritje za nezgodno smrt: ne
- Kritje za kritične bolezni: ne
- Znesek dodatne premije: 1200 €
- Obdobje za prikaz: 20 let

Pričakovani rezultati:

Za vpisani podatek znesek zavarovalne premije se mora izpisati sporočilo o napaki in informacija o pravilni višini vpisane premije za letno frekvenco plačevanja, ki je 600 €.

Zavarovalna vsota za smrt je lahko največ 16.000 € (starost do 50 let in brez dodatnega vprašalnika). Izdelava informativnega izračuna v tem primeru ni možna. Ko se poveča znesek premije in zniža zavarovalna vsota za smrt, mora biti možno izdelati informativni izračun.

Dejanski rezultati:

Za vpisani podatek znesek zavarovalne premije se izpiše opozorilo o višini najvišje vpisane premije za letno frekvenco plačevanja, ki je 600 €. Po vpisu zavarovalne vsote za smrt se izpiše ustrezno opozorilo. Izdelava informativnega izračuna v tem primeru ni možna.

Ko se poveča znesek premije in zniža zavarovalna vsota za smrt, je informativni izračun možno izdelati.

Dodatni opisi, pojasnila in opombe, priloge: /

Primer izpolnjenega zapisa o testiranju

Odkrivanje in odkritje napak pa je pravzaprav šele začetek. Ni namreč dovolj, da napake samo odkrijemo, znati moramo tudi poročati o njih. Slabo pripravljeno poročilo je ena od klasičnih napak, ki jih je pogosto zaznati pri testiranju (Marick, 2004) in bistveno vpliva na sam potek odpravljanja napak. Prav zaradi tega je bilo pri testiranju aplikacije Evite namenjene precej pozornosti tudi poročanju o napaki. V primeru ugotovljene napake je preizkuševalec o njej poročal tehnološkemu skrbniku. Za poročanje so preizkuševalci uporabljali pripravljen obrazec, ki se nahaja na zadnji strani testnega plana v Prilogi 1. Iz njega je razvidno, pri testiranju katere funkcionalnosti je nastala napaka, kakšni so bili vhodni podatki in opis napake. Za pomoč razvijalcem se pri testiranju spletne aplikacije lahko zapiše še URL naslov strani, kjer se napaka pojavi. Po pripravljenem poročilu preizkuševalec skuša ponovno izvesti tako zaporedje korakov, da pride do napake. S tem preveri ali je zaporedje zapisanih korakov pravilno. Če mu to ne uspe, to označi tudi na obrazcu.

V primeru, da je ugotovljena napaka takšna, da onemogoča normalen potek nadaljevanja testiranja oz. testiranje sistema s tako veliko napako ne bi imelo nobenega smisla, ker ta vpliva tudi na rezultate testiranja ostalih funkcionalnosti, mora preizkuševalec poskrbeti, da je z njo čimprej seznanjeni tehnološki skrbnik. S pošiljanjem napake torej ne odlaša do konca dneva, pač pa ga o tem nemudoma opozori. Tako se lahko izognemo večjim odstopanjem od terminskega plana testiranja. V primeru obvestila o večji napaki, tehnološki skrbnik napako preveri in po potrebi ustavi nadaljnje izvajanje testiranja.

7.3.3.3.2 Odpravljanje napak

Prejete preglednice rezultatov testiranja tehnološki skrbnik natančno analizira in zapisane napake preveri. Pogosto se namreč zgodi, da preizkuševalci poročajo o napaki, po analizi pa se pokaže, da je v sistemu vendarle ni, potrebujejo pa le kakšno dodatno informacijo ali razlago za nadaljevanje izvedbe testiranja.

Vsako napako, o kateri so poročali preizkuševalci, tehnološki skrbnik uvrstiti v dogovorjene tipe napak:

- napaka tipa A: je vsaka napaka, ki preprečuje delovanje in uporabo aplikacije ali nekega dela poslovnih aktivnosti,
- napaka tipa B: je vsaka napaka, ki povzroča večjo/resnejšo neustreznost programske podpore v primerjavi z izraženimi poslovnimi zahtevami banke. V praksi bi bilo možno delovanje programske podpore tudi brez te funkcionalnosti oz. z izbiro alternativne funkcionalnosti programske podpore,
- napaka tipa C: je vsaka napaka, ki ne vpliva na učinkovito uporabo aplikacije in je bolj »lepotna«, vključno z manjšimi napakami na ekranih ali manjšimi odstopanji od pričakovanega delovanja.

Če je napak veliko, se glede na razvrstitev v določeno skupino določi tudi prioriteta odpravljanja napak. Največjo prioriteto pri odpravljanju morajo tako imeti napake tipa A,

nato napake tipa B in nato še napake tipa C. V odvisnosti od tega, kako resna je napaka, tehnološki skrbnik organizira sestanek z razvijalci, kjer skupaj pregledajo ugotovljene napake in se dogovorijo o njihovem odpravljanju. Na začetku testiranja so takšni sestanki pogostejši, lahko tudi večkrat dnevno, ker pa se napake odpravljajo sproti, je potreb po takšnih sestankih vedno manj.

Po sestanku razvijalci začnejo z odpravljanjem ugotovljenih napak. Če gre za napake tipa A ali B, je treba ponoviti tudi vsa interna testiranja, nato se pripravi nova verzija aplikacije, kjer testiranje opravi najprej tehnološki skrbnik aplikacije. Ko so odpravljene vse napake, aplikacijo ponovno dobi v testiranje testna skupina. Tudi takrat je bilo treba pripraviti zapise o testiranju, tako na strani razvijalcev kot tudi na strani tehnološkega skrbnika.

Tehnološki skrbnik poskrbi, da je testna skupina ažurno obveščena o popravkih in opozori na morebitno dodatno pazljivost pri testiranju. Testna skupina opravi najprej regresijsko testiranje, nato pa, po pripravljenem terminskem planu, nadaljuje s testiranjem tistega dela aplikacije, ki je do takrat ostal še nepreverjen. Navadno se tak cikel testiranja, sporočanja in odpravljanja napak ponovi večkrat, še posebno v primeru, ko gre za razvoj kompleksnih informacijskih sistemov.

Ko je testna skupina opravila svojo nalogo in je bil vodja testiranja v celoti zadovoljen z rezultati, napak v aplikaciji pa načeloma ni bilo več, se tehnološki skrbnik dogovori za potrditev in podpis dokumenta Poročilo o uporabniškem sprejemnem testu. Dokument za sprejemni test pripravi tehnološki skrbnik, ki po planu testiranja opredeli odgovorne osebe za potrditev delovanja posamezne funkcionalnosti. Udeleženci testiranja morajo že ob začetku vedeti, za potrditev delovanja katerega področja so zadolženi. Iz poročila mora biti razvidno, kdo so bili udeleženci testiranja, datum začetka in konca testiranja, opis poteka sprejemnega testa in ugotovitve o ustreznosti testiranega informacijskega sistema. Primer pripravljenega dokumenta se nahaja v Prilogi 3.

7.3.3.4 Regresijsko testiranje

Regresijsko testiranje zajema izvedbo vseh testnih primerov po izvedenem popravku, spremembi ali izboljšavi. Za takšno vrsto testiranja je pomembno, da se ohranijo testni podatki, s katerimi smo odkrili napako ali pomanjkljivost, ki je privedla do popravkov. Preizkuševalec pa mora poskrbeti tudi za pripravo novih testnih primerov. S testnimi primeri pri regresijskem testiranju preverjamo:

- Ali so bile odpravljene napake, zaradi katerih je bilo izvedeno popravljanje programa; testiranje ponovimo na istih testnih podatkih, s katerimi smo našli napake v programu. Če zopet pridemo do napake, z regresijskim testiranjem ni smiselno nadaljevati.
- Ali so nastale kakšne nove napake: ob spremembah v programu je lahko nastala kakšna druga napaka v programu, ki je povezana z odpravljeno napako.

- Delovanje ostalega programa: spremembe v programu lahko povzročijo še kakšne nepričakovane napake v programu.

7.3.3.5 Stresno testiranje

Stresno testiranje opravi testna skupina v sodelovanju z administratorji podatkovnih baz, Skupaj preverijo, kako stabilen je sistem v različnih okoliščinah, ki nastajajo v produkcijskem okolju. Še celo več: namen stresnega testiranje je, da se ustvari okolje, ki je »zahtevnejše« od tistega v katerem bo delovala aplikacija. V našem primeru smo preverjali, kako se sistem obnaša v primeru:

- večjih obremenitev strežnika,
- prekinitev dostopa do podatkovne baze,
- nedelovanja podatkovne baze,
- prekinitve mrežne povezave.

Pri nameščanju testne verzije aplikacije smo poskrbeli, da je bila le-ta nameščena na strežniku skupaj z več drugimi aplikacijami. Tako smo lahko ves čas testiranja preverjali kako sistem deluje v povezavi z ostalimi aplikacijami. Lahko se namreč obnaša drugače na »praznem« strežniku kot pa na strežniku, kjer so nameščene še druge aplikacije. Pri tovrstnem stresnem testiranju smo bili pozorni na odzivne čase, posebno pri zahtevnejših poizvedbah, na katere opozorimo testno skupino pred začetkom testiranja.

V produkcijskem okolju pa naletimo še na vrsto okoliščin, ki jih poskusimo simulirati tudi v testnem okolju, npr. prekinitev dostopa do podatkovne baze, nedelovanje podatkovne baze, prekinjena mrežna povezava. V prvih dveh primerih za simulacijo okoliščin poskrbi administrator podatkovne baze. Testna skupina poskuša nemoteno nadaljevati s testiranjem po pripravljenem testnem planu, medtem ko administrator prekine povezavo oz. poskrbi za začasno nedelujoče testno okolje. Podobno lahko storimo tudi v primeru prekinitve mrežne povezave, kar lahko naredijo preizkuševalci sami (npr. izvlečejo mrežni kabel) ali pa za začasno prekinitev poskrbi strokovnjak s področja komunikacij. Testna skupina tako spremlja reakcijo aplikacije in sporočilo, ki se pojavi na ekranu, preveri, kako so, če sploh so, shranjeni podatki, ali je po vzpostavljeni mrežni povezavi možno nadaljevati z delom, itd.

Stresno testiranje je navadno zahtevnejše in od preizkuševalcev zahteva že določeno mero izkušenj. Izkušeni preizkuševalci namreč natančno vedo, kako se mora v primeru prekinitve obnašati aplikacija, ali vrne uporabniku pravo sporočilo in ga usmeri k pravilnim nadaljnjim ukrepom (npr. obvestilo Help Deska, obvestilo tehnološkemu skrbniku itd.).

7.3.3.6 Performančno testiranje

Prepočasen odziv aplikacije lahko kljub popolni funkcionalnosti povzroči neuporabnost aplikacije (Ireson, 1988). Pri performančnem testiranju spremljamo odzivni čas od posredovanja zahteve do prikaza rezultata na zaslonu uporabnika. Že v začetnih fazah

razvoja informacijskega sistema je bila, v dogovoru med uporabniško skupino in razvijalci, postavljena zahteva, naj odzivni čas posamezne transakcije, ki se pogosteje izvajajo (npr. iskanje komitenta, sklepanje ponudbe,…) ne traja dlje kot 1 – 1,5 sekunde, za zahtevnejše preglede, ki se prožijo manj pogosto (npr. pregled na nivoju banke, s katerim se formira promet posameznih organizacijskih enotah, skupaj z vsemi nadrejenimi enotami) pa je bil priporočljiv odzivni čas od 3 - 4 sekunde. Da pa bi lahko nadzirali in preverjali čas izvajanja posamezne transakcije, se po vsakem klicu komponente, ki na zahtevo uporabnika poskrbi za nabor ustreznih podatkov, naredi zapis v datoteko na strežniku, t. i. programsko sled (ang. log file). Iz nje je razvidno, poimensko katera transakcija se je izvedla, uporabnik, ki je transakcijo sprožil, kaj transakcija naredi (pregled, popravek, nov zapis) in koliko časa je trajala sama izvedba. S pregledom programske sledi je tako možno na zelo enostaven način spremljati izvajanje transakcij in poskrbeti za odpravo dolgih odzivnih časov.

Performančno testiranje se je tako izvajalo vzporedno z ostalimi vrstami testiranja. Razvijalci so zapise v datoteki programske sledi pregledovali večkrat dnevno v času izvajana internega testiranja, sprejemnega testiranja in izobraževanja uporabnikov. Takšen način spremljanja izvedbe časa posameznih transakcij pa omogoča nadzor nad izvajanjem tudi kasneje, ko sistem deluje v produkcijskem okolju.

