UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

RAZVOJ INFORMACIJSKEGA SISTEMA ZA PODPORO GOSTOVANJU VOJAŠKIH MISIJ

Ljubljana, september 2016 METKA MARUŠIČ ZUPANČIČ

IZJAVA O AVTORSTVU

Podpisana Metka Marušič Zupančič, študentka Ekonomske fakultete Univerze v Ljubljani, avtorica predloženega dela z naslovom Razvoj informacijskega sistema za podporo gostovanju vojaških misij, pripravljenega v sodelovanju s svetovalcem red. prof. dr. Mirom Gradišarjem

IZJAVLJAM

1. da sem predloženo delo pripravila samostojno;

2. da je tiskana oblika predloženega dela istovetna njegovi elektronski obliki;

3. da je besedilo predloženega dela jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem poskrbela, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam oziroma navajam v besedilu, citirana oziroma povzeta v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani;

4. da se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku Republike Slovenije;

5. da se zavedam posledic, ki bi jih na osnovi predloženega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom;

6. da sem pridobila vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v predloženem delu in jih v njem jasno označila;

7. da sem pri pripravi predloženega dela ravnala v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobila soglasje etične komisije;

8. da soglašam, da se elektronska oblika predloženega dela uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;

9. da na Univerzo v Ljubljani neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve predloženega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja predloženega dela na voljo javnosti na svetovnem spletu preko Repozitorija Univerze v Ljubljani;

10. da hkrati z objavo predloženega dela dovoljujem objavo svojih osebnih podatkov, ki so navedeni v njem in v tej izjavi.

V Ljubljani, dne 30. 9. 2016 Podpis študentke: _____

KAZALO

KAZALO TABEL

KAZALO SLIK

UVOD

Republika Slovenija je kot članica Organizacije severnoatlantskega sporazuma (angl. *North Atlantic Treaty Organisation*, v nadaljevanju NATO) dolžna zagotavljati podporo države gostiteljice zavezniškim silam. Vlada Republike Slovenije je v letu 2003 sprejela prvi Načrt izvajanja podpore države gostiteljice, s katerim je to področje na načelni ravni celovito uredila. Dejavnost podpore države gostiteljice poteka v okviru logistike in civilno-vojaškega sodelovanja. Zagotavljanje podpore države gostiteljice pomeni kontinuiran proces mednarodnega dogovarjanja, dopolnjevanja načrtov in usposabljanja izvajalcev, sprotnega prilagajanja infrastrukture, namestitvenih in oskrbnih zmogljivosti, kot tudi potrebam zagotavljanja podpore države gostiteljice ter nenazadnje zbiranja, obdelave in vnosa spremenjenih podatkov v kataloge zmogljivosti in računalniško vodene baze podatkov držav članic Nata (Pipenbaher, 2006).

Ministrstvo za obrambo Republike Slovenije v skladu z določili navedenega načrta vzpostavlja informacijsko podporo za opravljanje nalog s področja podpore države gostiteljice. Najprej so se vsi potrebni podatki zbirali ročno, dokumenti so bili zbrani v omarah v mapah in iskanje je bilo zamudno. Zelo kmalu se je izkazala potreba po računalniško podprtem informacijskem sistemu za to področje. Izvedena je bila začetna sistemska analiza, pri kateri smo ugotovili, da je področje zelo obsežno, da bi za razvoj potrebovali veliko virov in da imamo vire zelo omejene. Kljub velikim željam po celoviti računalniški podpori področja je bila zato najprej sprejeta odločitev za izvedbo majhnega začetnega koraka. Vsebinski nosilec na Ministrstvu za obrambo je določil prioritete in tako smo v prvem koraku razvili računalniško evidenco civilnih nastanitvenih zmogljivosti za namen nudenja podpore države gostiteljice. Rešitev EHONAS je razvita nad relacijsko podatkovno bazo oracle z razvojnim orodjem APEX in deluje v spletnem okolju. Razvili smo jo po zaporedni oziroma slapovni metodi.

Obstoječo informacijsko rešitev za vodenje civilnih zmogljivosti je treba nadgraditi oziroma prenoviti v skladu z novim Načrtom zagotavljanja podpre države gostiteljice v Republiki Sloveniji (Vlada RS, 2015). Treba je zagotoviti vsaj naslednje razširitve: namesto o objektih ene vrste - nastanitveni objekti, nameravamo zbirati podatke o objektih več vrst, kar pomeni zbiranje drugačnih podatkov - atributi objektov bodo odvisni od vrste objekta. Potrebna bo reorganizacija podatkovne baze.

Poleg vodenja podatkov o objektih nameravamo dodati tudi novo funkcionalnost - objava obrazcev. Doslej so program uporabljali samo registrirani uporabniki, zdaj bo v določene funkcionalnosti omogočen vpogled vsem uporabnikom informacijskega sistema. Zaradi tega bo potrebno drugače zastaviti varnostno politiko. Doslej smo imeli dve vrsti uporabnikov, z zbiranjem novih podatkov moramo omogočiti dostop do določenih delov programa tudi uporabnikom z drugih inštitucij, kar v internem omrežju naše organizacije ni možno. Potrebna bo selitev v drugo omrežje.

S programom zbrani in vodeni podatki se bodo periodično izvažali za potrebe poročanja mednarodni organizaciji, katere člani smo. Neposredne povezave med našim informacijskim sistemom in informacijskim sistemom mednarodne organizacije ni možno vzpostaviti, ker delujeta vsako v svojem ločenem omrežju. Treba je vzpostaviti zanesljiv način prenosa podatkov preko izmenljivih medijev. Podatke je treba pripraviti po standardu razširljivega označevalnega jezika (angl. *extensible markup language*, v nadaljevanju XML). XML je od strojne in programske opreme neodvisno orodje za hranjenje in prenos podatkov (Introduction to XML, 2016).

Prenove poslovnih procesov se lahko lotimo strateško. V tem primeru je prenova usmerjena v vsa ključna strateška vprašanja in zajema vse procese in njihovo avtomatizacijo. Lahko pa prenovimo in informatiziramo samo posamezne procese (Gradišar, Jaklič, Talib, & Baloh, 2005). V obravnavanem primeru bomo informatizirali posamezni proces. Pričakujemo, da bo ob tem prišlo tudi do hkratne prenove procesa.

Viri za razvoj nove programske rešitve so omejeni tako kadrovsko kot finančno, zato je treba najti pametno ravnotežje med uporabo notranjih virov in oddajo del zunanjim izvajalcem. Prednost notranjih virov je, da zanje ni potrebno načrtovati dodatnih finančnih sredstev. Razvoj po načelu "naredi sam" se sicer lahko dobro obnese pri izdelavi manjših internih aplikacij z nekaj funkcijami, a v praksi ne deluje za razvoj večjih rešitev (Berdugo, 2014).

Pri razvoju programskih rešitev se je uveljavilo več metodologij. Na metodologijo lahko gledamo kot na množico dogovorov, ki jih sprejme projektna organizacija. Obstajajo številne formalne metodologije, ki temeljijo na preizkušenih postopkih, ki so se v praksi izkazali kot dobri. Vendar organizacije si jih pri prevzemu vedno prilagodijo po svoji meri, tako da ustrezajo njihovemu načinu dela ter percepciji domene, za katero so vzpostavljene (Bajec & Krisper, 2003, str. 2).

V zadnjem času se uveljavljajo agilni pristopi k razvoju. Agilno metodologijo opredeljuje seznam značilnosti, ki jo uvršča med agilne (Agile Manifesto, 2016). Pri odločanju za metodologijo razvoja je smiselno preučiti tudi prednosti, ki jih agilne metodologije morda prinašajo v naše razvojno okolje.

Namen in cilji magistrskega dela. Namen magistrskega dela je izdelati analizo trenutnega stanja informacijske rešitve za podporo gostovanju vojaških misij in pripraviti načrt za njeno prenovo v skladu s pričakovanji naročnika na eni strani in tehnoloških, varnostnih, kadrovskih in finančnih omejitev na drugi strani. Načrt za prenovo naj vsebuje tako opredelitev funkcionalnosti novega informacijskega sistema kot tudi odločitev o izbiri razvojnega orodja, izbiri razvojne metodologije in predvidenih iteracijah razvoja.

Da bi dosegli ta namen, smo si zastavili več ciljev. Prvi cilj se nanaša na preučitev različnih pristopov oziroma metodologij k razvoju informacijskih sistemov, kot jih podaja literatura. Drugi cilj je analiza možnosti razvojnega orodja Oracle APEX in njegove primernosti za uporabo agilnega razvojnega pristopa. Tretji cilj je preučitev procesa podpore gostovanja vojaških misij, kot ga določa Načrt zagotavljanja podpre države gostiteljice v Republiki Sloveniji (Vlada RS, 2015). Četrti cilj je analiza trenutnega stanja informacijske podpore obravnavanega procesa. Peti cilj je opredelitev funkcionalnosti novega informacijskega sistema. Šesti cilj je izdelava izvedbenega načrta za izdelavo novega sistema in ocena ekonomičnosti tako zastavljenega projekta. Sedmi cilj je prikazati, kako zrelost tehnologije, razvojnih orodij in metodologij omogoča hitrejši in lažji razvoj zanesljivih spletnih programskih rešitev.

V tem delu nameravamo potrditi dve hipotezi. Prva hipoteza je, da je prenovo informacijskega sistema za podporo gostovanju vojaških misij možno ekonomično izvesti z razvojnim orodjem Oracle APEX in z lastno agilno metodologijo. Druga hipoteza je, da trenutna zrelost tehnologije, razvojnih orodij in metodologij omogoča hitrejši in lažji razvoj zanesljivih spletnih programskih rešitev v primerjavi z načinom dela v preteklosti.