Pri testiranju testne skupine tako performančnemu testiranju nismo posvečali posebne pozornosti. Kritični del, ki bi lahko bil pomemben za performančno testiranje, predstavlja izdelava poročil na različnih nivojih (nivo poslovalnice, podružnice, banke). Tu gre za prikaz večje količine podatkov na zaslonu uporabnika. Ker pa je narava dela tudi v tem primeru taka, da se poročila pripravljajo znotraj krajšega časovnega obdobja (npr. en mesec), tudi v tem primeru nismo načrtovali posebnega performančnega testiranja. Preizkuševalci so bili sicer opozorjeni na to, da morajo kot napako prijaviti tudi dolge odzivne čase na posredovane zahteve v sistemu.

Približno na sredini izvedbe sprejemnega testiranja, ko je bila odpravljena že večina večjih napak, smo izvedli še drugi del performančnega testiranja. Ocenili smo, koliko zapisov bo v osnovnih tabelah nastalo v produkcijskem okolju v obdobju 3-4 let. S posebej za to pripravljenimi procedurami smo povečali število zapisov za 100.000 in tako prve dni testiranja še natančneje spremljali odzivne čase posameznih transakcij.

V celotnem obdobju sprejemnega testiranja in izobraževanja uporabnikov smo tako dobili obvestilo o dveh napakah, ki sta izvirali iz performančnega testiranja. Po raziskavi prve smo ugotovili, da je bil daljši odzivni čas samo začasna težava zaradi upočasnitve delovanja testnega okolja, kjer se nahaja DB2 podatkovna baza aplikacije. V drugem primeru pa smo v sodelovanju z administratorjem podatkovnih baz preverili poizvedbo za nabor podatkov iz podatkovne baze. Do napake je prišlo v drugem delu performančnega testiranja zaradi povečanja števila zapisov v tabelah. Z izvedbo ukaza EXPLAIN v DB2 podatkovni bazi je administrator baze preverili način izvajanja napisane poizvedbe. Stavek EXPLAIN je namreč nepogrešljiv, ko je treba preveriti in razumeti, zakaj posamezne poizvedbe trajajo predolgo ter kako jih napisati bolje in kako jih prilagoditi. Po opravljeni

analizi, ki jo je naredil administrator podatkovnih baz, je bila narejena korekcija indeksov na tabelah, kjer se hrani največ podatkov. Tako smo izboljšali pristopno pot do podatkov in posledično tudi odzivni čas transakcije.

7.3.3.7 Obremenilno testiranje

Za informacijski sistem običajno trdimo, da deluje, ko z njim brez težav dela določeno število ljudi in ga obremeni do neke mere. Prav zato naj bi z obremenilnim testiranjem poskušali obremeniti sistem tako, kot pričakujemo, da bo obremenjen v produkcijskem okolju. Že v fazi zbiranja zahtev uporabnikov je bilo določeno, da se bo informacijski sistem za podporo bančnemu zavarovalništvu uporabljal v poslovni mreži NLB d. d.. Poslovno mrežo tvori približno 160 poslovalnic po vsej Sloveniji. Na začetku naj bi ga uporabljali v približno 60 poslovalnicah, kar pomeni največ na treh delovnih postajah v poslovalnici, postopno pa naj bi se uporaba razširila v vse ostale poslovalnice v poslovni mreži in pa tudi v banke bančne skupine (Koroška banka, Banka Domžale in Banka Zasavje) ter Banko Celje. Glede na začetne ocene o številu uporabnikov smo v razvojni skupini ocenili, da lahko obremenilno testiranje opravimo vzporedno z izobraževanjem, ki so ga opravljali bodoči uporabniki sistema. Skupina za izobraževanje je štela od 30 - 40 udeležencev. Izobraževanje pod vodstvom predavatelja poteka na podlagi seznanjanja z značilnostmi posamezne storitve in procesa ter posameznimi funkcijami v sistemu, sledi pa skupna izvedba funkcije. S takim načinom lahko zagotovimo npr. 30 - 40 hkratnih vnosov oz. poizvedb, ki se izvedejo v zelo kratkem časovnem intervalu, kar se kljub množični uporabi sistema v produkcijskem okolju le težko zgodi. Obremenilni test v času izobraževanja spremljata uporabniški skrbnik sistema, skrbnik strežnika, kjer je nameščena testna verzija sistema in administrator podatkovne baze. Oba skrbnika spremljata delovanje strežnika, v časovnih intervalih pregledujeta zapise o programski sledi v datoteki na strežniku, administrator pa spremlja aktivnosti v podatkovni bazi. Preverja, ali ne prihaja do medsebojnega zaklepanja tabel ali zapisov v tabelah oz. ali vsaka transakcija ob zaključku izvede tudi odklepanje na nivoju tabele, strani v tabeli ali zapisa v njej. Vsi skrbniki zapisujejo kakršnokoli odstopanje od želenega stanja. Za potrebe obremenilnega testiranja smo na ta način spremljali dve skupini, ki sta opravili izobraževanje za uporabo aplikacije Evita. Ker ni bilo ugotovljenih napak, so skrbnik poskrbeli samo še za pripravo ustreznih zapisov o testiranju, iz katerih je bilo razvidno, da je bilo testiranje opravljeno in smo bili z njim dobljeni pričakovani rezultati.

7.3.4 Izvedba testiranja v fazi uvedbe sistema

Po uspešno opravljenem sprejemnem, stresnem, performančnem in obremenilnem testiranju nastopi čas za uvedbo sistema. Faza uvedbe sistema zajema vrsto aktivnosti od definiranja tabel v produkcijskem okolju, formiranja in napolnitev šifrantov z ustreznimi podatki, dodeljevanja pooblastil, do definiranja parametrov za posamezno storitev, namestitve sistema v produkcijskem okolju,… Prav slednje zahteva posebno vrsto testiranja, ki se imenuje operacijsko testiranje.

7.3.4.1 Operacijsko testiranje

Z operacijskim testiranjem preverimo, kako poteka namestitev aplikacije. Potek operacijskega testiranja je odvisen od vrste aplikacije. V primeru aplikacije tipa odjemalec/strežnik lahko namestitev na delovno postajo opravi uporabnik sam po priloženih navodilih. V naše primeru, ko gre za spletno aplikacijo, pa se namestitev ne izvaja na delovne postaje uporabnika, pač pa samo na strežnik. Pri operacijskem testiranju sta v našem primeru sodelovala razvijalec aplikacije in skrbnik strežnika, na katerem bo nameščena produkcijska verzija aplikacije. Razvijalec aplikacije je zadolžen za pripravo navodil za namestitev, ki pa morajo biti pripravljena tako, da namestitev lahko kadarkoli opravi skrbnik strežnika sam. V navodilih mora biti naveden seznam vseh programov, komponent in datotek, ki so del aplikacije, zapisano mora biti mesto, kjer se posamezni elementi morajo nahajati, hkrati pa morajo biti podane tudi vse dodatne zahteve in nastavitve, ki so potrebne za uspešno delovanje aplikacije.

Testiranje namestitve opravlja skrbnik strežnika po navodilih, ki jih je pripravil razvijalec. Po opravljeni namestitvi se navadno takoj preveri prijava uporabnika. Ob prvi namestitvi aplikacije v produkcijskem okolje je možno izvesti tudi preverjanje delovanja ostalih funkcionalnosti (vnos, pregledovanje, popravljanje,…) še preden z delom začnejo tudi uporabniki. Tako preverimo, ali so bili v produkcijsko okolje preneseni res vsi potrebni elementi za delovanje aplikacije. Seveda moramo pred začetkom dela uporabnikov poskrbeti, da se testni primeri izbrišejo iz produkcijske podatkovne baze. Kot smo že zapisali, je takšno vrsto testiranja možno izvesti samo pred prvo produkcijo, ko s tem ne motimo dela uporabnikov, sicer pa se testiranje namestitve opravi na testni strežnik.

Tako kot za vse vrste testiranja mora tudi za takšno obstajati zapis, ki dokazuje kje in kdaj je bilo testiranje izvedeno, kdo je pri njem sodeloval in kako je potekalo. Na zapisih o testiranju mora biti nedvoumno zapisano, ali izdelek ustreza predpisanim standardom in ali se dovoljuje prenos aplikacije v produkcijsko okolje.

7.3.5 Priprava poročila o testiranju

Ob zaključku testiranja tehnološki skrbnik pripravi poročilo o poteku testiranja. V testnem poročilu navedemo, kaj smo testirali, naštejemo udeležence testiranja, navedemo dokumentacijo, ki smo jo uporabljali, opišemo vse vrste testiranja, ki smo jih opravili, priložimo zapise o testiranju in posredovana sporočila o napakah. Naloga tehnološkega skrbnika je, da naredi analizo vseh napak, jih razvrsti v skupine (napake tipa A, B, C) in zapiše, koliko od njih je bilo odpravljenih, koliko je še odprtih in razloge za to, da še niso bile odpravljene. V testnem poročilu se lahko zavedejo tudi kakršnakoli priporočila v zvezi z izgradnjo sistema, ki jih testna skupina sporoči tehnološkemu skrbniku. Priložen mora biti tudi podpisan dokument o potrjenem sprejemnem testu, kjer je zapisano, da je testna skupina opravila vsa predvidena testiranja in da sistem ustreza zapisanim zahtevam ter se dovoljuje prenos v produkcijsko okolje. Primer poročila o testiranju se nahaja v Prilogi 2.

Del poročila o testiranju je tudi zaključna ocena testiranja, kjer se oceni delo testne skupine.

7.3.6 Izvedba testiranja v fazi vzdrževanja sistema

Aplikacija Evita deluje v produkcijskem okolju že skoraj dve leti, od prvega prenosa v produkcijsko okolje pa smo opravili že kar nekaj sprememb in dopolnitev. Sprememba ali dopolnitev sistema v času vzdrževanja je lahko posledica neodkritih napak v programu v času razvoja, posledica zakonskih sprememb ali razširitev zahtev uporabnikov. Na srečo se nismo srečali z neodkritimi napakami, ki bi uporabnikom povzročale kakršnekoli težave oz. nedelovanje aplikacije. Smo se pa že večkrat srečali s spremembami zahtev uporabnikov oz. dopolnitvami, saj se obstoječe storitve neprestano dopolnjujejo, hkrati pa se razvijajo nove storitve. Ena od dopolnitev, ki smo jih izvedli kmalu po prvem prenosu v produkcijsko okolje, je bila:

- beleženje dodatnega komercialista za posamezno polico z namenom delitve provizije. Dodatni komercialist sicer nima ustreznih pooblastil za sklepanje življenjskih zavarovanj, vendar usmeri komitenta h komercialistu, ki ima za sklepanje bančno-zavarovalniških storitev ustrezno licenco in
- omejitev dostopa do zdravstvenega vprašalnika, ko je ponudba zaključena in je ni več možno popravljati. Ker so odgovori na zastavljena zdravstvena vprašanja osebni podatki, ki naj ne bi bili razkriti kateremukoli uporabniku aplikacije, jih po zaključku ponudbe ne prikazujemo več. Odgovori na vprašanja so, s posebnim pooblastilom, vidni samo še pooblaščenim delavcem na NLB Viti, ki jih pregledujejo za potrebe sklepanja polic.

Posredovani uporabniški zahtevi je najprej dobil v pregled tehnološki skrbnik, ki predloge za spremembo skrbno prouči in s spremembami seznani razvojni team. Skupaj se dogovorijo o popravkih v aplikaciji in določijo v katerih razvojnih fazah je treba narediti spremembe. Zgoraj opisana popravka v aplikaciji sta bila dokaj enostavna, vendar kljub vsemu zahtevata izvedbo vrste aktivnosti v skoraj vseh razvojnih fazah. Prva zahteva vpliva na spremembo logičnega in fizičnega modela sistema. V entiteti z imenom »Polica« je bilo dodan atribut z imenom »usmerjevalec«, kamor se bo vpisovala matična številka zaposlenega komercialista brez licence, ki bo sicer pridobil komitenta za sklenitev življenjskega zavarovanja in ga usmeril k drugemu komercialistu, ki bo sklenitev ponudbe lahko dejansko opravil. Spremembo v podatkovnem modelu opravi načrtovalec sistema skupaj s tehnološkim skrbnikom. Druga opisana sprememba pa ne zahteva nobenega popravka v načrtu sistema, ker gre le za spremembo v povezavi s pooblastili. Za vse ponudbe, ki so zaključene in se tako nahajajo v določenem statusu, uporabnikom s pooblastilom AGENT ob pregledovanju prikrijemo odgovore na vprašanja, pregledovanje pa omogočimo izključno uporabniku s pooblastilom VITA oz. delavcem, ki imajo dodeljeno to pooblastilo in so zaposleni na zavarovalnici NLB VITA d. d..

Na podlagi spremenjenega modela sistema administrator podatkovne baze opravi potrebne spremembe tabele v DB2 podatkovni bazi, tehnološki skrbnik pa pripravi programsko

specifikacijo za spremembo programov. Postopek testiranja se v fazi vzdrževanja sistema ne razlikuje od postopka testiranja med razvojem informacijskega sistema: razvijalci opravijo ustrezne spremembe in interna testiranja: testiranje enote in integralno testiranje. Nato pa sistemsko testiranje opravi tudi tehnološki skrbnik aplikacije, ki preveri, ali so dogovorjene spremembe ustrezno implementirane in so moduli ustrezno povezani. Vsako testiranje zahteva tudi v fazi vzdrževanja pripravo ustrezne dokumentacije oz. zapisov.

Testna skupina je bila ponovno pozvana k testiranju. Tehnološki skrbnik je pripravil ustrezen plan testiranja za njihovo delo. Navadno je za takšne spremembe plan testiranja v primerjavi s tistim, ki ga pripravljamo za testiranje med razvojem informacijskega sistema, precej krajši, saj se opišejo samo spremembe, ki morajo biti izvedene, časovni plan za njihovo izvedbo ter udeleženci testiranja. Testna skupina je pregledala plan testiranja in se seznanila s spremembami v aplikaciji. S testiranjem aplikacije je vsak preizkuševalec nadaljeval na svojem delovnem mestu. Komunikacija med testno skupino, tehnološkim skrbnikom in razvijalci je poteka po ustaljenem načinu, podobno kot smo ga opisali v poglavju o izvedbi testiranja v fazi realizacije sistema. Ob zaključku testiranja je vsak preizkuševalec pripravil ustrezne zapise o testiranju, po opravljenem sprejemnem testu pa je bilo pripravljeno še poročilo o testiranju.

V odvisnosti od sprememb v sistemu se tehnološki skrbnik in skrbnik aplikacije odločita, ali je treba ponovno preverjati namestitev sistema oz. ponovno izvesti performančno, stresno in obremenilno testiranje. Te vrste testiranja se v fazi vzdrževanja sistema opravi res samo takrat, ko gre za temeljito spremembo sistema, ki bi lahko vplivala na njegovo delovanje. V našem primeru smo po zaključenem sprejemnem testu opravili samo testiranje namestitve aplikacije na strežnik, saj so bile spremembe, ki smo jih opravili za to verzijo aplikacije, zares manjše in ponovna izvedba performančnega in stresnega testiranja ne bi imela pravega smisla.

8 ANALIZA OPISANEGA PRIMERA IN PREDLOGI ZA IZBOLJŠAVO

Razvoj informacijskih sistemov v banki poteka v skladu z metodologijo razvoja informacijskih sistemov, ki velja v NLB d. d.. V skladu s to metodologijo mora biti izveden tudi celoten postopek testiranja. Večina testiranja se opravi v fazi realizacije računalniške rešitve, za katero so zelo natančno definirani načini testiranja, komunikacija med izvajalci testiranja, tehnološkim skrbnikom in razvijalci ter vsa spremljajoča dokumentacija. Za izvedbo testiranja v ostalih razvojnih fazah pa smo imeli pri organizaciji precej proste roke.

V primerjavi s teoretičnim znanjem, ki sem ga pridobila ob prebiranju literature, bi za izvedbo opisanega primera testiranja lahko navedla kar nekaj pomanjkljivosti. Kljub vsemu pa glede na rezultate in delujoč sistem v produkcijskem okolju lahko rečem, da je bila izvedba in organizacija testiranja tudi v marsičem dobra. Predvsem je pomembno, da

smo si za izvedbo predpisanih testiranj vzeli dovolj časa in se predpisanega plana tudi držali, vsi preizkuševalci so svoje delo vzeli resno, skrbno so pripravili testne primere, veliko napak je bilo odpravljenih že s testiranjem razvijalcev, vsi preizkuševalci pa so pripravljali zahtevane zapise o testiranju, kar so dovolj velika dokazila o tem, da se je testiranje dejansko tudi izvajalo. Izbor preizkuševalcev je bil zelo pester, vključeni so bili strokovnjaki z različnih področij, ki so vsak s svojega vidika pripomogli h kakovosti končnega proizvoda. Glede na izbor testne skupine smo se odločili, da bomo testiranje poskušali izvesti na čim enostavnejši način, saj so bili v skupini tudi preizkuševalci, ki so se s tem srečali prvič. Vse to je od testne skupine in organizatorjev testiranja zahtevalo velikokrat tudi dobršno mero iznajdljivosti in prilagodljivosti. Testiranje aplikacije je bilo opravljeno na natančno določenih t. i. standardnih delovnih postajah, predpisanih po standardih, ki veljajo v banki. Standardna delovna postaja je vsaka delovna postaja v banki, na kateri tečejo poslovne aplikacije, za njihovo delovanje pa so odgovorni skrbniki teh aplikacij. Tako razvijalci vedno natančno vedo, za kakšno konfiguracijo mora biti informacijski sistem razvit. Ob razvoju novih sistemov tako testiranja konfiguracije ne izvajamo ponovno, vendar pa je tako vrsto testiranja treba opraviti za vse sisteme ob vsaki spremembi konfiguracije delovne postaje.

V nadaljevanju bi rada opozorila na nekaj nepravilnosti, o katerih bi v prihodnje veljalo razmisliti, dopolniti postopek testiranja in ga na ta način še izboljšati.

Testna skupina je s testiranjem začela šele v fazi realizacije sistema, takrat smo se tudi natančno dogovorili, kdo vse bo v testiranje vključen. Pri aktivnostih ob zaključku faze zbiranja uporabniških zahtev in faze priprave logičnega in fizičnega modela še ne moremo reči, da je šlo za pravo izvedbo testiranja, pač pa bolj za potrditev narejene dokumentacije s strani uporabnikov oz. predstavnikov uporabnikov. Ker je pri razvoju informacijskih sistemov pomembna tudi časovna komponenta, so se takšne potrditve izvršile bolj kot neka zadostitev ustaljenim predpisanim postopkom. Na srečo nam v fazi realizacije sistema to ni prineslo težav, saj je bil sistem dovolj dobro načrtovan in ni zahteval večjih popravkov v predhodnih fazah. Da pa bi tudi v bodoče preprečili takšne težave in s tem povezane visoke stroške razvoja, bi morali testno skupino formirati povsem na začetku projekta, vzporedno z določanjem ostalih sodelavcev na projektu. Takrat mora testna skupina tudi začeti izvajati testiranje, saj samo na ta način lahko zagotovimo, da so vsi člani enakovredno vključeni in spremljajo potek razvoja sistema od začetka do konca.

Izbira članov testne skupine je bila tako prilagojena potrebam izvedbe uporabniškega testiranja: sestavljali so jo bodoči uporabniki sistema in vpleteni v proces razvoja informacijskega sistema. Lahko rečemo, da v njej praktično ni bilo izkušenih preizkuševalcev programske opreme, kar smo pogrešali ves čas testiranja, še posebno pa se je to pokazalo pri stresnem testiranju. Precej težav smo namreč imeli pri tem, kako ne-informatiku razložiti, zakaj je treba med izvajanjem funkcij v programu npr. simulirati prekinitev mrežne povezave. Po natančni razlagi o smiselnosti takšnega početja smo seveda naleteli na odobravanje, vendar pa lahko trdim, da bi bila učinkovitost takšnega testiranja s prisotnostjo izkušenih preizkuševalcev precej večja. Prav zaradi tega bi morali

v bodoče razmišljati tudi o tem, da bi se nekaj zaposlenih ukvarjalo izključno ali večinoma samo s testiranjem. Tako bi imeli možnost v skupino povabiti ljudi, ki so nabrali že nekaj izkušenj s sodelovanjem pri testiranju na drugih projektih. Testiranje bi tako potekalo hitreje in bolj učinkovito, ob začetku bi se izognili pojasnjevanju, zakaj je tako pomembno, kakšne vrste testiranja bomo uporabili in na kakšen način. V samem postopku testiranja tudi ni zaznati jasne razlike med testiranjem testne skupine po prenosu sistema v testno okolje in uporabniškim testiranjem. Ker so testno skupino sestavljali tudi končni uporabniki sistema, smo združili obe vrsti testiranja v eno, kar pomeni, da so uporabniki poleg potrditve delovanja vgrajenih funkcionalnosti izvajali različne vrste testiranja (stresno, performančno), čeprav za to niso imeli potrebnih izkušenj. Z ločitvijo obeh faz bi tako dosegli, da bi se bodoči uporabniki sistema res lahko osredotočili na potrjevanje, da sistem z vidika funkcionalnosti ustreza njihovemu delu.

Člani testne skupine so se s stališča projekta podpore bančno zavarovalniškim storitvam v formalnem smislu srečevali samo na skupnih sestankih, ki jih je organiziral tehnološki skrbnik. Po njih pa so se vračali vsak na svoje delovno mesto, kjer so tudi nadaljevali s testiranjem. Poleg testiranja so morali izvrševati še svoje običajne redne delovne obveznosti, kar je malenkostno vplivalo tudi na izvedbo testiranja po terminskem planu, saj so morali testiranje večkrat prekiniti in opraviti krajše neodložljive naloge. Na srečo je bilo to upoštevano že pri pripravi plana, tako da je bilo testiranje opravljeno brez večjih časovnih zamikov. Veliko lažje in nemoteno bi ga lahko izvajali, če bi imeli za izvajanje testiranja posebno testno sobo, ki bi bila ločena od delovnih mest in delovnih nalog preizkuševalcev. Člani testne skupine bi si v tem primeru lahko priskočili na pomoč pri testiranju, si izmenjevali izkušnje in skupaj razreševali morebitne težave. V takšnem primeru bi lahko v skupino vključili tudi informatika, ki bi takoj pomagal pri razrešitvi vprašanj tehnične narave.

Pripravljen testni plan je bil izdelan za fazo sprejemnega testiranja in fazo uvedbe sistema. Glede na to, da bi se testna skupina morala vključiti v delo na projektu že na samem začetku, bi morali v testni plan vključiti tudi vsa testiranja, ki jih moramo opraviti v predhodnih fazah: faza zbiranja uporabniških zahtev ter načrtovanje in realizacija sistema. Testiranje, ki ga opravijo razvijalci in tehnološki skrbnik, preden se začne sprejemno testiranje, so prav tako zelo pomemben del testiranja, ki bi ga bilo veljalo v prihodnosti skrbno načrtovati.

V fazi realizacije računalniške rešitve smo poskrbeli tudi za pripravo ustrezne dokumentacije. Procesni skrbnik je skupaj z razvijalci sistema poskrbel za pripravo procesnih navodil in navodil za uporabo aplikacije. Navodila lahko tvorita dva ločena dokumenta, ali pa sta združena, tako kot to velja v našem primeru. Testni skupini so bila pripravljena navodila sicer dostopna, večinoma so se jih posluževali v primeru kakršnihkoli nejasnosti, vendar pa posebno preverjanje skladnosti napisanih navodil s sistemom ni bilo predvideno. Testiranje dokumentacije je prav tako ena od pomembnih nalog, saj dokumentacija, v kateri so napake in neskladnosti s sistemom, lahko povzroči veliko težav uporabnikom. Na podoben način bi v bodoče morali razmisliti tudi o

testiranju pomoči v aplikaciji in pripravljene sistemske dokumentacije, kar je še posebno pomembno z vidika vzdrževanja sistema.

Vsa testiranja so bila opravljena ročno, kar pomeni, da niso bila uporabljena nikakršna orodja za avtomatsko testiranje. Za takšen način smo se odločili, ker v banki še ni povsem ustaljena praksa, da bi za testiranja uporabljali orodja, preizkuševalci pa tudi niso imeli nobenih izkušenj z uporabo takšnih orodij. V bodoče bi bilo smiselno razmisliti o razumni avtomatizaciji testiranja, s katero bi lahko še povečali učinkovitost, kar je še posebno pomembno, ko gre za testiranje večjih kompleksnih sistemov. Izbira bi seveda morala biti stroškovno upravičena ob upoštevanju, da bi na začetku to pomenilo nekaj dodatnih stroškov (npr. nakup orodja, izobraževanje izvajalcev testiranja), na daljši rok pa bi to lahko prineslo zmanjšanje stroškov. Stresno in performančno testiranje, ki smo ga izvedli v našem primeru ročno in z nekaj iznajdljivosti, pri bolj kompleksnih sistemih verjetno ne bi prineslo pričakovanih rezultatov, saj takšnega testiranja praviloma ne moremo izvajati ročno.

S testiranjem smo zaključili po preverjanju namestitve spletne aplikacije, pripraviti pa bi morali še oceno izvedenega testiranja. Na prvi pogled se to zdi le še ena od dodatnih nepomembnih aktivnosti, vendar lahko prinese veliko koristnih informacij za delo v bodoče. Ker se vsi zavedamo, da so tuje napake najcenejše, bi se testna skupina ob koncu testiranja morala sestati, pregledati delo, ki ga je opravila in ga kritično oceniti ter definirati njegove dobre in slabe strani. Še posebno natančno je treba analizirati napake, ki so jih preizkuševalci naredili pri sami izvedbi testiranja in opredeliti vzroke, zakaj je do njih prišlo. Namen takšne analize ni v odkrivanju dobrih ali slabih preizkuševalcev, pač pa izmenjava izkušenj in učenje iz napak, vse seveda z namenom izboljšanja izvedbe postopka testiranja pri razvoju bodočih sistemov.