Metode dela. Pri izdelavi magistrskega dela bomo uporabili znanja, pridobljena s študijem in znanja, pridobljena pri delu na področju razvoja informacijskih rešitev. Delo bo temeljilo na preučevanju teoretičnih osnov in dokumentiranih praktičnih izkušenj. Uporabili bomo domačo in tujo literaturo, pomagali si bom z gradivom, dostopnim na svetovnem spletu.

Pri pregledovanju literature in virov bomo uporabili metodo analize in sinteze. Pri pisanju naloge bomo uporabili deskriptivni in analitični način.

1 PRENOVA IN INFORMATIZACIJA POSLOVNIH PROCESOV

1.1 Prenova in informatizacija poslovnih procesov

Sedanje organizacije delujejo v hitro spreminjajočem se okolju, hkrati pa se spremembe nenehno dogajajo tudi znotraj organizacij. Spremembe so vse bolj nepredvidljive. Če organizacije želijo preživeti, morajo svoje delovanje nenehno prilagajati tem spremembam. Izboljšati morajo način pridobivanja podatkov in jih učinkoviteje uporabljati pri izvajanju svojih poslovnih procesov (Kovačič, Jaklič, Indihar Štemberger, & Groznik, 2004, str. 2).

Uspešna podjetja obravnavajo informacije kot enega od vitalnih delov svojega premoženja. Uporabljajo jih učinkovito, stalno jih posodabljajo in jih skrbno varujejo (Shelly, Cashman, & Rosenblatt, 2008, str. 4).

Poslovni proces je skupek logično povezanih izvajalskih in nadzornih postopkov, katerih posledica je nek izid. Izvajanje procesa sproži nek dogodek ali več dogodkov. Proces ima svoj vhod in rezultat izvajanja. Pomembno je, da ima proces svojega lastnika, to je posameznik, ki proces nadzira in je odgovoren za njegovo uspešnost. Vsak proces ima določene omejitve glede izvajanja kot tudi omejitve pristojnosti lastnika procesa. Pri izvajanju prrocesa nastajajo stroški in porablja se čas. Kot ključne dejavnike uspeha procesa razumemo tiste cilje, katerih izpolnitev zagotavlja uspešnost izvajanja procesa. Proces je sestavljen iz manjših delov, to so podprocesi in/ali aktivnosti (Kovačič et al., 2004, str. 78).

Konkurenčna prednost organizacije je sposobnost, da uspešno konkurira ostalim organizacijam. V življenjskem ciklu organizacije se pojavljajo številne priložnosti za doseganje konkurenčne prednosti. Nekatere organizacija uporabi, mnogih ne. Praksa in raziskave pa kažejo, da ustrezno razvita informatika predstavlja eno izmed poslovnih priložnosti organizacije v boju s tržno konkurenco (Kovačič et al., 2004, str. 3).

Izraz **informatika** je skovanka iz dveh besed: informacija in avtomatika. Področje oziroma vsebina te znanstvene discipline je avtomatizacija obdelave informacij. V šestdesetih letih dvajsetega stoletja se je informatika ukvarjala predvsem z razvojem računalnikov in jo zato imenujemo **tehnična informatika**. S hitrim razvojem tehnične informatike so računalniki postajali vedno bolj zmogljivi, cene pa so se zniževale. Tako se je področje uporabe računalnikov začelo širiti. Pojavila se je nova veja informatike, imenovana **uporabna informatika**, ki se je ukvarjala z učinkovito in smiselno uporabo računalnikov v praksi. Tisti del uporabne informatike, ki se ukvarja z uporabo računalnikov v poslovnih sistemih, se imenuje **poslovna informatika**. Poslovno informatiko lahko obravnavamo tudi kot poslovno infrastrukturo, saj je za poslovanje danes nujno potrebna (Gradišar et al., 2012, str. 3).

Informacijska tehnologija je kombinacija računalniške strojne in programske opreme ter druge opreme in storitev, ki omogoča zbiranje, obdelavo, shranjevanje distribucijo in izmenjavo podatkov (Gradišar et al., 2012, str. 62). Sistematično in načrtno vključevanje informacijske tehnologije v poslovanje podjetja, ki ima za cilj razviti poslovnim potrebam primeren informacijski sistem, imenujemo **informatizacija** (Gradišar et al., 2012, str. 144).

Uporaba komunikacijske in informacijske tehnologije danes organizacijam omogoča hitro in kakovostno obdelavo podatkov in hiter prenos na poljubno mesto. Uvajanje informacijskih sistemov močno spreminja podjetja, tako njihovo delovanje navzven kot tudi njihovo notranjo organiziranost. Dogajajo se velike spremembe v gospodarstvu, državi in celotni družbi. Družba prehaja iz industrijske v informacijsko družbo (Gradišar et al., 2012, str. 5).

Osnovna značilnost **informacijske družbe** je, da je največ ljudi zaposlenih v informacijskih poklicih. V teh poklicih se ljudje ukvarjajo z ustvarjanjem, obdelavo in prenašanjem informacij (Gradišar et al., 2012, str. 5). Informatizacija je splošen in celovit proces uvedbe in uporabe informacijske tehnologije, ki ga v informacijski družbi glede na njegov pomen lahko enačimo s procesom industrializacije industrijske družbe (Kovačič et al., 2004, str. 2).

Še tako dobra informacijska tehnologija pa za večjo uspešnost poslovanja ni dovolj. Če informatiziramo dejavnosti, ki jih izvajamo na neučinkovit način ali morda celo brez prave potrebe, nam nobena tehnologija ne bo pomagala k večji uspešnosti poslovanja. Za uspešno informatizacijo so pogosto potrebne tudi organizacijske spremembe, torej prenova poslovanja, ki jo dosežemo s prenovo poslovnih procesov. Same naložbe v informacijsko tehnologijo neposredno ne vplivajo na dvig poslovne uspešnosti in konkurenčnosti organizacij. S prenovo poslovnih procesov lahko dosežemo, da organizacija dela prave stvari na pravi način. Hkrati z informatizacijo je treba poskrbeti tudi za prenovo poslovanja (Gradišar et al., 2012, str. 144).

Prenova poslovnih procesov je dejavnost, pri kateri poslovne procese organizacije poenotimo, včasih tudi na novo definiramo ali jih radikalno spremenimo. Šele potem jih je smiselno informatizirati oziroma podpreti s primerno informacijsko tehnologijo (Gradišar et al., 2012, str. 148). Prenova poslovnih procesov ima za cilj doseganje dramatičnih izboljšav kazalcev uspešnosti, kot so stroški, kakovost storitev in hitrost. Pri uporabi informacijske tehnologije moramo biti pozorni, da ne avtomatiziramo morebitnih neoptimalnih procesov, ampak da stremimo k uporabi informacijske tehnologije za doseganje novih ciljev. Za uspešno prenovo procesov je ključna sposobnost prepoznavanja novih načinov uporabe tehnologije (Gradišar et al., 2012, str. 154).

Prenova poslovnih procesov zajema naslednje globalne cilje (Gradišar et al., 2012, str. 160):

1. skrajšanje poslovnega cikla oziroma vseh procesov v organizaciji, dvig odgovornosti in s tem znižanje stroškov,
2. dvigovanje dodane vrednosti in ob tem postopno dvigovanje kakovosti proizvodov in storitev organizacije,
3. zniževanje stroškov postopkov ob ohranjanju ustreznega razmerja do kakovosti in časa in
4. dvigovanje zanesljivosti in doslednosti izvajanja postopkov, s tem pa tudi kakovosti poroizvodov in storitev.

Prenove poslovnih procesov in informatizacije se lahko lotimo na več načinov, v novejši literaturi se poudarja izvedba v naslednjih treh korakih (Gradišar et al., 2012, str 145):

1. strateško načrtovanje prenove procesov in informatizacije,
2. prenova poslovanja in razvoj informacijske arhitekture,
3. razvoj in/ali uvedba informacijskih rešitev.

Pri tem načinu v drugem koraku hkrati s prenovo poslovanja razvijemo tudi informacijsko infrastrukturo. To pomeni, da opredelimo (Gradišar et al., str 145):

1. globalni podatkovni model (podatkovno arhitekturo),
2. poslovna pravila,
3. informacijsko infrastrukturo (to je tehnološka arhitektura),
4. metode in metodologije za prenovo poslovnih procesov in razvoj informacijskih sistemov.

Informacijska infrastruktura je sklop tehnologij, ki podjetju ali organizaciji omogočajo praktično izvedbo informacijske arhitekture. Zagotavlja tehnološke pogoje za informacijsko arhitekturo. Običajno s tem pojmom zajamemo (Gradišar et al., 2012, str. 138):

1. računalniško platformo, to je arhitekturo strojne opreme,
2. tehnologije za izvedbo prenosnih storitev in storitev omrežij in
3. vmesno programsko opremo, ki je namenjena povezovanju uporabniških programov z drugimi uporabniškimi programi.

Strateško načrtovanje prenove poslovanja obravnava strateške usmeritve organizacije in ukrepe, ki imajo za cilj poslovno uspešnost organizacije. Začne se z obravnavo perspektive organizacije o njenih poslovnih ciljih ter o načinu, kako jih uresničiti. To naj bi bilo že opredeljeno v strateškem načrtu organizacije. Šele potem pridejo na vrsto tehnične značilnosti računalniške opreme. Rezultat je **strateški načrt prenove**. Strateško načrtovanje prenove obsega naslednje aktivnosti (Gradišar et al., 2012, str. 146):

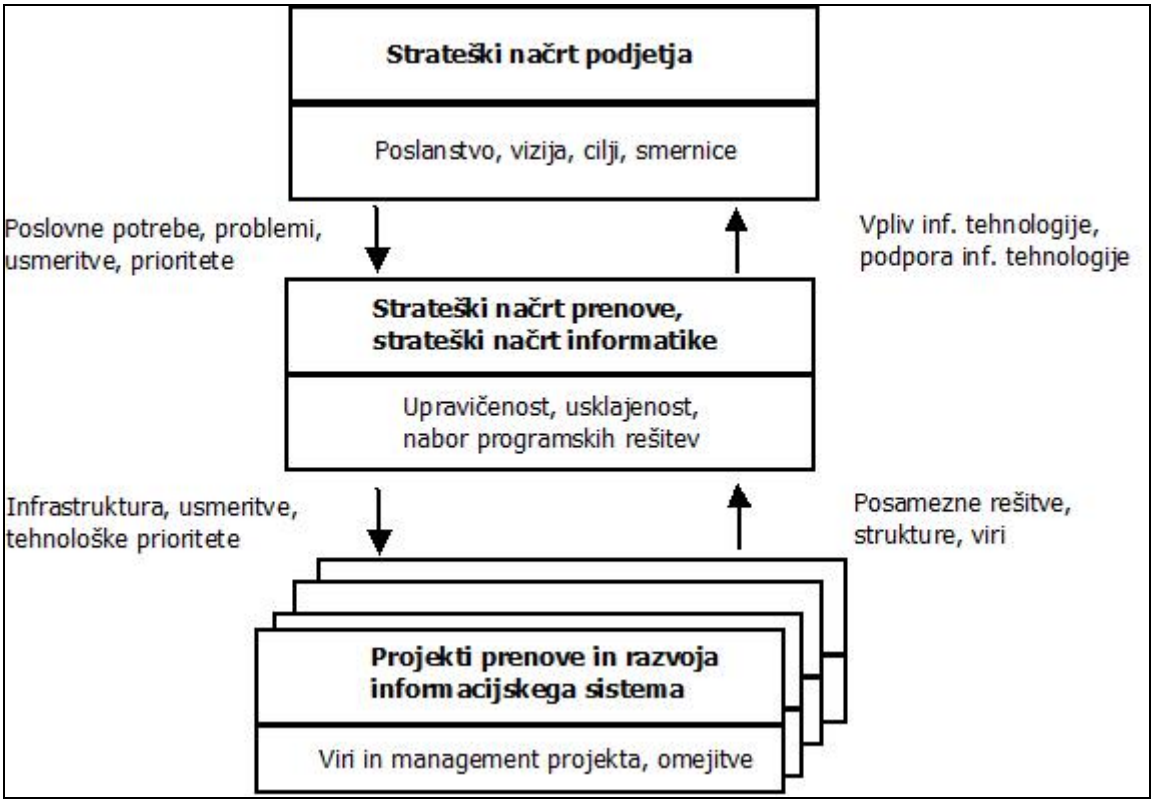
1. oceno ustreznosti izvajanja procesov po sedanjem načinu in njihove trenutne informatizacije,
2. določitev strategije, ciljev, politike in ključnih dejavnikov uspeha,
3. določitev ključnih poslovnih procesov in grobo analizo informacijskih potreb ter
4. oceno možnosti in grob načrt prenove ter informatizacije poslovanja.

Strateški načt informatike je dokument, v katerem organizacija opredeli dolgoročne cilje glede informacijskega sistema organizacije, ki so potrebni za uresničitev ciljev organizacije. Praviloma vsebuje opredelitev poslanstva organizacijske enote, pristojne za informatiko. Poslanstvo službe za informatiko naj bi bilo takšno, da lahko ta služba bistveno prispeva k učinkovitosti, uspešnosti in konkurenčnosti podjetja. Načrt se običajno

dopolnjuje enkrat letno. Organizacija mora nenehno skrbeti, da sta strateški načrt organizacije in strateški načrt informatike usklajena (Gradišar et al., 2012, str. 146).

Praviloma posamezni projekti prenove poslovnih procesov in projekti razvoja informacijskega sistema sledijo strateškemu načrtovanju informatike (Gradišar et al., 2012, str. 147). Odnos med strateškim načrtom podjetja, strateškim načrtom prenove in informatizacije ter posameznimi projekti prenove in razvoja je prikazan na Sliki 1.

Slika 1: Strateški načrt podjetja, strateški načrt prenove in informatizacije in posamezni projekti prenove in razvoja informacijskega sistema



Vir: M. Gradišar, J. Jaklič, & T. Turk, Osnove poslovne informatike, 2012, str. 147, slika 4.1.

Prenove poslovnih procesov se lahko organizacije lotijo bolj ali manj radikalno. Celovita ali strateška prenova poslovanja zajema vse poslovne procese organizacije in njihovo informatizacijo. Lahko pa se organizacija loti le prenove in informatizacije posameznega procesa (Gradišar et al., 2012, str. 154).

Tudi v inštitucijah državne uprave potekajo prizadevanja za kakovost delovanja in kakovost storitev. Kakovost pomeni skladnost s specifikacijami, standardi in s pričakovanji strank. V državni upravi to pomeni skladnost s sprejetimi predpisi in skladnost s pričakovanji državljanov, to je strank državne uprave oziroma uporabniki njenih storitev. Vedno boj se uveljavlja zahteva, da je tudi v delovanje organizacij državne uprave treba

uvesti mehanizme zagotavljanja učinkovitosti in nenehnega izboljševanja. Sredstva za delovanje državnega aparata so omejena in zato so velike tudi zahteve po znižanju izdatkov. V tem smislu se išče učinkovitejše načine vodenja, upravljanja in delovanja (Žurga, 2015, str. 7). Razvoj in uporaba ustreznih informacijskih sistemov lahko tako kot v gospodarskih družbah tudi v državni upravi pripomoreta k doseganju teh ciljev.

1.2 Baza podatkov

Podatkovna struktura je okvir za organizacijo in shranjevanje podatkov v informacijskem sistemu. Sestavljajo jo datoteke ali tabele, ki so povezane na več načinov. Glede na način povezav med datotekami ali tabelami ločimo datotečni (datotečno orientirani) sistem in sistem baze podatkov (Shelly et al., 2008, str. 340).

Datotečni sistem shranjuje podatke v eni ali več ločenih datotekah. Glavna pomanjkljivost datotečnega sistema je, da so isti podatki shranjeni na večih mestih. Kronološko je ta sistem starejši od baze podatkov in ima v primerjavi z njo več pomanjkljivosti. Še vedno pa se uporablja takrat, ko se redno obdeluje velike količine strukturiranih podatkov (Shelly et al., 2008, str. 340).

Sistem baze podatkov pa sestavljajo medsebojno povezane tabele, ki vse skupaj tvorijo eno podatkovno strukturo. V primerjavi s procesiranjem na datotetečnem sistemu pomeni sistem baze podatkov večjo fleksibilnost in učinkovitost (Shelly et al., 2008, str. 340).

Sistem za upravljanje s podatkovno bazo (angl. *database management system*, v nadaljevanju SUPB) je program, ki nadzira shranjevanje, organizacijo in priklic podatkov (Introduction to Oracle Database, 2016). To je zbirka orodij, lastnosti in vmesnikov, ki uporabnikom omogočajo dodajanje in spreminjanje vsebine zbirke podatkov, kot tudi njeno upravljanje in analizo. Shematsko so komponente SUPB prikazane na Sliki 2. S stališča uporabnika je glavna prednost SUPB v tem, da omogoča hiter, interaktiven in prožen dostop do podatkov (Shelly et al., 2008, str. 342).

Prednosti uporabe SUPB v primerjavi z datotečnim sistemom so predvsem (Shelly et al., 2008, str. 343):

1. skalabilnost, kar pomeni, da sistem lahko zlahka povečujemo, spreminjamo ali zmanjšujemo glede na nove zahteve, ki se v hitro spreminjajočem se poslovnem svetu stalno pojavljajo,
2. boljša podpora sistemom odjemalec/strežnik,
3. ekonomija obsega: kadar organizacija uporablja eno bazo podatkov za vse svoje potrebe, je strojna oprema bolje izkoriščena. Delovanje je cenejše na zmogljivem osrednjem računalniku kot pa uporaba več manjših računalnikov,

4. prožnejša skupna raba podatkov: podatki, zbrani na enem osrednjem mestu, se zlahka dajo na uporabo vsem, ki dostop do njih potrebujejo,
5. uporaba v celotni organizaciji: SUPB bolj učinkovito kot pa datotečni sistem podpirajo delovanje programskih rešitev, razvitih za uporabo v celotni organizaciji,
6. boljša standardizacija: učinkovita administracija podatkovne baze pomaga zagotoviti, da so imena entitet in atributov ter formati podatkov po vseh programskih rešitvah v organizaciji dodeljeni po istem sistemu,
7. nadzorovana redundanca: ker podatke shranjujemo v zbirki medsebojno povezanih tabel, jih ni potrebno podvajati na več lokacijah; tudi če uvedemo nekaj redundance zaradi boljših performans ali zaradi možnosti vzpostavitve prvotnega stanja po katastrofi, nam pristop s podatkovno bazo omogoča nadzor nad podvajanjem podatkov,
8. večja varnost: administrator baze lahko opredeli avtorizacijske postopke, ki zagotavljajo, da imajo različni uporabniki različen dostop do podatkov,
9. večja učinkovitost razvijalcev programske opreme, saj jim ni potrebno kreirati datotečne strukture za bazo podatkov, lahko se osredotočijo na logično načrtovanje in tako novo rešitev razvijejo hitreje, in
10. neodvisnost podatkov: sistemski administrator lahko spreminja podatkovne strukture brez poseganja v informacijske sisteme, ki te podatke uporabljajo.