Ob zaključku bi rada opozorila še na eno pomanjkljivost, na katero so sicer ves čas testiranja opozarjali tudi naši preizkuševalci. Sporočanje odkritih napak, komunikacija med preizkuševalci in tehnološkim skrbnikom ter razvijalci je bila dostikrat zelo zamudna. Z namenom zmanjšanja obremenitve razvijalcev je bilo dogovorjeno, da se vse odkrite napake posredujejo tehnološkemu skrbniku, ta pa zahtevo za njihovo odpravljanje posreduje pravemu razvijalcu. Opisani način prinaša vrsto pomanjkljivosti, že npr. ob odsotnosti tehnološkega skrbnika se je tako odpravljanje napak odložilo. Ker se v banki veliko ukvarjamo z razvojem in testiranjem informacijskih sistemov, bi za bodoča testiranja morali razmisliti o uvedbi sistema za beleženje napak. Danes je na trgu mogoče najti že veliko takšnih sistemov. Bistveno je, da ta sistem nudi možnost vpisa lokacije odkrite napake, opisa napake, opis natančnih korakov, s katerimi smo prišli do napake, možnost vstavljanja datotek (npr. slike vnosnih ekranov), možnost postavitve prioritete za reševanje napak, spremljavo statusa napake in spremljanje statistike (Gunn, 2004). Vpogled v tak sistem imajo lahko preizkuševalci, razvijalci, skrbniki, vodje projekta, managerji, če pa nudi še možnost izdelave različnih poročil, se lahko izognemo zamudni pripravi tedenskih, mesečnih poročil o statusu testiranja oz. statusu projekta. Vodjem pa je

tako na enostaven način omogočeno spremljanje dela in napredovanja testne skupine v skladu s pripravljenim testnim planom.

Postopek testiranja je možno izpeljati na različne načine. Ni natančnih pravil, kako in na kakšen način. Če je končni rezultat razvoja in testiranj sistem, ki brez večjih težav deluje v produkcijskem okolju, potem ne moremo reči, da je bil izbrani način testiranja slab, lahko le rečemo, da je bil z vidika stroškov predrag. Z namenom zmanjšanja stroškov se postopek testiranja da vedno še izboljšati, dopolniti in olajšati delo preizkuševalcev. Nekatere od zgoraj navedenih pomanjkljivosti so takšne, da njihova uvedba ne zahteva veliko priprav in aktivnosti (npr. razširitev testnega plana), druge pa zahtevajo res temeljit premislek in nekoliko dolgotrajnejšo pripravo (npr. izbira orodij za testiranje, uvedba sistema za beleženje napak). Vsekakor bi s pripravo morali začeti že precej pred zagonom novega projekta, saj je v zadnjem hipu, ko bi s testiranjem že morali začeti, pogosto že prepozno, da bi začeli iskati primerno orodje, izpeljali vsa izobraževanja in ostale potrebne aktivnosti.

8.1 Stroški testiranja

Ob zaključku projekta smo pripravili tudi oceno stroškov razvoja informacijskega sistema. Stroški obsegajo samo stroške dela delavcev, ki so sodelovali pri projektu razvoja. Za izračun tako upoštevamo število porabljenih ur za delo na projektu in urno postavko, ki velja za zaposlene v NLB d. d. v odvisnosti od dela, ki ga posameznik opravlja na projektu. Drugih stroškov nismo zabeležili, saj je bila aplikacija razvita s standardnimi orodji, razvijalci pa so programska orodja poznali oz. so z njimi že predhodno razvijali druge informacijske sisteme, prav tako ni bil potreben nakup strežnika ali kakršnokoli izobraževanje.

Tudi stroški testiranja niso obsegali nobenih dodatnih stroškov (npr. nakupa novega testnega orodja, izobraževanje testne skupine,…). Stroške testiranja torej tvorijo izključno stroški dela izvajalcev testiranja. V primerjavi z ostalimi stroški razvoja smo ocenili, da stroški testiranja obsegajo 22% vseh razvojnih stroškov. Od tega skoraj 3% zajemajo stroški priprave plana in izvedbe testiranja enot, ostalo pa so stroški funkcionalnega, regresijskega, stresnega, performančnega, obremenilnega in operacijskega testiranja.

Takšna ocena je v primerjavi s tisto, ki jo najdemo v literaturi, povsem realna, saj je testiranje druga najdražja aktivnost v celotnem procesu razvoja informacijskega sistema. Pri oceni moramo upoštevati, da smo v našem primeru s testiranjem začeli precej pozno in da bi bil delež stroškov še za kakšen odstotek višji, če bi testna skupina začela z delom že v prvih dveh fazah razvoja sistema.

9 ZAKLJUČEK

Cilj razvoja vsakega informacijskega sistema je njegovo pravilno delovanje. Do potrditve tega pa lahko pridemo le preko dobro načrtovane strategije testiranja programske opreme,

kar končnemu izdelku zagotavlja visok nivo kakovosti (Rupnik et al., 1997). Dobro načrtovana strategija pa mora vključevati testiranje oz. preverjanje izdelka v vseh razvojnih fazah. Na splošno za projekte velja, da se dobro planirani projekti zaključijo prej in z manj stroški kot slabo planirani. To je treba upoštevati tudi pri izvedbi testiranja. Do nedavnega je še veljala miselnost, da je testiranje izdelka le ena od faz ob zaključku razvojnega procesa. Ker je projektna skupina na koncu že pošteno pod časovnim pritiskom, hkrati pa verjetno že tudi počasi naveličana dela na projektu, se je testiranju doslej posvečalo le malo pozornosti. Navdano samo toliko, da so se odpravile večje napake, ki jih pred tem razvijalci že niso odpravili. Tako se je zadostilo nekim predpisanim pravilom, odpravljanje manjših, a zato nič manj pomembnih napak, pa je bilo prestavljeno kar v fazo vzdrževanja sistema. V takem primeru gre torej samo za to, da je sistem čimprej predan v delovanje, ne glede na to, ali uporabnikom ustrezno služi ali pa imajo na koncu z njim še več težav kot pri delu s starim sistemom ali ročnimi postopki. Najmanj, kar pri tem sledi, je, da si nakopljemo dobršno mero jeze in pritožb uporabnikov, lahko pa se celo zgodi, da uporabniki ugotovijo, da od njega nimajo praktično nobenih koristi, saj ni izdelan po zahtevah, ki so jih podali. V današnjem poslovnem svetu, kjer se podjetja borijo za svoj delež na tržišču in je konkurenca vsak dan večja, takšen način dela seveda ne zdrži več.

Za uspešno testiranje še zdaleč ni zadosti, da ob zaključku projekta zapremo nekaj ljudi v testno sobo in jim damo za nekaj časa testirati programsko opremo. Izvedba testiranja mora biti skozi vse faze razvoja skrbno načrtovana (Black, 2002). Razmišljati je treba o zniževanju stroškov, kakovosti izdelkov, hitrim odzivom na poslovne zahteve in konkurenčnost. In razvoj informacijskih sistemov pri tem ni nobena izjema. Z uvedbo testiranja v vse razvojne faze informacijskega sistema pa si zagotovimo pravilnost razvoja in skladnost z uporabniškimi zahtevami. Zlato pravilo celotnega razvojnega procesa je, da mora biti enostavno in lahko razumljivo tako izvajalcu testiranja kot tudi drugim udeležencem razvojnega procesa (Zorko, 2004). Bistvenega pomena pri uvajanju testiranja v vse razvojne faze sistema pa je, da sproti poskrbimo za odpravljanje napak, ki so nastale v posamezni fazi in da jih ne prenašamo v naslednje faze, hkrati pa zagotovimo enostavnost izvedbe postopka testiranja, kar potrjuje tudi trditve, ki so navedene v namenih in ciljih.

Seveda pa se takoj ob tem pojavi tudi druga misel: Ali ne povzroča testiranje samo visokih stroškov? Preveč testiranja je torej zločin – premalo pa greh (Perry, 2000). Kako torej postaviti mejo, da stroški razvoja ne bodo naraščali zaradi izvedbe testiranja? Testiranje je drag postopek in verjetno je v celotni izvedbi postopka še najtežja odločitev, kdaj z njim prenehati, saj vemo, da je v sistemu vedno še kakšna napaka. Na dolžino izvedbe testiranja vsekakor vpliva čas in finančna sredstva, ki jih imamo na razpolago. Velika večina napak je odkritih v relativno kratkem času, za odkrivanje še tistih nekaj napak pa navadno na koncu porabimo zelo veliko časa. Poleg tega se pogosto zgodi, da tik pred zaključkom testiranja odkrijemo še kakšno veliko napako. Ne glede na vse to pa se moramo tudi zavedati, da preveč temeljito testiranje povzroča dodatne stroške in nepotrebno angažiranje določenih resursov. Tega managerjem ponavadi ni treba posebej poudarjati,

saj se dobro zavedajo posledic nepotrebnega testiranja. Pri odločitvi verjetno še največ pomenijo izkušnje preizkuševalcev, ki so sodelovali pri testiranju na mnogih projektih. Glede na velikost projekta in ugotovljene napake bodo skupaj s svojimi izkušnjami znali precej dobro svetovati, kdaj je primeren čas za zaključek testiranja.

V nalogi so prikazane tako teoretične osnove kot tudi praktičen primer izvedbe testiranja informacijskega sistema za podporo bančnemu zavarovalništvu, kjer sem opisala način izvedbe testiranja v posameznih razvojnih fazah, v nadaljevanju pa v povezavi s pridobljenim teoretičnim znanjem skušala opozoriti tudi na pomanjkljivosti oz. prikazati možnosti za izboljšanje procesa testiranja. Glede na rezultate (sistem namreč že dalj časa uspešno deluje v produkcijskem okolju) lahko rečemo, da smo bili pri testiranju zelo uspešni. V času delovanja sistema ni bilo opaziti večjih napak, dve manjši smo popravili zaradi sprememb zahtev uporabnikov. Odzivi končnih uporabnikov so pozitivni, pohvalijo predvsem enostavnost uporabe sistema in uvedbo kontrol v njem, ki praktično ne omogočajo napačnih vnosov podatkov, reklamacij in pritožb končnih uporabnikov pa skoraj ne poznamo. Za vse to gre pravzaprav zasluga pazljivemu preverjanju narejenega izdelka. Ves čas testiranja smo tudi pazili na pripravo ustrezne dokumentacije, kar je zahtevala uvedba ISO standarda. Za vse, ki se ukvarjajo z razvojem, je priprava dokumentacije zelo pomembna, saj nekateri naročniki že zahtevajo tudi dokaze o izvedenih testih.

Testiranje je umetnost! Dobro testiranje zahteva kreativnega preizkuševalca z izkušnjami, intuicijo in izbiro pravih tehnik za testiranje. Sama organizacija in izvedba testiranja je precej zahtevno opravilo, saj je pogosto treba združevati nasprotne interese vpletenih. Na eni strani so neučakane končne uporabnike, ki želijo s čim manjšimi stroški (torej tudi malo testiranja) čimprej dobiti svoj izdelek, na drugi strani pa so zahteve po kakovostnem izdelku. V veliko pomoč nam je lahko uvedba metodologije testiranja, s katero je vnaprej določena izvedba posameznega postopka, oblike dokumentov, izdelkov,… Tako ni praktično nobenih dilem, kako posamezen korak izvesti, vsi projekti se testirajo na podoben način in tako lahko pričakujemo tudi enotne rezultate testiranja. Hkrati še pred začetkom s tem lahko seznanimo uporabnike. Kljub neprestanim pritiskom nam nikakor ne sme biti žal časa, ki ga porabimo za verificiranje izdelkov, ki nastajajo v fazi načrtovanja. Če delo ne poteka dosledno in v pravilnem zaporedju z aktivno vključitvijo preizkuševalcev, tvegamo nastanek velike količine napak, ki nas bodo dokončno streznile v zadnjih fazah, ko bi moral biti izdelek že pripravljen.

Ob pripravi na testiranje in kasneje, pri pisanju magistrskega dela, sem prelistala precej literature o tematiki testiranja. Žal skoraj nisem naletela na uporabne praktične primere, ki bi mi pomagali pri pripravi izvedbe testiranja. Splošni opisi so pogosto preveč kompliciran ali pa nerazumljivi in tako neprimerni, sama pa sem želela, da je postopek testiranja enostaven in razumljiv. Tako upam, da bo zapisano v nalogi v pomoč še komu, ki bo stal pred podobnim problemom kot sem sama: pred razčiščevanjem osnovnih pojmov o testiranju, pred raziskovanjem različnih metod in tehnik testiranja, pred samo izvedbo testiranja za konkreten primer. Naveden primer naj služi kot vzorec, kako je

testiranje mogoče izpeljati, seveda pa je možnosti zares veliko. Tudi sami bomo v prihodnje pri postopku testiranja verjetno še marsikaj popravili in spremenili, toda z rezultati smo lahko zadovoljni in s testiranjem bi bilo na tak način smiselno nadaljevati.