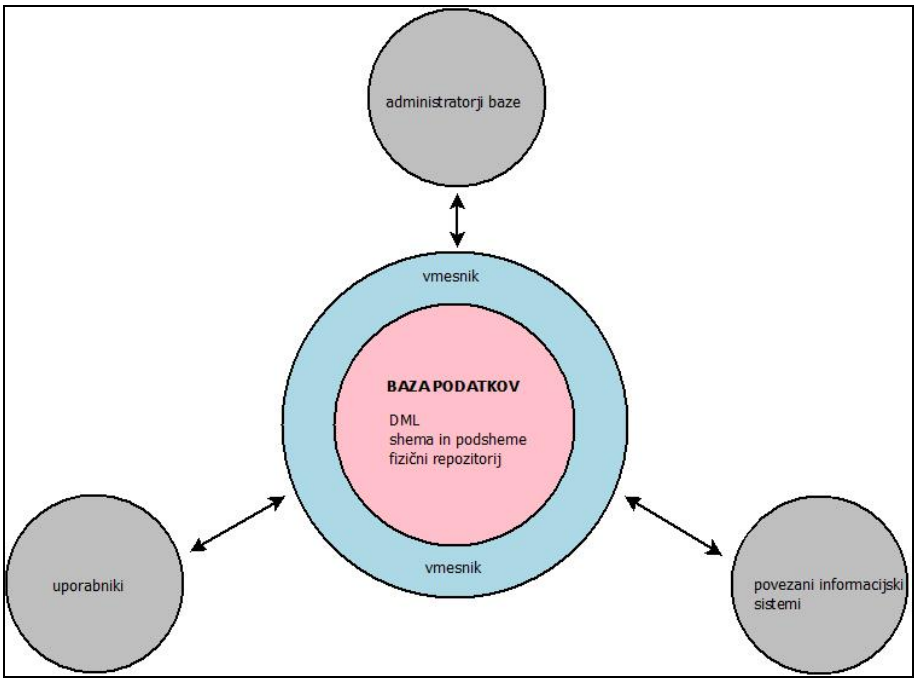
Sistemi za upravljanje baze podatkov imajo veliko prednosti pred datotečnim sistemom, a obstaja tudi nekaj težav, povezanih z njimi (Shelly et al., 2008, str. 343):

1. SUPB so zelo zmogljivi sistemi, zato pa potrebujejo za svoje delovanje dražjo strojno opremo, dražjo programsko opremo in omrežja, ki so sposobna podpirati večuporabniško okolje,
2. SUPB so tudi mnogo bolj kompleksni od datotečnih sistemov, zato je krivulja učenja za sistemske analitike, administratorje baze in uporabnike bolj strma, kar poveča skupne stroške lastništva (angl. *total cost of ownership*, v nadaljevanju TCO) in nazadnje
3. varnostni postopki, izdelava varnostnih kopij in obnovitev sistema so bolj kompleksni in kritični. Kadar ima organizacija vse svoje vitalne podatke v eni podatkovni bazi, potem prekinitev delovanja te baze resno ogrozi poslovne operacije.

Mnogi SUPB omogočajo **nadzor celovitosti podatkov** (angl. *data integrity*). Celovitost podatkov pomeni pomenske omejitve na podatkih, ki jih lahko načrtovalec baze definira v sami bazi in so odraz poslovnih pravil. Kadar omejitve celovitosti določamo v sami bazi, je velika prednost ta, da jih določimo enkrat in veljajo za vse načine vnosa in spreminjanja podatkov (Jaklič, 2002, str. 80):

1. preko že izdelanih informacijskih rešitev,
2. preko posegov v podatke z namenskimi orodji z DML jezikom in
3. za vse programske rešitve, ki jih bomo mi ali kdo drug nad temi podatki izdelali v prihodnosti.

Slika 2: Komponente SUPB



Vir: G. B. Shelly, T. J. Cashman, & H. J. Rosenblatt, Systems analysis and design, 2008, str. 344, slika 8-6.

Jezik za manipulacijo podatkov (angl. *data manipulation language*, v nadaljevanju DML) je namenjen upravljanju baznih operacij nad podatki, to je shranjevanju, priklicu, spreminjanju in brisanju podatkov (Shelly et al., 2008, str. 345).

Uporabniki bodo v primeru poskusa kršitve omejitve celovitosti na ekran dobili sistemsko sporočilo SUPB, da operacija ni dovoljena. Če nam ali uporabnikom oblika sporočila ni všeč ali je nerazumljiva, se bomo morali potruditi in v uporabniških rešitvah doprogramirati prestrezanje sistemskih sporočil in dodati prikazovanje uporabnikom bolj prijaznih in razumljivih sporočil. Takšnemu delu se pogosto ne moremo izogniti v primeru, ko je jezik SUPB angleščina, saj uporabniki pričakujejo prijazna sporočila v domačem jeziku.

Kadar SUPB ne omogoča realizacije omejitve celovitosti v sami bazi ali pa se za izdelavo te omejitve v bazi ne odločimo, pa se moramo zavedati, da bo treba zagotavljati kontrolo vnosa in spreminjanja podatkov posebej v vsaki programski rešitvi, ki do teh podatkov dostopa. Posebej bodo morali na upoštevanje dogovorjenih, a v bazi nerealiziranih pravil paziti tisti uporabniki, ki do podatkov lahko dostopajo mimo sprogramiranih rešitev, z orodji za uporabo DML. Takšna rešitev je načeloma manj zanesljiva.

Običajno ima podatkovna baza veliko uporabnikov, mnogi med njimi pa opravljajo enako ali podobno delo. Tako da potrebujejo tudi enake ali pa podobne pravice za delo. Zato se v

praksi pogosto kreirajo **uporabniške vloge**. Vsaka uporabniška vloga je namenjena eni vrsti uporabnikov. Uporabniška vloga je množica pravic. Na primer, vsi skladiščniki v organizaciji imajo enake pravice za vnos in spreminjanje podatkov v tabeli, ki vsebuje podatke o zalogah, le da vsak sme delati le za svoje skladišče. V takšnem primeru lahko definiramo eno uporabniško vlogo za skladiščnika, ob vsakokratni prijavi uporabnika v sistem pa se iz tabele uporabnikov prebere še podatek, za katero skladišče ima dovoljen dostop in se mu ustrezno temu prikazujejo podatki. Na tak način ima vsak skladiščnik definiran svoj pogled na podatke iz tabele, do katere vsi dostopajo.

V besedilih, ki obravnavajo podatkovne baze in njihovo načrtovanje, se srečujemo z uporabo besede **model** v dveh pomenih. S to besedo poimenujemo prikaz proučevanega realnega sistema na drugačen, poenostavljen in formaliziran način (Model, 2016). Poenostavitev pomeni, da model odraža določene značilnosti realnega sistema, ne pa vseh. V tem pomenu besedo uporabljamo, ko govorimo o analizi zahtev.

Z besedo **model** pa poimenujemo tudi ustaljeno obliko česa, po kateri se kaj dela, to je vzorec oziroma kalup. V tem pomenu besedo uporabljamo, ko govorimo o načinu organiziranja podatkov v podatkovni bazi oziroma o načinu delovanja obravnavanega SUPB.

Model podatkovne baze (pri nekaterih avtorjih imenovan tudi podatkovni model) je množica pravil, ki določajo, kako smejo biti podatki v podatkovni bazi organizirani oziroma strukturirani. **Relacijski podatkovni model** temelji na matematični teoriji relacij, od tod izvira tudi ime. Relacijske podatkovne strukture so fizično predstavljene z dvodimenzionalnimi tabelami.

Na trgu je več relacijskih SUPB. Po poročilu Gartner group za leto 2011, ki ga v svoji predstavitvi navaja podjetje Oracle, imajo največje tržne deleže Oracle DBMS proizvajalca Oracle, DB2 proizvajalca IBM in MS SQL proizvajalca Microsoft (Oracle, 2010). Veliko pa se uporablja tudi odprtokodni SUPB mySQL.

Shema je konkretna uporaba podatkovnega modela. To je načrt podatkovne baze, ki je v skladu z modelom baze, torej s predpisi podatkovnega modela. Odraža obravnavano problemsko področje, torej del realnega sveta, ki ga želimo informatizirati (Jaklič, 2004, str. 86). Shema je kompletna definicija podatkovne baze, ki vsebuje opise vseh polj, tabel in povezav. Definiramo lahko tudi eno ali več podshem. **Podshema** je pogled na bazo, namenjen enemu ali več uporabnikom ali sistemom. Vsebuje le tiste dele podatkovne baze, do katerih morajo ali smejo dostopati (Shelly et al., 2008, str. 345).

Načrtovanje baze podatkov ponavadi poteka v petih korakih (Grad & Jaklič, 1996, str. 40):

1. zbiranje in analiza zahtev,
2. izdelava konceptualnega podatkovnega načrta,
3. izbira sistema za upravljanje podatkovne baze (SUPB)
4. prevedba konceptualnega podatkovnega načrta v logični podatkovni načrt in
5. izdelava fizičnega podatkovnega načrta.

Zbiranje in analizo zahtev izvajajo sistemski analitiki, ki v postopku sistemske analize hkrati analizirajo tokove podatkov kot tudi ugotavljajo strukturo podatkov. Rezultati analize zahtev so tako diagrami toka podatkov kot tudi konceptualni podatkovni načrt ali **slovar podatkov** (angl. *data dictionary* ali *data repository)*. V slovarju podatkov so definirani in opisani vsi podatkovni elementi. **Podatkovni element** je najmanjša enota podatkov, ki ima pomen v informacijskem sistemu. Podatkovni elementi se združujejo v podatkovne zapise. **Podatkovni zapis** je smiselna kombinacija podatkovnih elementov, ki je vključena v tok podatkov ali shranjena v bazi (Shelly et al., 2008, str. 167).

Vsak podatkovni element v slovarju podatkov ima svoje ime, opredeljen ima tip podatka in dolžino. Lahko ima določeno tudi privzeto vrednost in dovoljene vrednosti.

Z izbiro SUPB, ki implementira določen podatkovni model, na primer relacijski, smo izbrali tudi podatkovni model, v katerem bomo realizirali bazo podatkov. Ko konceptualni podatkovni načrt prevedemo v model podatkovne baze, smo izdelali logični podatkovni načrt. Fizični podatkovni načrt opredeljuje notranji model podatkovne baze (Grad & Jaklič, 1996, str. 41). Fizični podatkovni model ali repozitorij je lahko centraliziran na enem mestu ali pa je distribuiran na več lokacijah (Shelly et al., 2008, str. 346).

1.3 Informacijski sistem

1.3.1 Pojem informacijski sistem

S pojmom **sistem** poimenujemo neko sestavljeno celoto, torej množico elementov, ki so medsebojno povezani in dajo nek rezultat. **Informacijski sistem** zbira, obdeluje, analizira, shranjuje in posreduje informacije za določen namen. **Sestavine informacijskega sistema** organizacije so strojna oprema, programska oprema, podatki, postopki in ljudje. Pod pojmom **strojna oprema** razumemo fizični del informacijskega sistema, torej računalnike, vhodne in izhodne enote, telekomunikacijske naprave in ostalo informacijsko infrastrukturo. **Programsko opremo** sestavljajo sistemski programi in uporabniške programske rešitve. Sistemski programi so tisti, ki nadzirajo delovanje računalnika (operacijski sistem, gonilniki naprav, protivirusni programi, programi za izdelavo varnostnih kopij, omrežni informacijski sistem). Uporabniške programske rešitve rešujejo specifične probleme uporabnikov (Shelly et al., 2008, str. 6).

Satzinger in soavtorji definirajo **informacijski sistem** kot množico medsebojno povezanih računalniških komponent, ki zbira, obdeluje, shranjuje in na izhodu zagotavlja informacije, ki so potrebne za poslovanje organizacije. Posebej ti avtorji opredelijo **računalniško aplikacijo** kot računalniški program, ki se izvaja na neki napravi z namenom, da izvede določeno funkcijo ali množico povezanih funkcij. Včasih se pojma informacijski sistem in aplikacija v govoru ali pisanju zamenjujeta, a običajno se pojem aplikacija nanaša le na računalniški program, z izrazom informacijski sistem pa običajno poimenujemo tako programsko opremo, bazo podatkov, kot tudi povezane ročne procese (Satzinger, Jackson & Burd, 2012, str. 4).

1.3.2 Projekt razvoja in deležniki

Da bo razvoj informacijskega sistema uspešen, mora biti dobro načrtovan in organiziran (Satzinger et al., 2012, str. 209). Pri razvoju konkretnega informacijskega sistema gre za neponovljivo enkratno nalogo, ki ima svoj začetek in konec. Zato izgradnja novega informacijskega sistema običajno poteka projektno.

S pojmom **projekt** tu mislimo na načrtovano dejavnost, ki ima začetek in konec in proizvede nek konkreten rezultat. Nekateri projekti so vodeni zelo formalno, drugi spet so tako neformalni, da jih kot projekt komajda prepoznamo. Ne glede na stopnjo formalnosti pa projektni način dela pomeni, da so znane aktivnosti, ki jih bo potrebno narediti, da so te aktivnosti načrtovane, organizirane in tudi nadzorovane (Satzinger et al., 2012, str. 5).

Deležniki na projektu so vse tiste osebe, ki jim je v interesu uspešna izvedba projekta. Deležniki so glavni vir informacij pri definiciji zahtev. Pred zbiranjem podrobnih informacij je treba identificirati vse vrste oziroma skupine deležnikov. Za uspešno delo na projektu si moramo zagotoviti sodelovanje vsaj ene ključne osebe iz vsake skupine. Pri iskanju deležnikov si lahko pomagamo z naslednjo osnovno klasifikacijo deležnikov (Satzinger et al., 2012, str. 44):

1. notranji deležniki (angl. *internal stakeholders*): to so tisti ljudje, ki delujejo znotraj organizacije, za katero razvijamo informacijsko rešitev, in bodo delali s sistemom ali pa je za njihovo delo uspešno delovanje sistema pomembno,
2. zunanji deležniki (angl. *external stakeholders*): so ljudje, ki ne delujejo v sklopu organizacije, pa je vseeno delovanje sistema zanje pomembno,
3. operativni deležniki (angl. *operational stakeholders*): ljudje, ki bodo redno uporabljali sistem bodisi za opravljanje svojega dela na delovnem mestu bodisi za urejanje zadev v svojem življenju in
4. deležniki na nivoju vodstva (angl. *executive stakeholders*): ti se ne bodo neposredno ukvarjali s sistemom, bodo pa uporabljali informacije, ki jih bo sistem zagotavljal ali pa imajo večji finančni ali drug interes za uspeh projekta.

Posebno pozornost velja nameniti še dvema skupinama deležnikov, ki ju ne moremo neposredno brez zadržkov uvrstiti v katero od gornjih kategorij, to sta (Satzinger et al., 2012, str. 45):

1. sponzor oziroma naročnik (angl. *client*), to je posameznik ali skupina, ki zagotavlja sredstva za izvedbo projekta in
2. tehnični kader organizacije, to so strokovnjaki informacijsko komunikacijske tehnologije, in kader za podporo uporabnikom sistema.

Sponzorju je treba obdobno poročati o stanju na projektu. Sponzor je lahko posamezen član managementa organizacije, ali pa je to posebna skupina, na primer projektni svet. Sponzor običajno tudi potrjuje zaključke faz projekta in odobrava črpanje finančnih sredstev za projekt (Satzinger et al., 2012, str. 45).

Tehnični kader oziroma strokovnjaki informacijsko komunikacijske tehnologije organizacije skrbijo za računalniško okolje organizacije. Pri projektih razvoja in uvedbe novih informacijskih rešitev v organizaciji vedno sodelujejo. Sodelujejo pri odločanju glede izbire računalniške platforme, programskega jezika in orodij, omrežij, so vir informacij o tehničnih vidikih sistemov, ki so že v uporabi. Kader za podporo uporabnikom je vir informacij o težavah s trenutno delujočimi sistemi (Satzinger et al., 2012, str. 45).

1.3.3 Sistemski analitiki

Pri izgradnji informacijskega sistema imajo pomembno vlogo **sistemski analitiki**. Ti morajo predvsem razumeti potrebe in vizijo sponzorjev projekta. Biti morajo sposobni to vizijo pretvoriti v načrt informacijskega sistema do takšnih podrobnosti, da lahko programerji sistem izdelajo. Bistvo sistemske analize in načrtovanja (angl. *systems analysis & design*, v nadaljevanju SA&D) je v tem, da razvijalcu informacijskega sistema ponuja orodja in tehnike, z uporabo katerih je možno razumeti poslovne potrebe, zabeležiti vizijo, definirati rešitev, vizijo in rešitev komunicirati, zgraditi rešitev in usmerjati druge pri gradnji rešitve, potrditi, da rešitev ustreza zahtevam in vpeljati rešitev v uporabo. To so veščine, koraki, vodila in orodja, ki podpirajo in vodijo do dejanskega programiranja sistema (Satzinger et al., 2012, str. 5).

Rezultate analize analitiki predstavijo tudi z modeli. Modeli so grafična predstavitev obravnavanega koncepta ali procesa. Tako poznamo poslovni model ali model zahtev, model podatkov, model objektov, model omrežja, procesni model (Shelly et al., 2008, str. 16).

Sistemska analiza so vse tiste aktivnosti, ki analitiku omogočijo razumeti in opisati zahteve. To je mnogo več kot le definicija problema. Vse funkcije sistema je potrebno tako

dobro razumeti, da jih lahko opišemo do podrobnosti natančno (Satzinger et al., 2012, str. 5). Med aktivnosti sistemske analize sodijo predvsem zbiranje podrobnih informacij, podrobna definicija zahtev za nov sistem, ugotavljanje prioritet zahtev, razvoj uporabniških dialogov in ocenjevanje zahtev z uporabniki sistema (Satzinger et al., 2012, str. 38).

Zbiranje informacij poteka predvsem na dva načina: s pogovori z uporabniki in z opazovanjem uporabnikov pri opravljanju njihovega dela. Dodatne informacije analitiki pridobijo tudi s preučevanjem razpoložljive dokumentacije in s preučevanjem delovanja in dokumentacije obstoječega informacijskega sistema. Velikokrat si analitiki pomagajo tudi z vpogledom v dokumentacijo ali delovanje informacijskih rešitev drugih ponudnikov. Dober sistemski analitik po končani sistemski analizi postane strokovnjak za obravnavano poslovno področje obravnavne organizacije (Satzinger et al., 2012, str. 38).

Tehnike zbiranja informacij o podrobnih zahtevah so predvsem pogovori z uporabniki in z drugimi deležniki, razdeljevanje in zbiranje vprašalnikov, pregledovanje vhodov, izhodov in dokumentacije, opazovanje in dokumentiranje poslovnih procesov, raziskovanje rešitev na trgu in zbiranje pripomb in predlogov aktivnih uporabnikov. Večinoma sistemski analitiki uporabljajo neko svojo kombinacijo teh metod. (Satzinger et al., 2012, str. 46).