V prihodnje bi bilo zanimivo spoznati še kakšno od orodij za testiranje, ki jih je trenutno mogoče dobiti na trgu, jih oceniti in primerjati med seboj ter ugotoviti, katero je primerno za uporabo v različnih razvojnih fazah. Uporabo katerega od njih pa bi lahko prikazali tudi na praktičnem primeru.

10 LITERATURA

1. Alter Steven: Information System: A Management Perspective, 3th Edition. Menlo Park: The Benjamin/Cummings Publishing Company, Inc, 1996. 703 str.
2. Avson E. D., Fitzgerald G.: Information System development: Metodologies, Techniques anh Tools, 2nd Edition. London: The McGraw-Hill Companies, 1995. 505 str.
3. Black Rex: Investing in Software Testing: The Cost of Software Quality. [URL:http://www.rexblackconsulting.com/publications/Investing%20in%20Testing%20(Article%201).pdf], 2000.
4. Black Rex: Critical Software testing Processes. [URL:http://www.rexblackconsulting.com/publications/Introduction%20to%20Critical%20Processes%20(Article).pdf], 2002.
5. Dogša Tomaž: V&V Verifikacija in validacija programske opreme. [URL: http://mafalda.uni-mb.si/backup/2002/predstavitve/testiranje.html], 9.6.2004
6. Drake Thomas: Testing Software Based Systems: The Final Frontier. [URL: http://www.softwaretechnews.com/stn3-3/final-frontier.html], 2004.
7. Gričar Jože: Poslovni informacijski sistem, Študijsko gradivo. [URL: http://ecom.fov.uni-mb.si/Studenti/Predmeti/Gradiva/Gradivo-PIS.pdf], 2002.
8. Gunn Mark: Defect Tracking…a must for all IT project. [URL: http://www.mgdservices.com/November%20Newsletter.doc], 2004.
9. Hajšek Metka: Kako naj informacijski sistem zaživi? Zbornik posvetovanja Dnevi slovenske informatike. Portorož: Slovensko društvo Informatika, 1999. str. 551-558.
10. Ireson William Grant: HANDBOOK of reliability engineering and management. New York: MCGraw-Hill Book Company, 1988.
11. Kaner Cem: Quality Cost Analsis: Benefits and Risks. [URL: http://www.kaner.com/pdfs/Quality_Cost_Analysis.pdf], januar 1996.

12. Kaner Cem, Jack Falk, Hung Quoc Nguyen: Testing Computer Software, 2 nd Edition. New York: Joh Willy & Sons, Inc, 1999. 480 str.

13. Kiger Jack E.: Auditing. Boston, New York: Houghton Mifflin cop., 1997. 918 str.

14. Kovačič Andrej, Vintar Mirko: Načrtovanje in gradnja informacijskih sistemov. Ljubljana: DZS, 1994. 316 str.

15. Križnik Katarina: Sistematično testiranje programske opreme v računalniških podjetjih. Magistrsko delo. Ljubljana: Ekonomska fakulteta, 2002. 79 str.

16. Marick Brain: Classic Testing Mistakes. [URL: http://www.testing.com/writings/classic/mistakes.pdf], 2004.

17. Marick Brian: When Should a test Be Automated. [URL: http://www.testing.com/writings/automate.pdf], 2004a.

18. Martin J. MCClure C.,: Software Maintenance: The problem and It's Solutions. Englewood Cliffs. New York: Prentice Hall, 1983.

19. McGregor John D.: Planning for Testing. [URL: http://www.cs.clemson.edu/~johnmc/joop/col2/column2.html], 15. 1. 2004.

20. Možina Stane, Kavčič Bogdan, Tavčar Mitja, Pučko Danijel, Ivanko Štefan, Lipičnik Bogdan, Gričar Jože, Repovž Leon, Vizjak Andrej, Vahčič Aleš, Rus Veljko, Bohinc Rado: Management. Radovljica: Didakta, 1994. str. 706-740.

21. Myers Glenford J.: The art of software testing. New York: John Wiley and Sons, 1997. 175 str.

22. Nguyen Hung Quoc: Testing Applications on the Web. New York: Joh Willy & Sons, Inc, 2001. 402 str.

23. Pan Jiantao: Software Testing. [URL: http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/index.html], 1999.

24. Perry William E.: Effective Methods for Software Testing, 2 nd Edition. New York: Joh Willy & Sons, Inc, 2000. 812 str.

25. Rothman Jahanna: No More Second-Class Testers! Better Software. [URL: http://www.stickyminds.com/BetterSoftware/magazine.sap?fn=cifea], 2003.

26. Rupnik Rok, Bajec Marko, Krisper Marjan: Metoda medfaznega testiranja pri razvoju informacijskih sistemov. Zbornik posvetovanja Dnevi slovenske informatike. Portorož: Slovensko društvo Informatika, 1997. str. 50-58.

27. Shelly Gary B., Cashmann Thomas J., Rosenblatt Harry J.: Systems Analysis and Design, 3th Edition. Cambridge [etc.]: Course Technology, 1998.

28. Shelly Gary B., Cashmann Thomas J., Rosenblatt Harry J.: Systems Analysis and Design, 4th Edition. Boston: Course Technology, 2001.

29. Slovensko društvo informatika: Islovar, Slovar informatike, poskusni snopič. Ljubljana: Slovensko društvo informatika, 2004.

30. Solina Franc: Projektno vodenje razvoja programske opreme, 1. izdaja. Ljubljana: Fakulteta za računalništvo in informatiko, 1997. 212 str.

31. Stare Aljaž: Priprava in izvedba projekta. Ljubljana: Agencija Poti, d. o. o., 2001. 36 str.

32. Zorko Samo R.: Lovci na žuželke. Moj Mikro, Marec 2004.

33. Zupan Blaž: Preizkušanje programskih proizvodov. [URL:http://magix.fri.uni-lj.si/predavanja/skis/slides/skis12-PreskusanjeProgramskihProizvodov.ppt], 2004.

34. Whittaker A. James: What Is Software Testing? And Why Is It So Hard? IEEE Software, January/February 2000. str. 70-79.

11 VIRI

1. ANSI/IEEE Std 792-1983.

2. Interni standard o konstrukcij intranet aplikacij, Interno gradivo. Ljubljana: Nova Ljubljanska banka, maj 2003.

3. Kaner Cem: Avoiding Shelfware: A Manager's View of Automated GUI Testing. ZDA, 1998.

4. Malus Marta, Razvojno življenjski ciklus informacijskega sistema, [URL: http://krota.pti.fgg.uni-lj.si/edu/tius/seminarji/2003/Malus/Razvojno%20%C5%BEivljenjski%20ciklus%20IS.doc], 2004.

5. Metodologija razvoja in sprememb informacijskih sistemov v NLB, 5. izdaja, Interno gradivo. Ljubljana: Nova Ljubljanska banka d.d., julij 2004.

6. Navodilo za testiranje IT rešitve, 3. izdaja. Interno gradivo. Ljubljana: Nova Ljubljanska banka d.d., julij 2004.

7. Nova Ljubljanska banka, [URL: www.nlb.si], 2005.

8. Sistem kakovosti v UCIT – osebni priročnik. Interno gradivo. Ljubljana: Nova Ljubljanska banka d. d., september 2004.

9. Zagonska koncepcija projekta Bančno zavarovalništvo.Interno gradivo. Ljubljana: Nova Ljubljanska banka d.d., julij 2003.

10. Življenjski cikel programske opreme, [URL:http://lrv.fri.unilj.si/~franc/VP/prosojnice_VP.pdf], 2004

12 KAZALO SLIK

13 KAZALO TABEL

14 SLOVAR IZRAZOV

Tuj izraz	**Slovenski prevod**
audit trail	sledljivost sprememb podatkov
aplikacija	računalniški program, ki je namenjen uporabniku za opravljanje določenih nalog
black box	črna škatla
cost/benefit analysis	analiza stroškov in koristi: primerjava neto koristi med različnimi možnimi rešitvami pri razvoju informacijskih sistemov
definicija	opredelitev, določitev
explain	SQL ukaz v DB2 podatkovni bazi, ki se uporablja za podrobnejšo razlago delovanja določene poizvedbe.
glass-box	prozorna škatla
identificirati	ugotoviti
inicializacija	postavitev na začetno vrednost
integracija	povezovanje posameznih enot v večjo celoto
integralno testiranje	celotno testiranje proizvoda, vseh njegovih modulov in enot
integration testing, link testing	integralno testiranje
kompleksen	zapleten, raznovrsten
link testing, integration testing	preverjanje medsebojnega delovanja modulov
manager	vodilni delavec
numeričen	nanašajoč se na številke
planiranje	načrtovanje
peer review	pregled računalniškega programa z namenom odpravljanja sintaktičnih in logičnih napak, ki ga opravi druga oseba
return of investment	donosnost naložbe
Rapid Application Development (RAD)	hiter razvoj aplikacij – koncept, na podlagi katerega je mogoče hitreje razvijati informacijske sisteme
structured walkthrough, code review	pregled računalniškega programa z namenom odpravljanja sintaktičnih in logičnih napak, ki ga opravijo razvijalci
system functional techniques	funkcionalne tehnike testiranja
system structural testing techniques	strukturne tehnike testiranja
system test	sistemski test
system testing	sistemsko testiranje
team	skupina ljudi, ki opravlja skupno delo
unit testing	testiranje enote
unit test technique	tehnike testiranja enot
vmesnik	računalniški program za medsebojno povezovanje dveh sistemov
waterfall model	zaporedni ali kaskadni razvojni model, ki po svoji obliki spominja na slapove
white box	bela škatla

15 Priloge

Priloga 1: Plan testiranja

Priloga 2: Poročilo o testiranju

Priloga 3: Poročilo o uporabniškem sprejemnem testu

Priloga 1: Plan testiranja

PLAN TESTIRANJA

INFORMACIJSKI SISTEM ZA PODPORO BANČNEMU ZAVAROVALNIŠTVU – APLIKACIJA EVITA

1 KAZALO

2 SPLOŠNE INFORMACIJE

Produkt – program in oznaka verzije: **Evita 01.01.00**

Ime in oznaka naloge: **Bančno zavarovalništvo – Evita R040106/500_1**

Aplikacija Evita je spletna aplikacija, ki se bo v banki uporabljala za sklepanje življenjskih zavarovanj za zavarovalnico NLB VITA d. d.. Zaključene ponudbe, ki se bodo v podatkovno bazo vpisane preko spletne aplikacije, bodo enkrat dnevno s paketnimi obdelavami prenesene na zavarovalnico v datoteki, ki bo odložena na dogovorjeno mesto na strežniku. Podatki iz datoteke bodo preneseni v informacijski sistem IGAS, ki bo v nadaljevanju skrbel za vodenje življenjskih zavarovanj.

Aplikacija je namenjena komercialistom v poslovalnicah, ki dobo tržili zavarovalniške storitve, vodjem poslovalnic, direktorjem podružnic in delavcem v Sektorju za marketing za spremljanje prodaje v poslovni mreži, delavcem na NLB Viti d. d. za potrjevanje/zavračanje informativnih ponudb za storitev življenjsko zavarovanje kreditojemalcev in tehnološkemu skrbniku aplikacije za nastavitev osnovnih parametrov za delovanje aplikacije.

2.1 Sistem pooblastil

Sistem pooblastil v aplikaciji omogoča različen obseg dela v aplikaciji. V aplikacijo so vgrajena pooblastila:

Naziv pooblastila	**Predviden nosilec pooblastila**	**Opis pooblastila**	**Opombe**
INFORMAT	Komercialist brez licence, ki bo izdeloval informativne izračune, vendar ne bo sklepal ponudb	vnos, izpis in pregledovanje informativnih izračunov	
SKRBNIK	tehnološki skrbnik aplikacije	ažuriranje šifrantov, parametrov, vrednosti skladov in ostalih temeljnih podatkov	
AGENT	Komercialist z licenco, ki bo sklepal ponudbe	vnos in pregledovanje posameznih ponudb in simulacij	lahko pregleduje vse ponudbe in ni omejen le na svojo poslovalnico
VITA	zaposleni na NLB VITA d. d.	pregledovanje in potrjevanje ali zavrnitev ponudb za storitve: življenjsko zavarovanje kreditojemalcev	
POSLOVAL	vodje poslovalnic	pregled poročil o sklenjenih ponudbah na nivoju poslovalnice za katero je pooblastilo dodeljeno	

PODRUZNI	direktorji podružnic	pregled poročil o sklenjenih ponudbah na nivoju podružnice za katero je pooblastilo dodeljeno	
BANKA	Sekto za marketing, Sektor za spremljavo poslovanja s fizičnimi osebami	pregled poročil o sklenjenih ponudbah na nivoju banke	

2.2 Funkcionalnosti sistema

Aplikacija Evita omogoča sklepanje ponudb za storitve:

- NLB Varčevanje Vita plus (VVP),
- NLB Naložba Vita plus (NVP),
- NLB Naložba Vita X (NVx),
- Življenjsko zavarovanje kreditojemalcev (ŽZK).