Svoja spoznanja sistemski analitiki pogosto interpretirajo z **modeli.** Model je interpretacija pomembnega dela realnega sveta. Pogosto rečemo, da gre za abstrakcijo, saj na modelu prikažemo le tiste karakteristike realnega sveta, ki so bistvene za obravnavano tematiko. Modeli so lahko grafični ali pa matematični. Pri analizi in razvoju informacijskih sistemov večinoma uporabljamo grafične modele. Izbira načina modeliranja je povezana tudi z možnostmi razvojnih orodij, ki jih imamo pri delu na razpolago. Modeli, ki jih najpogosteje rišejo sistemski analitiki, so (Satzinger et al., 2012, str. 217):

1. diagram poteka (angl. *flowchart*),
2. diagram toka podatkov (angl. *data flow diagram*, v nadaljevanju DFD),
3. diagram entitet povezav (angl. *entity relationship diagram*, v nadaljevanju E-R diagram),
4. strukturni diagram (angl. *structure diagram*),
5. diagram razredov (angl. *class diagram*).

Podrobna definicija zahtev za nov sistem obsega tako definicijo funkcionalnih zahtev, torej opredelitev tega, kar mora sistem delati, kot tudi nefunkcionalnih zahtev. Pod pojmom nefunkcionalne zahteve razumemo (Satzinger et al., 2012, str. 41):

1. uporabnost, torej enostavnost uporabniškega vmesnika,
2. zanesljivost, kamor štejemo pogostost odpovedi sistema in postopke za povrnitev v uporabno stanje,
3. odzivnost sistema, kjer opredelimo zahtevane odzivne čase in prepustnost sistema,

4. varnostne zahteve, kot sta kontrola dostopa in kriptiranje in
5. omejitve pri načrtovanju, kjer upoštevamo omejitve izbrane strojne opreme, podporne programske opreme, razvojnih orodij, formatov za izmenjavo podatkov in podobno.

Načrtovanje sistema so vse tiste aktivnosti, ki omogočijo definirati in opisati sistem, ki bo predstavljal rešitev problema. V koraku načrtovanja povemo, kako bo novi sistem deloval. Do podrobnosti definiramo sestavne dele sistema in njihovo medsebojno delovanje (Satzinger et al., 2012, str. 5).

1.3.4 Strukturni in objektno orientirani pristop

Razvoj informacijskih sistemov poteka na različne načine. Pogosto imajo tudi v isti organizaciji različne razvojne ekipe vsaka svoj pristop k razvoju. V splošnem pa ločimo dva pristopa k izgradnji in modeliranju (Satzinger et al., 2012, str. 218):

1. strukturni pristop in
2. objektno orientirani pristop.

Strukturni pristop je tradicionalen v praksi preizkušen pristop, ki ga je lahko razumeti. Po vrsti izvajamo zaporedje faz razvoja, ki ga imenujemo življenjski cikel razvoja sistema. Podrobneje bomo življenjski cikel opisali v naslednjem razdelku. Strukturna analiza se je pojavila in razvila, ko so se računalniški programi izvajali še na osrednjih strežnikih. Še vedno pa ostaja zelo razširjena metoda razvoja. Pri strukturni analizi se uporablja zbirko procesnih modelov, s katerimi se grafično prikaže sistem. Pristop se osredotoča na procese, ki iz podatkov tvorijo koristne informacije. Zato ga imenujemo tudi procesno orientirani pristop. Pri tem pristopu procese in podatke obravnavamo ločeno. Strukturni pristop sestavljajo tri tehnike: strukturno načrtovanje, strukturna analiza in strukturno programiranje (Shelly et al., 2008, str. 18).

Strukturno programiranje ima za rezultat program, ki ima en začetek in en konec, vsak korak pri izvajanju programa pa pomeni enega izmed treh gradnikov (Satzinger et al., 2012, str. 219):

1. zaporedje programskih stavkov,
2. točka odločitve, kjer se izvaja en ali drug nabor stavkov in
3. ponavljanje nabora stavkov.

Strukturno načrtovanje je načrtovanje programa, ki ima za cilj organizirati program kot zbirko modulov, ki sestavljajo hierarhično strukturo. Grafično organizacijo modulov prikažemo na strukturnem diagramu. Pri tem načinu načrtovanja mora načrtovalec vnaprej vedeti, kaj bo sistem delal, katere so glavne funkcije sistema, kateri so potrebni podatki in kakšne informacije mora sistem nuditi uporabnikom (Satzinger et al., 2012, str. 220).

Strukturna analiza je predhodni korak strukturnega načrtovanja. V tem koraku se ne ukvarjamo s tehnologijo, ki bo omogočila postavitev in delovanje informacijskega sistema, ampak se osredotočamo na ugotavljanje potrebne funkcionalnosti bodočega sistema. Ugotavljamo, kaj mora sistem delati (procesne zahteve), katere podatke je treba shranjevati in uporabljati (podatkovne zahteve) kateri vhodi v sistem in izhodi iz sistema so potrebni in kako morajo funkcije medsebojno sodelovati, da bo sistem dosegel svoj namen. Osnovni grafični model strukturne analize je diagram toka podatkov toka podatkov (DFD). Na tem diagramu so prikazani vhodi, shrambe podatkov, procesi in izhodi iz sistema in njihovo medsebojno delovanje. Novejši pristop k strukturni analizi vključuje tudi analizo vseh dogodkov, ki privedejo do določenih reakcij sistema. Izdela se tudi model podatkov. Najpogosteje je to E-R diagram (Satzinger et al., 2012, str. 221).

Na Sliki 3 je prikazan odnos med strukturno analizo, strukturnim načrtovanjem in strukturnim programiranjem.

Objektno orientirani pristop je novejši. Informacijski sistem vidi kot zbirko medsebojno delujočih objektov, ki sodelujejo, da opravljajo določene naloge (Satzinger, 2012, str. 223). Objektno orientirana analiza podatke in procese, ki na podatkih delujejo, kombinira v stvari, imenovane objekti. Rezultat po tem načelu izvedene sistemske analize je množica programskih objektov, ki predstavljajo objekte iz realnega sveta, na primer ljudi, stvari, dogodke. Programerji z uporabo objektno orientiranih jezikov pišejo programsko kodo, ki kreira te programske objekte. Objekti so člani razredov. Razred je zbirka podobnih objektov. Objekt ima karakteristike, ki jih imenujemo lastnosti. Lastnosti podeduje od svojega razreda ali pa ima svoje lastne. V objektnem svetu procesom pravimo metode. Metode lahko spreminjajo lastnosti objektov. Objekt lahko pošlje informacijo drugemu objektu preko sporočila (Shelly et al., 2008, str. 18).

1.4 Življenjski cikel razvoja informacijske rešitve

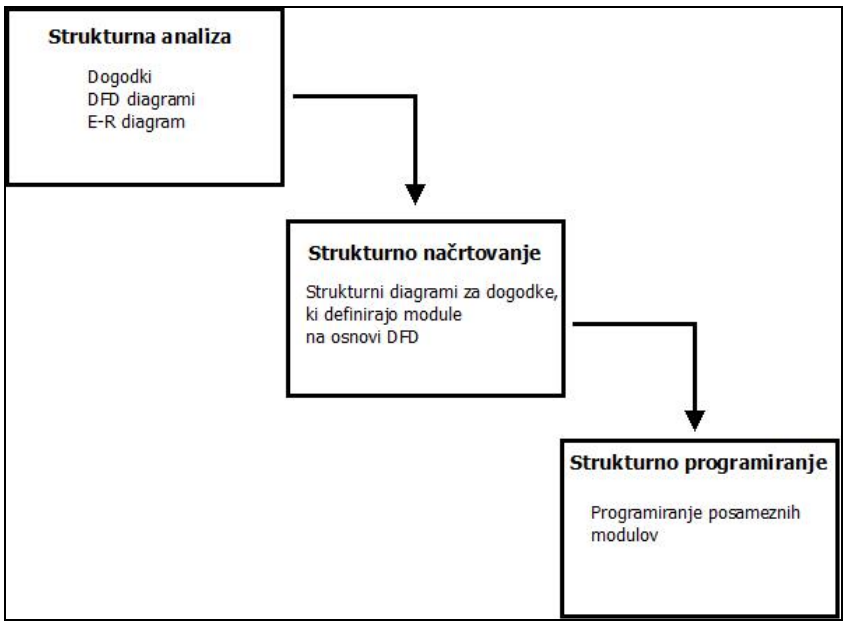
Veliko je različnih pristopov k izgradnji informacijskih sistemov. Vedno pa je tako ali drugače treba izvesti šest osnovnih korakov (Satzinger et al., 2012, str. 6):

1. začetne aktivnosti: identifikacija problema ali potrebe po novem informacijskem sistemu, priprava začetnih dokumentov in pridobitev soglasja za začetek projekta,
2. načrtovanje in spremljanje realizacije projekta,
3. podrobna analiza zahtev,
4. načrtovanje komponent novega sistema,
5. izdelava, testiranje in integracija izdelanih programskih sklopov in
6. izvedba testov celotnega sistema in uvedba izdelane rešitve v uporabo.

Stroški, ki jih povzročajo napake, storjene v prvih treh stopnjah, so lahko zelo veliki, saj zgodaj narejene napake vplivajo na delo v vseh nadaljnjih korakih. Poenostavljeno lahko

sklepamo, da bolj zgodaj ko naredimo napako, več nas bo ta stala. V praksi faze razvoja ne potekajo vedno strogo zaporedno, ampak se je ob odkritju napak pogosto potrebno vračati nazaj.

Slika 3: Strukturna analiza, strukturno načrtovanje in strukturno programiranje



Vir: J. W. Satzinger, R. B. Jackson, & S. D. Burd, Introduction to systems analysis and design: an agile, iterative approach, 2012, str. 223, slika 8-17.

Programiranje je izdelava programskih sklopov. V preteklosti smo programerji programsko kodo res pisali vrstico za vrstico. Orodje za pisanje kode je bil le enostaven urejevalnik besedila, na primer Notepad. Sintaktične napake so prvič lahko opozorile nase šele z napačnim delovanjem programa pri prvem testnem zagonu programske kode. Iskanje vzrokov za napačno delovanje je bilo lahko zelo zamudno. Ni bilo orodij za razhroščevanje, to je iskanje napak. Ob pojavu težav je bilo treba v kodo s premislekom ročno dodati izpise vrednosti posameznih spremenljivk in prekinitve programa, da smo se lahko s koračnim izvajanjem kode približali vzroku za nedelovanje ali napačno delovanje.

Sčasoma so se razvila orodja za pomoč razvijalcem programske opreme. Zanje se je uveljavilo poimenovanje s kratico CASE, ki v angleščini pomeni *computer-aided software engineering*, torej računalniško podprto načrtovanje programske opreme. Samo programiranje so močno pospešili tudi sodobni programski jeziki. Vendar pa to ni bistveno skrajšalo razvoja obsežnih projektov, saj programiranje ne pomeni večine potrebnega dela, ampak le okrog ene petine (Gradišar et al., 2012, str. 169).

Sodobna orodja razvijalcem zelo olajšajo delo. Po eni strani razvijalca vodijo skozi sam postopek razvoja. Manj je tudi neposrednega pisanja programske kode. Orodja imajo

grafičen vmesnik za delo. Marsikakšen del kode se zgenerira na naš ukaz ali samodejno v ozadju, medtem ko z miško rišemo po ekranu. Kadar pa neposredno pišemo kodo, pa jo pišemo v urejevalniku, ki nam že med pisanjem sproti predlaga sintakso, ki nas opozarja na sintaktične napake v napisanih ukazih. Lahko rečemo, da urejevalnik »zna« programski jezik, v katerem pišemo kodo. Orodja omogočajo, da lahko delamo na več delih kode hkrati, kodo imamo odprto na več zavihkih ali v več oknih.

Možni so štirje osnovni načini uvajanja novega sistema v uporabo (Shelly et al., 2008, str. 482):

1. neposreden prehod na nov sistem,
2. vzporedno delovanje starega in novega sistema,
3. pilotska uvedba novega sistema v enem delu organizacije in
4. postopna uvedba novega programa po posameznih modulih.

Neposredni prehod pomeni hkratno prenehanje delovanja starega sistema in takojšnji začetek dela na novem sistemu. Neposredni prehod na nov sistem je običajno najcenejši način od navedenih, saj morajo informatiki skrbeti le za en delujoči sistem. Ni podvojenega delovanja. Vendar pa je ta način bolj tvegan od drugih. Ne glede na to, kako pazljivo in temeljito smo pred uvedbo pregledali in testirali nov sistem, se nam vedno lahko zgodi, da se določene pomanjkljivosti ali napake odkrijejo šele med delovanjem sistema. Težave se lahko pojavijo le pri določenih kombinacijah podatkov. Vseh kombinacij pa pred uvedbo ni možno testirati. Drugačne težave se lahko pojavijo šele pri velikih količinah podatkov, česar se pa pogosto tudi ne more testirati vnaprej. Organizacije se za neposredni prehod pogosto odločajo pri uvajanju komercialnih programskih rešitev, ki so že v uporabi pri drugih odjemalcih dobavitelja programske opreme. Za unikatne doma izdelane programske rešitve pa se neposredni prehod uporablja bolj pri sistemih, ki niso ključni za delovanje organizacije. Včasih pa kapacitete organizacije ne dopuščajo hkratnega delovanja starega in novega sistema. V takšnih okoliščinah smo ne glede na prisotna tveganja prisiljeni izvesti neposredni prehod. Čas prehoda iz starega na nov sistem je koristno načrtovati z upoštevanjem intenzivnosti uporabe sistema. Večina poslovnih informacijskih sistemov deluje v ciklih. Prehod je modro načrtovati na začetku cikla. Tako na primer je računovodski informacijski sistem modro menjati tik po končanih aktivnostih zaključnega računa in ne sredi poslovnega leta (Shelly et al., 2008, str. 483).

Podpora tekočemu delu je naloga informatikov. Skrbijo za tehnično brezhibnost sistema in pomagajo uporabnikom, da sistem uporabljajo pravilno in učinkovito.

Vzdrževanje pomeni prilagajanje informacijskega sistema na spremembe v sami organizaciji ali uvajanje izboljšav glede na ugotovitve pri uporabi sistema. Vzdrževanje se stalno izvaja ves čas uporabe sistema.

Pogosto v literaturi najdemo razdelitev razvojnih aktivnosti v sedem glavnih skupin (Satzinger et al., 2012, str. 210):

1. iniciacija projekta,
2. načrtovanje projekta,
3. analiza,
4. načrtovanje sistema,
5. izdelava sistema,
6. uvedba v uporabo in
7. podpora delovanju.

Opisali smo življenjski cikel informacijskega sistema od zamisli o novem sistemu do njegove uporabe. Kadar sistem razvijamo v tem vrstnem redu in se aktivnosti odvijajo strogo po navedenem vrstnem redu in se ne prekrivajo, pravimo, da sistem razvijamo **po metodi življenjskega cikla sistema** (angl. *system development life cycle*, v nadaljevanju SDLC).

Metoda je primerna za razvoj večjih sistemov, ki jih bomo uporabljali dalj časa. Pri izgradnji velikega sistema je zelo pomembno, da vse delo skrbno načrtujemo, izvajamo po načrtu in dosledno dokumentiramo. Pri delu sodeluje veliko ljudi in s skrbno načrtovanim in spremljanim delom poskušamo čim bolj zmanjšati možnost napak (Gradišar et al., 2012, str. 171).

Metoda življenjskega cikla predpostavlja, da bo razvojna ekipa sposobna v fazi analize zahtev opredeliti vsebino sistema do vseh podrobnosti in da se zahteve glede vsebine sistema do uvedbe sistema v uporabo ne bodo spreminjale. Dejansko pa se poslovno okolje v današnjem dinamičnem svetu zelo hitro spreminja. Zato je le malo verjetno, da se tekom razvoja ne bodo pojavile realne nove potrebe, ki bi jih bilo smiselno upoštevati pri izgradnji informacijskega sistema (Gradišar et al., 2012, str. 172).

Upoštevati pa je treba tudi v praksi opaženo dejstvo, da dostikrat člani razvojne ekipe niso sposobni vnaprej identificirati vseh potrebnih značilnosti novega sistema. Včasih se uporabniki lahko glede teh dokončno opredelijo šele tekom razvoja, ko sodelujejo pri testiranju delov sistema ali celo šele takrat, ko testirajo sistem kot celoto. Možno je tudi, da prihaja do nerazumevanja glede potreb med uporabniki in sistemskimi analitiki zaradi težav pri komunikaciji. Takšni nesporazumi, ki imajo za posledico napačno izdelan sistem, se lahko v najslabšem primeru odkrijejo šele takrat, ko je sistem že dokončno izdelan.

1.5 Metodologije razvoja informacijskih sistemov

1.5.1 Metodologija

Po slovarju slovenskega knjižnega jezika je **metodologija** skupek metod (Metodologija, 2016). **Metoda** pa je oblika načrtnega, premišljenega dejanja, ravnanja ali mišljenja za dosego cilja, tudi način ali postopek (Metoda, 2016).

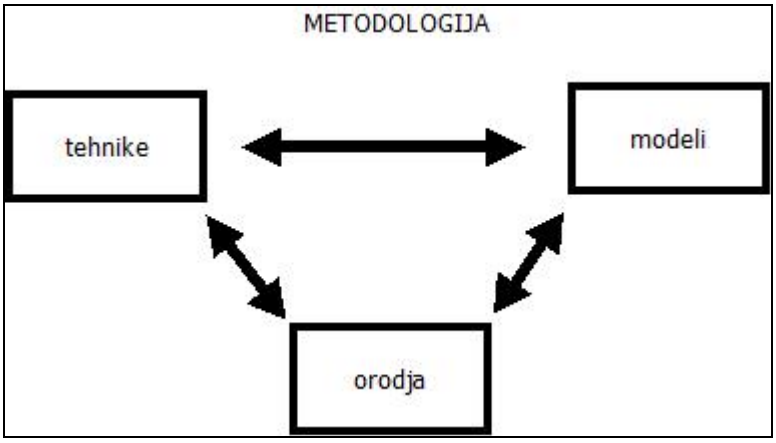
Metodologija razvoja informacijskega sistema zagotavlja smernice za vse vidike življenjskega cikla razvoja informacijskega sistema. Definira življenjski cikel razvoja, torej faze in korake in njihovo zaporedje, kot tudi modele, orodja in tehnike, ki naj se jih pri razvoju poslužujemo. Shematsko je takšno pojmovanje metodologije prikazano na Sliki 4. Primeri modelov, katerih uporabo nam predpisuje metodologija, so določeni diagrami, na primer diagram poteka, diagram entitet in povezav, procesni diagram. Z modeli povezane so tehnike, torej navodila, kako naj delo opravimo. Primer takšne tehnike so smernice za izvedbo intervuja z uporabnikom. Orodja za delo so na primer pripomočki za zapisovanje ugotovitev, računalniški programi za izdelavo modelov in programerska orodja (Satzinger et al., 2012, str. 215).