Aplikacija Evita bo podpirala naslednje funkcionalnosti (v oklepaju so navedena pooblastila, s katerimi lahko navedeno funkcionalnost testiramo):

Informativni izračun:

- izdelava in izpis informativnega izračuna za vse zgoraj navedene storitve (INFORMAT, AGENT),
- pregledovanje že izdelanega informativnega izračuna (INFORMAT, AGENT).

Informativna ponudba:

- izdelava in izpis informativne ponudbe za storitev Življenjsko zavarovanje kreditojemalcev (AGENT),
- pregledovanje informativne ponudbe (AGENT),
- popravljanje informativne ponudbe (AGENT),
- potrjevanje informativne ponudbe (VITA),
- zavračanje informativne ponudbe (VITA).

Ponudba:

- priprava in izpis ponudbe za zgoraj navedene storitve (AGENT),
- pregledovanje ponudbe (AGENT),
- popravljanje ponudbe (AGENT),
- zaključevanje ponudbe (AGENT).

Polica:

- pregledovanje podatkov o polici (AGENT).

Poročila:

- izdelava poročil na nivoju poslovalnice (POSLOVAL),
- izdelava poročil na nivoju podružnice (PODRUZNI),
- izdelava poročil na nivoju banke (BANKA),
- izdelava dnevnega zaključka v okviru organizacijske enote (AGENT).

Skrbniški modul:

- spreminjanje šifrantov (Zdravstvena vprašanja, Dokumenti, Skladi, Drugi šifranti) (SKRBNIK),
- tablice za določanje nevarnostnih premij (SKRBNIK).

2.3 Vrsta in način testiranja

Pri testiranju aplikacije Evita se bo izvajalo funkcionalno, regresijsko, stresno, performančno in operacijsko testiranje. Podrobnejša navodila o tem, kako se izvaja posamezna vrsta testiranja, se nahajajo v Navodilih za testiranje IT rešitve.

Za testiranje se ne uporablja orodij za testiranje, kar pomeni, da bomo vse vrste testiranja izvajali ročno. Postopek testiranja se izvaja po vnaprej pripravljenih tesnih primerih, ki jih pripravijo preizkuševalci sami. Po vnosu testnega primera preizkuševalec pregleda dobljene rezultate in jih primerja s pričakovanimi rezultati. Če pride do odstopanja, ugotovitve le-te zapiše in jih pošlje tehnološkemu skrbniku, sicer pa pripravi zapis o testiranju, iz katerega je razvidno, da je delovanje programa (ali dela programa) pravilno.

Testiranje se izvaja izključno na standardnih delovnih postajah. Opis standardnih delovnih postaj je možno najti v Standardih za PC opremo v NLB d. d.

2.4 Cilji in pričakovani rezultati testiranja

Cilj testiranja je:

- preveriti pravilnost delovanja sistema glede na uporabniške zahteve in zagotoviti, da bo v sistemu ob predaji kar najmanj napak,
- preveriti prisotnost in pravilnost delovanja kontrol,
- preveriti delovanje sistema ob različnih prekinitvah.

Pričakovani rezultati testiranja so:

- izdelave informativnega izračuna za vse storitve,
- izpis informativnega izračuna na tiskalnik,
- identifikacija komitenta v registru komitentov,
- sklenitev ponudbe,
- sklenitev informativne ponudbe,

- potrditev ali zavrnitev ponudbe za storitev Življenjsko zavarovanje kreditojemalcev,
- zaključevanje ponudbe,
- izdelava poročil na različnih nivojih,
- izdelava dnevnega zaključka,
- priprava podatkov za pošiljanje v informacijski sistem IGAS.

2.5 Referenčni dokumenti

Izvajalcem testiranja so na voljo naslednji dokumenti:

- procesna navodila za bančno zavarovalništvo,
- navodila za delo z aplikacijo Evita,
- navodilo za testiranje IT rešitev,
- standardi za PC opremo.

3 ORGANIZACIJA TESTIRANJA

Testno skupino sestavljajo:

- preizkuševalec 1 – Oddelek za razvoj procesne org. in stand. poslov,
- preizkuševalec 2 – Oddelek za uvajanje tehnoloških sprememb,
- preizkuševalec 3 – NLB Vita d. d.,
- preizkuševalec 4 - NLB Vita d. d.,
- preizkuševalec 5 - NLB Vita d. d.,
- preizkuševalec 6 - NLB Vita d. d.,
- preizkuševalec 7 – Oddelek za podporo prodaji prebivalstvu,
- preizkuševalec 8 - Oddelek za podporo prodaji prebivalstvu,
- preizkuševalec 9 - komercialist,
- preizkuševalec 10 – komercialist - mentor,
- preizkuševalec 11 - komercialist,
- preizkuševalec 12 - NLB Vita d. d.,
- preizkuševalec 13- NLB Vita d. d.,
- tehnološki skrbnik,
- skrbnik strežnika,
- administrator podatkovne baze,
- 2 skupini tečajnikov.

3.1 Pravila, postopki in dogovori

Plan testiranja se nanaša na aplikacijo Evita. Preverjanje pravilnosti pripravljene datoteke s podatki o sklenjeni policah se preverja v drugem sistemu (IS IGAS), kar pa ni predmet tega plana testiranja.

Podatke za testiranje pripravijo udeleženci testiranja sami. Pripravljeni morajo biti v skladu s priporočili za pripravo testnih podatkov, ki so zapisani v nadaljevanju. Testirajo se vse mejne vrednosti: najnižja in najvišja vrednost, vrednosti zunaj določenega intervala in vrednosti znotraj intervala. Pri kreiranju testnih primerov ne smemo pozabiti na napačne vnose v okviru ekvivalentnega razreda. Testni primer vsebuje naslednje podatke:

- datum testiranja,
- ime in priimek izvajalca testiranja,
- ime in verzijo programskega proizvoda, ki se testira,
- opis testnega primera,
- vhodne podatke,
- pričakovane rezultate,
- dejanske rezultate.

Izvajalci testiranja ugotovljene napake pošiljajo uporabniškemu skrbniku po elektronski pošti na obrazcu za sporočanje napak, ki se nahaja v Prilogi 1. Na obrazcu naj bo navedeno:

- pri katerem postopku je prišlo do napake,
- URL naslov strani, kjer je prišlo do napake,
- vhodni podatki,
- dobljeni rezultati,
- pričakovani rezultati.

Če se pri testiranju ugotovijo večje napake, ki onemogočajo normalen nadaljnji potek testiranja, se testiranje prekine, odpravi napake in ponovi. Manjše napake, ki ne vplivajo na nemoten potek testiranja, se zapišejo in posredujejo razvijalcem, da jih bodo odpravili. Po odpravi napak je testiranje treba ponoviti.

Pred začetkom vnosa podatkov v spletno aplikacijo je treba preveriti, ali so ustrezni podatki o komitentih vpisani v testnem Registru komitentov (za bančne številke, ki jih bomo uporabljali za testiranje).

V prvem delu se bo izvajalo samo testiranje spletne aplikacije. Pred začetkom testiranja paketnih obdelav je treba zagotoviti, da se bodo podatki iz spletne aplikacije pravilno formirali, saj ne bi imelo smisla testirati pripravo datoteke na napačnih podatkih. Ko bo odpravljena že večina večjih napak, pa bomo ob koncu vsakodnevnega testiranja pripravili tudi datoteko s paketnimi obdelavami, da bodo lahko začeli tudi s testiranjem prevzema podatkov v informacijski sistem IGAS.

3.2 Pogoji za začetek testiranja

Pred začetkom testiranja je treba preveriti, ali:

- so vse storitve, ki jih testiramo, z vsemi parametri vpisane v Katalogu storitev,
- so napolnjeni vsi potrebni šifranti,
- je vnesena ustrezna organizacijska struktura banke,
- so za pripravljene testne primere vneseni ustrezni podatki v Register komitentov in
- imajo vsi preizkuševalci dodeljena ustrezna pooblastila za testiranje.

3.3 Vodenje zapisov o testiranju

Med izvajanjem testiranja morajo vsi preizkuševalci pripraviti zapise o testiranju. Za vse testne podatke se naredijo možni izpisi (izpis simulacije, izpis ponudbe, izpis BN01 obrazca, poročila…), ki se bodo shranili kot del projektne dokumentacije.

3.4 Priprava poročila o testiranju

Po zaključku testiranja se pripravi Poročilo o testiranju in podpiše Poročilo o sprejemnem testu.

4 PRIPOROČILA ZA PRIPRAVO TESTNIH PRIMEROV

Testni primeri so simulacija poslovnih dogodkov v praksi, so simulacija resnične aktivnosti, zato morajo biti vsi podatki in izračuni enaki, kot bodo v resničnem življenju. Pripravljeni morajo biti v skladu z lastnostmi testirane storitve in prilagojeni pogojem, ki jih določa testiranje.

4.1 Ekvivalentni razredi

4.1.1 Informativni izračun

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Vgrajene kontrole**
Zavarovalna storitev	padajoči meni	- Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev	ni izbrana nobena od možnosti	
Ime	vnosno polje	vse črke	vsi numerični znaki	
Priimek	vnosno polje	vse črke	vsi numerični znaki	
Telefonska številka	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	

Spol	izbirni gumb	- moški, - ženske	ni izbrana nobena od možnosti	
Datum rojstva	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki,razen pike	Preverjanje oblike vpisanega datuma - DD.MM.LLLL
Frekvenca plačila	izbirni gumb	- mesečno, - četrtletno, - polletno, - letno, - enkratna premija	ni izbrana nobena od možnosti	
Dodatni vprašalnik	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Znesek premije	vnosno polje	- numerični znaki, stotine ločene z vejico - > = znesek glede na izbrano frekvenco plačila	- črke in ostali ne-numerični znaki, - znesek, ki je manjši od minimalnega zneska za izbrano frekvenco plačila	- preverja minimalno premijo za izbrano frekvenco plačila: ▪ mesečno = 50 €, ▪ četrtletno = 150 €, ▪ polletno = 300 €, ▪ letno = 600 €, ▪ enkratno = 1000 €
Kritje za smrt	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Zavarovalna vsota – kritje za smrt	vnosno polje	- vsa števila, stotine ločene z vejico, - vpis standardne zavarovalne vsote (določi se programsko), - znesek >= 1000 € (enkratno plačilo), - znesek >= 1200 € (obročno plačilo)	- črke in ostali ne-numerični znaki, razen decimalne vejice, - znesek , ki je < od minimalne zavarovalne vsote, - znesek, ki je > večji od maximalne zavarovalne vsote	- preverjanje minimalne zavarovalne vsote glede na izbrano frekvenco plačila - preverjanje maksimalne zavarovalne vsote glede na starost zavarovanca in dodatni vprašalnik: ▪ starost < 50 let, brez dodatnega vprašalnika: 16.000 € ▪ starost < 50 let, z dodatnim vprašalnikom: 36.000 € ▪ starost > = 50 let, brez dodatnega vprašalnika: 8.000 € ▪ starost >= 50 let, z dodatnim vprašalnikom: 18.000 €
Kritje za nezgodno smrt	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Zavarovalna vsota - kritje za	vnosno polje	ni možnosti vnosa, znesek vpiše aplikacije, če je vključeno stikalo Kritje za		

nezgodno smrt		nezg.smrt (2-kratnik ZV za osnovno kritje)		
Kritje za kritične bolezni	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Zavarovalna vsota – kritje za kritične bolezni	vnosno polje	ni možnosti vnosa, znesek vpiše aplikacije, če je vključeno stikalo Kritje za kritične bolezni (polovična vrednost ZV za osnovno kritje)	/	
Znesek dodatne premije	vnosno polje	- numerični znaki, stotine ločene z vejico, - znesek >= 500 €	- črke in ostali ne-numerični znaki, razen decimalne vejice, znesek < 500 €	minimalna dodatna premija = 500 €
Profil zavarovalca – odgovori	izbirni gumb	- portfelj1, - portfelj2, - portfelj3, - portfelj4	ni izbrana nobena od možnosti	
Izbira sklada	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Delež sklada	vnosno polje	cela števila od 1–100 (v %)	- 0 in negativna števila, - črke in ostali ne-numerični znaki, - števila > 100	
Obdobje za prikaz	vnosno polje	numerični znaki od 1- 39	- 0, - števila > 39, - negativna števila, - črke, ostali ne-numerični znaki	preverjanje, da vpisano število ni večje od 39

4.1.2 Ponudba

Naziv polja	**Tip polja**	**<u>Veljavni</u> ekvivalentni razred**	**<u>Neveljavni</u> ekvivalentni razred**	**Vgrajene kontrole**
Zavarovalna storitev	padajoči meni	- Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev	ni izbrana nobena od možnosti	
Zavarovanec	iskanje po Registru komitentov	- komitenti iz Registra komitentov, - komitent mora biti ob začetku zavarovanja star vsaj 14 let (>=),	- ni vpisan v Register komitentov, - zavarovanec mlajši od 14 let,	preverjanje starosti zavarovanca
Zavarovalec	iskanje po Registru komitentov	- komitenti iz Registra komitentov, - komitent mora biti ob začetku zavarovanja star vsaj 18 let (>=)	- ni vpisan v Register komitentov, - zavarovalec mlajši od 18 let	preverjanje starosti zavarovalca
Frekvenca	izbirni	- mesečno,	ni izbrana nobena od	

plačila	gumb	- četrtletno, - polletno, - letno, - enkratna premija	možnosti	
Dodatni vprašalnik	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Znesek premije	vnosno polje	- numerični znaki, stotine ločene z vejico - > = znesek glede na izbrano frekvenco plačila	- črke in ostali ne-numerični znaki, - znesek, ki je manjši od minimalnega zneska za izbrano frekvenco plačila	- preverja minimalno premijo za izbrano frekvenco plačila: ▪ mesečno = 50 €, ▪ četrtletno = 150 €, ▪ polletno = 300 €, ▪ letno = 600 €, ▪ enkratno = 1000 €
Znesek dodatne premije	vnosno polje	- numerični znaki, stotine ločene z vejico, - znesek >= 500 €	- črke in ostali ne-numerični znaki, razen decimalne vejice, - znesek < 500 €,	preverja se znesek minimalne dodatne premije = 500 €
Indeksacija	izbirni gumb	- ni povišanja, - premija, - premija in zavarovalna vsota	ni izbrana nobena od možnosti	
Način plačila obročnih premij	izbirni gumb	- direktna obremenitev, - izstavitev računa pravni osebi (če je zavarovalec pravna oseba)	ni izbrana nobena od možnosti	
Osebni račun	vnosno polje	numerični znaki v treh sklopih (šifra finančne institucije, organizacijka enota, partija+kontrolna številka)	črke in ostali ne-numerični znaki	preverja se vpis kontrolne številke po ustreznem modulu
Dan plačila	izbirni gumb	- 8. v mesecu, - 18. v mesecu, - 28. v mesecu	ni izbrana nobena od možnosti	
Datum začetka zavarovanja	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Zavarovalna vsota za smrt	vnosno polje	- vsa števila, stotine ločene z vejico, - vpis standardne zavarovalne vsote (določi se programsko), - znesek >= 1000 € (enkratno plačilo), - znesek >= 1200 € (obročno plačilo)	- črke in ostali ne-numerični znaki, razen decimalne vejice, - znesek, ki je < od minimalne zavarovalne vsote, - znesek, ki je > večji od maximalne zavarovalne vsote	- preverjanje minimalne zavarovalne vsote glede na izbrano frekvenco plačila, - preverjanje maksimalne zav. vsote glede na starost zavarovanca in dodatni vprašalnik: ▪ starost < 50 let, brez dodatnega vprašalnika: 16.000 € ▪ starost < 50 let, z dodatnim vprašalnikom:

				36.000 € ▪ starost > = 50 let, brez dodatnega vprašalnika: 8.000 € ▪ starost >= 50 let, z dodatnim vprašalnikom: 18.000 €
Kritje za nezgodno smrt	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Zavarovalna vsota za nezgodno smrt	vnosno polje	ni vnosa, znesek vpiše aplikacije, če je izbrano stikalo Kritje za nezgodno smrt (2-kratnik ZV za osnovno kritje)	/	
Kritje za kritične bolezni	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Zavarovalna vsota – kritje za kritične bolezni	vnosno polje	ni vnosa, znesek vpiše aplikacije, če je izbrano stikalo Kritje za kritične bolezni (polovična vrednost ZV za osnovno kritje)	/	
Izbira sklada	stikalo	- da, - ne	ni izbrana nobena od možnosti	
Delež sklada	vnosno polje	cela števila od 1–100	- 0 in negativna števila, - črke in ostali ne-numerični znaki, - števila > 100	
Upravičenci v primeru smrti	izbirni gumb	- klavzula 1, - zavarovalec, - otroci zavarovalca, - vnuki zavarovalca, - zakonec zavarovalca, - dediči zavarovalca, - klavzula 2, - drugo (fizične osebe)	- ni izbrana nobena od možnosti, - ni vpisa podatkov o fizični osebi, če je izbrana opcija »drugo«	
Delež v odstotkih	vnosno polje	- cela števila od 1–100	- 0 in negativna števila, - črke in ostali ne-numerični znaki, - števila > 100	
Upravičenci v ostalih primerih	izbirni gumb	- klavzula 3, - zavarovalec, - zavarovanec, - otroci zavarovalca, - vnuki zavarovalca, - zakonec zavarovalca, - dediči zavarovalca, - drugo (fizične osebe)	- ni izbrana nobena od možnosti, - ni vpisa podatkov o fizični osebi, če je izbrana opcija »drugo«	

Odgovori na zdravstvena vprašanja	izbirni gumb	- da, - ne	ni izbrana nobena od možnosti	
Osebni zdravnik	vnosno polje	vsi alfanumerični znaki	/	
Matična številka usmerjevalca	vnosno polje/iskanje po registru zaposlenih	- vse matične številke iz registra zaposlenih, - numerični znaki	črke in ostali ne-numerični znaki	
Iskanje po registru zaposlenih:				
Priimek	vnosno polje	vse črke	numerični znaki	
Ime	vnosno polje	vse črke	numerični znaki	
Profil zavarovalca – odgovori	izbirni gumb	- portfelj1, - portfelj2, - portfelj3, - portfelj4	ni izbrana nobena od možnosti	
Iskanje po registru komitentov:				
Bančna številka komitenta	vnosno polje	- numerični znaki, - zapisi iz Registra komitentov	črke in ostali ne-numerični znaki	
Ime	vnosno polje	vse črke	numerični znaki	
Priimek	vnosno polje	vse črke	numerični znaki	
EMŠO	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	
Davčna številka	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	

4.1.3 Pregledovanje ponudb

Naziv polja	**Tip polja**	**<u>Veljavni</u> ekvivalentni razred**	**<u>Neveljavni</u> ekvivalentni razred**	**Vgrajene kontrole**
Zavarovalna storitev	padajoči meni	- Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev	ni izbrana nobena od možnosti	

Številka komitenta	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	
Številka ponudbe	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	
Priimek zavarovanca	vnosno polje	vse črke	numerični znaki	
Ime zavarovanca	vnosno polje	vse črke	numerični znaki	
Datum veljavnosti	vnosno polje	numerični znaki,	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum podpisa	vnosno polje	numerični znaki,	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum zadnje spremembe	vnosno polje	numerični znaki,	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Status ponudbe	padajoči meni	- vsi statusi, - v pripravi, - podpisana, - sklenjena, - potrjena, - zavrnjena	ni izbrana nobena od možnosti	

4.1.4 Popravljanje ponudb

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Vgrajene kontrole**
Zavarovalna storitev	padajoči meni	- Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev	ni izbrana nobena od možnosti	ni izbrana nobena od možnosti
Številka komitenta	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	
Številka ponudbe	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	
Priimek zavarovanca	vnosno polje	vse črke	numerični znaki	
Ime zavarovanca	vnosno polje	vse črke	numerični znaki	

4.1.5 Zaključevanje ponudb

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Vgrajene kontrole**
Zavarovalna storitev	padajoči meni	- Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev	ni izbrana nobena od možnosti	
Številka komitenta	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	
Številka ponudbe	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki	
Priimek zavarovanca	vnosno polje	vse črke	numerični znaki	
Ime zavarovanca	vnosno polje	vse črke	numerični znaki	

4.1.6 Pregled na nivoju poslovalnice

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Vgrajene kontrole**
Poslovalnica	padajoči meni	vse OE, za katere ima uporabnik pooblastila	ni izbrana nobena od možnosti	v padajočem meniju so prikazane samo OE, za katere ima uporabnik pooblastila za pregledovanje
Datum veljavnosti	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum podpisa	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum zadnje spremembe	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Storitev	padajoči meni	- vse storitve, - Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev, - NLB Naložba Vita 1, - NLB Naložba Vita 2, - NLB Naložba Vita 3, - NLB Naložba Vita 4, - NLB Naložba Vita 5,	ni izbrana nobena od možnosti	

		- NLB Naložba Vita 6		
Status storitve	padajoči meni	- vsi statusi, - v pripravi, - podpisana, - sklenjena, - potrjena, - zavrnjena	ni izbrana nobena od možnosti	
Tip pregleda	izbirni gumb	- posamezno, - sumarno	ni izbrana nobena od možnosti	

4.1.7 Pregled na nivoju podružnice

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Vgrajene kontrole**
Podružnica	padajoči meni	vse OE, za katere ima uporabnik pooblastila	ni izbrana nobena od možnosti	v padajočem meniju so prikazane samo OE, za katere ima uporabnik pooblastila za pregledovanje
Datum veljavnosti	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum podpisa	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum zadnje spremembe	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Storitev	padajoči meni	- vse storitve, - Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev, - NLB Naložba Vita 1, - NLB Naložba Vita 2, - NLB Naložba Vita 3, - NLB Naložba Vita 4, - NLB Naložba Vita 5, - NLB Naložba Vita 6	ni izbrana nobena od možnosti	
Status storitve	padajoči meni	- vsi statusi, - v pripravi, - podpisana, - sklenjena - potrjena, - zavrnjena	ni izbrana nobena od možnosti	

4.1.8 Pregled na nivoju banke

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Vgrajene kontrole**
Datum veljavnosti	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum podpisa	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Datum zadnje spremembe	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL
Storitev	padajoči meni	- vse storitve, - Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev, - NLB Naložba Vita 1, - NLB Naložba Vita 2, - NLB Naložba Vita 3, - NLB Naložba Vita 4, - NLB Naložba Vita 5, - NLB Naložba Vita 6	ni izbrana nobena od možnosti	
Status storitve	padajoči meni	- vsi statusi, - v pripravi, - podpisana, - sklenjena, - potrjena, - zavrnjena	ni izbrana nobena od možnosti	

4.1.9 Seznam ponudb za dnevni zaključek

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Kontrole**
Datum zajema sklenjenih ponudb	vnosno polje	numerični znaki	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL

4.1.10 Pregled vrednosti točk skladov

Naziv polja	**Tip polja**	**Veljavni ekvivalentni razred**	**Neveljavni ekvivalentni razred**	**Kontrole**
Zavarovalna storitev	padajoči meni	- Varčevanje Vita plus, - Naložba Vita plus, - Življenjsko zavarovanje kreditojemalcev, - NLB Naložba Vita 1,	ni izbrana nobena od možnosti	

		- NLB Naložba Vita 2, - NLB Naložba Vita 3, - NLB Naložba Vita 4, - NLB Naložba Vita 5, - NLB Naložba Vita 6		
Ime sklada	padajoči meni	- KBC Life Invest Fund Defensive, Class B, - KBC Life Invest Fund Dynamic, Class B, - KBC Life Invest Fund Neutral, Class B, - Garanteed Fund , - KBC Fund Security NLB Vita Stock selection	ni izbrana nobena od možnosti	preverja izbrani sklad glede na izbrano storitev
Vrednost sklada od datuma	vnosno polje	numerični znaki,	črke in ostali ne-numerični znaki, razen pike	preverjanje oblike vpisanega datuma - DD.MM.LLLL

5 ČASOVNI PLAN TESTIRANJA

5.1 Časovni plan testiranja – po vrsti testiranja

Vrsta testiranja	**Termin**	**Izvajalci**
Funkcionalno testiranje Regresijsko testiranje	15. 9. – 22. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Stresno testiranje	25. 10. - 29. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13, administrator podatkovne baze, Oddelek za omrežja
Performančno testiranje	25. 10. – 29. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13, razvijalci
Obremenilno testiranje	1. 11. - 3. 11. 2003	udeleženci izobraževanja (2 skupini), razvijalci, skrbnik strežnika, kjer je

	8. 11. -10. 11. 2003	nameščen sistem
Operacijsko testiranje	14. 11. 2003	razvijalci, skrbnik strežnika, kjer je nameščen sistem

5.1.1 Podrobni časovni plan za funkcionalno testiranje

Aktivnost	**Termin**	**Izvajalec**
Nastavitve	15. 9. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Pooblastila v sistemu Evita	15. 9. - 20. 9. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11
Informativni izračuni (za vse storitve)	17. 9. - 24. 9. 2003	preizkuševalec 1, preizkuševalec 4, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11,
Izpis informativnega izračuna	17. 9. - 24. 9. 2003	preizkuševalec 4, preizkuševalec 7, preizkuševalec 8
Iskanje informativnih izračunov po različnih kriterijih	17. 9. - 24. 9. 2003	preizkuševalec 4, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11,
Vnos informativne ponudbe (za storitev življenjsko zavarovanje kreditojemalcev)	27. 9. – 30. 9. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Izpis informativne ponudbe (za storitev življenjsko zavarovanje kreditojemalcev)	27. 9. – 30. 9. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Potrjevanje, zavrnitev informativne ponudbe	27. 9. – 30. 9. 2003	preizkuševalec 3, preizkuševalec 4, preizkuševalec 12, preizkuševalec 13
Pregledovanje in popravljanje informativnih	30. 9. – 1. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4,

ponudb		preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Vnos ponudbe	4. 10. – 7. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Izpis ponudbe	4. 10. – 7. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Popravljanje podatkov na ponudbi	4. 10. – 7. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Pregledovanje podatkov na ponudbi	4. 10. – 7. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Zaključevanje ponudbe – priprava za pošiljanje	8. 10., 11. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 3, preizkuševalec 4, preizkuševalec 5, preizkuševalec 6, preizkuševalec 7, preizkuševalec 8, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 12, preizkuševalec 13
Priprava dnevnega zaključka	12. 12. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 7, preizkuševalec 8
Priprava poročila na nivoju poslovalnice	11. 10. - 15. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 7, preizkuševalec 8

Priprava poročila na nivoju podružnice	11. 10. - 15. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 7, preizkuševalec 8
Priprava poročila na nivoju banke	11. 10. - 15. 10. 2003	preizkuševalec 1, preizkuševalec 2, preizkuševalec 9, preizkuševalec 10, preizkuševalec 11, preizkuševalec 7, preizkuševalec 8
Skrbniški modul	18. 10. - 22. 10. 2003	tehnološki skrbnik

Priloga 1: Poročilo o napaki informacijskega sistema za bančno zavarovalništvo – aplikacija EVITA

Verzija aplikacije:
Datum testiranja: Testiral:

Postopek, ki se testira	Rezultati testiranja		Naslov strani, kjer se pojavi napaka	Vhodni podatki	Opis napake:
	Pravilno	Nepravlno			

Priloga 2: Poročilo o testiranju

POROČILO O TESTIRANJU

INFORMACIJSKI SISTEM ZA PODPORO BANČNEMU ZAVAROVALNIŠTVU – APLIKACIJA EVITA

KAZALO

1 Uvod

1.1 Splošni podatki o testiranju

Oznaka izdelka: AW4020 Evita

Uporabljena konfiguracija strojne in programske opreme: spletna aplikacija Evita se je testirala na standardnih delovnih postajah (po standardih, ki veljajo v NLb d. d.).

Uporabljeni dokumenti:

- procesna navodila za bančno zavarovalništvo,
- navodila za delo z aplikacijo Evita na strežniku,
- navodilo za testiranje IT rešitev,
- standardi za PC opremo,
- interni standard o konstrukciji intranet aplikacij.

2 Udeleženci testiranja

Testno skupino sestavljajo:

- preizkuševalec 1 – Oddelek za razvoj procesne organizacijo in standardizacijo poslovanja,
- preizkuševalec 2 – Oddelek za uvajanje tehnoloških sprememb,
- preizkuševalec 3 – NLB Vita d. d.,
- preizkuševalec 4 - NLB Vita d. d.,
- preizkuševalec 5 - NLB Vita d. d.,
- preizkuševalec 6 - NLB Vita d. d.,
- preizkuševalec 7 – Oddelek za podporo prodaji prebivalstvu,
- preizkuševalec 8 - Oddelek za podporo prodaji prebivalstvu,
- preizkuševalec 9 - komercialist,
- preizkuševalec 10 – komercialist - mentor,
- preizkuševalec 11 - komercialist,
- preizkuševalec 12 - NLB Vita d. d.,
- preizkuševalec 13 - NLB Vita d. d.,
- tehnološki skrbnik,
- skrbnik strežnika,
- administrator podatkovne baze,
- 2 skupini tečajnikov.

2.1 Testiranje funkcionalnosti aplikacije

Informativni izračun:

- izdelava in izpis informativnega izračuna za vse zgoraj navedene storitve, pregledovanje že izdelanega informativnega izračuna.

Informativna ponudba:

- izdelava in izpis informativne ponudbe za storitev Življenjsko zavarovanje kreditojemalcev,
- pregledovanje informativne ponudbe,
- popravljanje informativne ponudbe,
- potrjevanje informativne ponudbe,
- zavračanje informativne ponudbe.

Ponudba:

- priprava in izpis ponudbe za zgoraj navedene storitve,
- pregledovanje ponudbe,
- popravljanje ponudbe,
- zaključevanje ponudbe.

Polica:

- Pregledovanje podatkov o polici.

Poročila:

- izdelava poročil na nivoju poslovalnice,
- izdelava poročil na nivoju podružnice,
- izdelava poročil na nivoju banke,
- izdelava dnevnega zaključka v okviru organizacijske enote.

Skrbniški modul:

- Spreminjanje šifrantov (Zdravstvena vprašanja, Dokumenti, Skladi, Drugi šifranti) (SKRBNIK),
- Tablice za določanje nevarnostnih premij (SKRBNIK).

3 Opis izvedbe testiranja

Testiranje informacijskega sistema se je odvijalo po načrtu, ki je opisan v Planu testiranja v času od 15. 9. - 14. 11. 2003 v prostorih NLB d. d.. Večjih odstopanj od predvidenega plana ni bilo, zato je plan testiranja ostal nespremenjen od začetka do konca. V predvidenem času so bile izvedene naslednje vrste testiranja:

- funkcionalno testiranje,

- sprejemno testiranje,
- stresno testiranje,
- performančno testiranje,
- obremenilno testiranje in
- operacijsko testiranje.

Ugotovljene napake pri testiranju so se odpravljale sproti, saj je testna skupina ves čas sodelovala s tehnološkim skrbnikom, le-ta pa z razvijalci. Vse posredovane napake so bile odpravljene sproti, torej ni nobene odprte neskladnosti.

3.1 Statistika ugotovljenih neskladnosti

Pregled/Testiranje	**Ustreza**	**Delno ustreza**	**Ne ustreza**
Opis izdelka	X		
Dokumentacija	X		
Programi/podatki	X		
SKUPAJ NESKLADNOSTI			

Razvrstitev napak	**Skupaj**	**Odpravljenih**	**Odprtih**
Napake tipa A	6	6	0
Napake tipa B	56	56	0
Napake tipa C	24	24	0
SKUPAJ NESKLADNOSTI			**0**

3.2 Rezultati testiranja

Izvedba vsakega testiranja je ustrezno dokumentirana z zapisi o testiranju. Vsi zapisi o testiranju, ki so jih pripravili udeleženci testiranja se hranijo v projekti dokumentaciji. Prav tako se v projektni dokumentaciji hrani vsa dokumentacija in sporočila o ugotovljenih napakah.

Udeleženci sprejemnega testiranja so ob zaključku testiranja podpisali sprejemni test in s tem izrazili, da testiran sistem ustrezna predpisanim zahtevam in da se dovoljuje prenos v produkcijsko okolje.

3.3 Zaključna ocena

Delo testne skupine je potekalo nemoteno po predpisanem planu. Preizkuševalci so se ves čas testiranja držali predpisanih rokov, tako, da so popravljanjem in spremembami časovnega plani ni bilo nobenih težav.

Udeleženci so aplikacijo sicer testirali na svojih delovnih mestih, kar pomeni, da so testiranje opravljali poleg svojih rednih obveznosti. Sodelovanje med preizkuševalci ocenjujemo kot dobro, prav tako pa tudi sodelovanje s tehnološkim skrbnikom.

Komunikacija je večinoma potekala s pomočjo elektronske pošte oz. na sestankih, kjer so se preizkuševalci pogovarjali o tekočih problemih v zvezi s testiranjem.

Končna ocena testiranja za to verzije je zadovoljiva. Za prihodnja testiranja bi za testno skupino morali imeti na razpolago ustrezno testno sobo, kjer bi se preizkuševalci lahko posvečali izključno svoji nalogi povezani s testiranjem in bi med seboj lahko ves čas sodelovali.

Priloga 3: Poročilo o uporabniškem sprejemnem testu

POROČILO O UPORABNIŠKEM SPREJEMNEM TESTU

Produkt – program in oznaka verzije: **Evita**, verzija 01.00.00

Ime in oznaka naloge: **Bančno zavarovalništvo**

Sodelavci in njihove zadolžitve:

- preizkuševalec 1 – Oddelek za razvoj procesne org. in standardizacijo poslovanja,
- preizkuševalec 2 – Oddelek za uvajanje tehnoloških sprememb,
- preizkuševalec 3 – NLB Vita d. d.,
- preizkuševalec 4 - NLB Vita d. d.,
- preizkuševalec 5 - NLB Vita d. d.,
- preizkuševalec 6 - NLB Vita d. d.,
- preizkuševalec 7 – Oddelek za podporo prodaji prebivalstvu,
- preizkuševalec 8 - Oddelek za podporo prodaji prebivalstvu,
- preizkuševalec 9 - komercialist,
- preizkuševalec 10 – komercialist - mentor,
- preizkuševalec 11 - komercialist,
- preizkuševalec 12 - NLB Vita d. d.,
- preizkuševalec 13 - NLB Vita d. d.,
- tehnološki skrbnik,
- skrbnik strežnika,
- administrator podatkovne baze,
- 2 skupini tečajnikov.

Datum začetka in datum konca sprejemnega testa: 15. 9. – 14. 11. 2003

Opis poteka sprejemnega testa in razlogi za morebitna odstopanja od načrta sprejemnega testa:

Sprejemno testiranje je potekalo v skladu s pripravljenim planom testiranja. Udeleženci testiranja so testne podatke za testiranje pripravili sami. Po vnosu podatkov so testirali delovanje aplikacije, beležili in sporočali ugotovitve. Dodatno se je pravilnost delovanja aplikacije preverjala tudi pri izobraževanju končnih uporabnikov (bančnih komercialistov, ki bodo tržili te storitve).

Potrditev pravilnosti delovanja posamezne funkcionalnosti:

Aktivnost	**Termin**	**Podpis izvajalca, ki potrdi pravilnost delovanja**
Nastavitve	15. 9. 2003	preizkuševalec 9
Pooblastila v sistemu Evita	15. 9. - 20. 9. 2003	preizkuševalec 1 preizkuševalec 9
Informativni izračuni (za vse storitve)	17. 9. - 24. 9. 2003	preizkuševalec 4
Izpis informativnega izračuna	17. 9. - 24. 9. 2003	preizkuševalec 4 preizkuševalec 7
Iskanje informativnih izračunov po različnih kriterijih	17. 9. - 24. 9. 2003	preizkuševalec 4 preizkuševalec 9
Vnos informativne ponudbe (za storitev življenjsko zavarovanje kreditojemalcev)	27. 9. – 30. 9. 2003	preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Izpis informativne ponudbe (za storitev življenjsko zavarovanje kreditojemalcev)	27. 9. – 30. 9. 2003	preizkuševalec 7 preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Potrjevanje, zavrnitev informativne ponudbe	27. 9. – 30. 9. 2003	preizkuševalec 4
Pregledovanje in popravljanje informativnih ponudb	30. 9. – 1. 10. 2003	preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Vnos ponudbe	4. 10. – 7. 10. 2003	preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Izpis ponudbe	4. 10. – 7. 10. 2003	preizkuševalec 4

Aktivnost	**Termin**	**Podpis izvajalca, ki potrdi pravilnost delovanja**
Popravljanje podatkov na ponudbi	4. 10. – 7. 10. 2003	preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Pregledovanje podatkov na ponudbi	4. 10. – 7. 10. 2003	preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Zaključevanje ponudbe – priprava za pošiljanje	8. 10., 11. 10. 2003	preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Priprava dnevnega zaključka	12. 12. 2003	preizkuševalec 9 preizkuševalec 10 preizkuševalec 11
Priprava poročila na nivoju poslovalnice	11. 10. -15. 10. 2003	preizkuševalec 1 preizkuševalec 10
Priprava poročila na nivoju podružnice	11. 10. -15. 10. 2003	preizkuševalec 1 preizkuševalec 10
Priprava poročila na nivoju banke	11. 10. - 15. 10. 2003	preizkuševalec 1 preizkuševalec 10
Skrbniški modul	18. 10. - 22. 10. 2003	tehnološki skrbnik

Ugotovitev o ustreznosti: Na podlagi izvedenega testiranja ugotavljamo, da testirana oprema **ustreza** podanim zahtevam in zato predlagamo, da **se** nadaljuje z uvedbo rešitve v produkcijsko okolje.

Datum: 25. 10. 2003 podpis predstavnika uporabnikov