Bajec in Krisper navajata, da so prve elaborirane metodologije razvoja informacijskih sistemov nastale v zgodnjih sedemdesetih letih dvajsetega stoletja, ko so na področju razvoja programske opreme vladali Codd, DeMarco in drugi. Strokovno javnost so uspeli prepričati, da je potrebno tudi za področje razvoja programske opreme urediti postopke, predpisati metode, tehnike in orodja. S tem so ovrgli smiselnost uporabe ad hoc pristopov in postavili temelje za obdobje birokracije, kot ga danes nekateri imenujejo. Z željo po standardizaciji postopkov in izdelkov so nastali priročniki za razvoj informacijskih sistemov. Nekateri so bili tako obsežni, da so na več tisoč straneh predpisovali vsak najmanjši korak razvoja, za vsako aktivnost so opredelili številne dokumente in obrazce, za izvedbo so bila določena orodja in navodila za uporabo. Pri tako predpisanem načinu dela so udeleženci na projektu obremenjeni poleg svojega primarnega dela še s skrbjo za številne izdelke in naloge, ki se vsaj na videz zdijo sekundarnega pomena (Bajec & Krisper, 2003, str. 68).

Metodologija je prisotna pri vsakem organiziranem delu. Metodologija zajema vse, kar redno počnemo, da bi dosegli želen rezultat. V primeru razvoja programske opreme to pomeni tako postopke, ki so neposredno povezani z razvojem, kot tudi podporne postopke, komunikacijo med sodelujočimi, pravila odločanja in podobno. Z razvojem neposredno povezani postopki so analiza zahtev, načrtovanje sistema, itd. **Metodologijo** lahko opredelimo kot množico dogovorov oziroma konvencij, s katerimi se projektna skupina ali organizacija strinja (Bajec & Krisper, 2003, str. 69).

Navedena avtorja ugotavljata, da je metodologija prežeta s filozofijo, načeli, idejami in pogledi organizacije in njenih članov. S tem posebej poudarjata njeno sociološko komponento. Obstajajo številne formalne metodologije, ki temeljijo na postopkih, ki so se v praksi izkazali kot dobri. Organizacije, ki jih prevzamejo, pa si jih vedno prilagodijo tako, da ustrezajo njihovemu načinu dela (Bajec & Krisper, 2003, str. 69).

Slika 4: Metodologija



Vir: J. W. Satzinger, R. B. Jackson, & S. D. Burd, Introduction to systems analysis and design: an agile, iterative approach, 2012, str. 215, slika 8-8.

Metodologija zajema formalno opredeljene elemente, to so postopki, pravila, napotki, smernice, standardi, zapisani v priročnikih za delo. Vsaka metodologija pa ima tudi številne nedokumentirane elemente, še posebnega pomena je tu znanje, ki ga člani organizacije uporabljajo pri svojem delu. Ravno neformalni del metodologije tisti, ki je za organizacijo ključnega pomena, saj predstavlja tiste vrednote, ki so organizaciji lastne in so njena konkurenčna prednost. Temeljni elementi metodologije se skrivajo ravno v znanju in izkušnjah posameznikov. Znanje posameznikov je lahko eksplicitno, ki ga je možno izraziti v formalni obliki, ali pa skrito znanje, ki je navadno subjektivne narave in prepleteno z izkušnjami, čustvi, intuicijo, in ga je zato težko izraziti, povzemata avtorja ugotovitve s področja upravljanja znanja po avtorju Nonaka (Nonaka, 1991). Z uporabo se metodologija bogati s skritim znanjem posameznikov (Bajec & Krisper, 2003, str. 69).

Številna računalniška podjetja še vedno razvijajo programsko opremo na podlagi neformalno opredeljenih metodologij, ki niso dokumentirane. Postopki so znani in ustaljeni, a jih podjetja redko eksplicitno zapišejo, še bolj poredko pa svoje zapise osvežujejo v skladu s pridobljenimi izkušnjami in znanjem. Obsežna neformalna metodologija je zelo tvegana za organizacijo, saj je popolnoma odvisna od njenih uporabnikov. Če je metodologija le neformalno opredeljena, je razvoj težko standardizirati, postopek lahko postane nepregleden in neobvladljiv (Bajec & Krisper, 2003, str. 70).

Primer obsežne formalizacije razvoja informacijskega sistema je **mednarodni standard za področje razvoja informacijskih sistemov,** ki ga je izdala Mednarodna organizacija za standardizacijo ISO (angl. *International Organization for Standardization*). To je standard z oznako ISO/IEC 12207:2008 in angleškim imenom *Systems and software engineering – Software life cycle processes*.

Glavni namen navedenega standarda je opredelitev neke skupne strukture, tako da bi imeli kupci, dobavitelji, razvijalci, vzdrževalci in managerji skupen jezik, kadar bi se pogovarjali o razvoju informacijskega sistema. Standard ISO/IEC 12207 delo na življenjskem ciklu informacijskega sistema razdeli na dve skupini procesov (ISO, 2008):

1. procesi življenjskega cikla sistema in
2. specifični procesi programske opreme.

Procesi življenjskega cikla sistema se nanašajo na:

1. vzpostavitev projekta,
2. oddajo naročila,
3. sklepanje pogodbe,
4. izvajanje pogodbe,
5. vodenje projekta in
6. tehnična opravila.

Pod tehnična opravila tu sodijo:

1. definicija zahtev deležnikov projekta,
2. sistemska analiza zahtev,
3. načrtovanje arhitekture sistema,
4. implementacija zahtev oziroma izgradnja programske kode,
5. integracija sistema,
6. testiranje ustreznosti sistema,
7. namestitev sistema,
8. podpora preverjanju ustreznosti,
9. delovanje,
10. vzdrževanje in
11. upokojitev oziroma umik iz uporabe.

Specifični procesi programske opreme po tem standardu so:

1. procesi implementacije, ki obsegajo tudi analizo zahtev, načrtovanje programske opreme, programiranje, integracijo ter testiranje,

2. procesi podpore, ki obsegajo upravljanje dokumentiranja programov, upravljanje konfiguracij, zagotavljanje kakovosti, verifikacijo, validacijo, pregledovanje, revidiranje ter reševanje problemov in
3. procesi ponovne uporabe programske opreme.

S pojmom verifikacija tu označujemo preverjanje produkta faze z zahtevami, opredeljenimi v predhodni fazi. S pojmom validacija označujemo vrednotenje produkta na koncu razvoja, z namenom ugotovitve skladnosti z zahtevami. (Dogša, 1993).

Vsak od procesov ima opredeljen nabor aktivnosti in nalog, ki jih je treba izvesti. Prav tako je naveden seznam izdelkov, ki naj bodo rezultat procesa.

V Republiki Sloveniji je bivši Center Vlade RS za informatiko vzpostavil Enotno metodologijo za razvoj informacijskih sistemov, v nadaljevanju **EMRIS**. Izdal je priročnik, katerega cilj je predstavitev postopkov in tehnik, ki jih metodologija EMRIS priporoča za strateško planiranje in razvoj informacijskih sistemov. Metodologija se nanaša tako na strateško planiranje kot tudi na razvoj informacijskih sistemov po strukturnem pristopu in objektnem pristopu (Vlada RS, 2000).

V grobem ločimo dva **pristopa k načrtovanju in vodenju projektov izgradnje** informacijskih sistemov (Satzinger et al., 2012, str. 209):

1. **prediktivno ali napovedno programiranje**: sloni na previdevanju, da lahko projekt v celoti načrtujemo vnaprej in da bomo nov informacijski sistem lahko razvili v skladu z začetnim načrtom. Primerno je za razvoj sistemov, ki jih dobro razumemo in so dobro definirani, na primer prevedba obstoječega sistema na novo tehnologijo, in
2. **adaptivni ali prilagodljivi pristop**: upošteva, da mora biti projekt bolj fleksibilen (prilagodljiv) in se mora spreminjati v skladu s spremembami uporabniških zahtev v času izvajanja projekta. Pristop se uporablja v razmerah, ko ne razumemo popolnoma lastnosti novega sistema in ko uporabniške zaheve niso popolnoma definirane. Nekatere zahteve bodo znane šele po prvi fazi razvoja.

1.5.2 Prediktivni razvojni pristopi

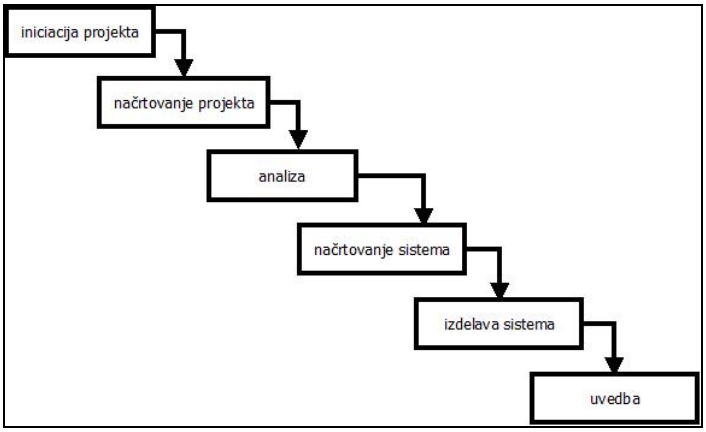
Zaporedni ali slapovni model ali metoda življenjskega cikla sistema (angl. *system development life cycle*, v nadaljevanju SDLC), imenujemo ga tudi linearno-sekvenčni model življenskega cikla. Faze razvoja se izvajajo strogo zaporedno, nobena faza se ne sme začeti, preden ni zaključena predhodna faza. Korake smo navedli v prejšnjem razdelku. Shematsko je prikazan na Sliki 5.

V-model je modificirana oblika SDLC modela razvoja. Za vsako od faz definiramo tudi testne postopke. Načrtovanje testiranja poteka vzporedno z vsako fazo razvoja. Ko je

pisanje programov končano, se začne testiranje, ločeno po fazah (Tutorialspoint, 2016). Shema modela je prikazana na Sliki 6.

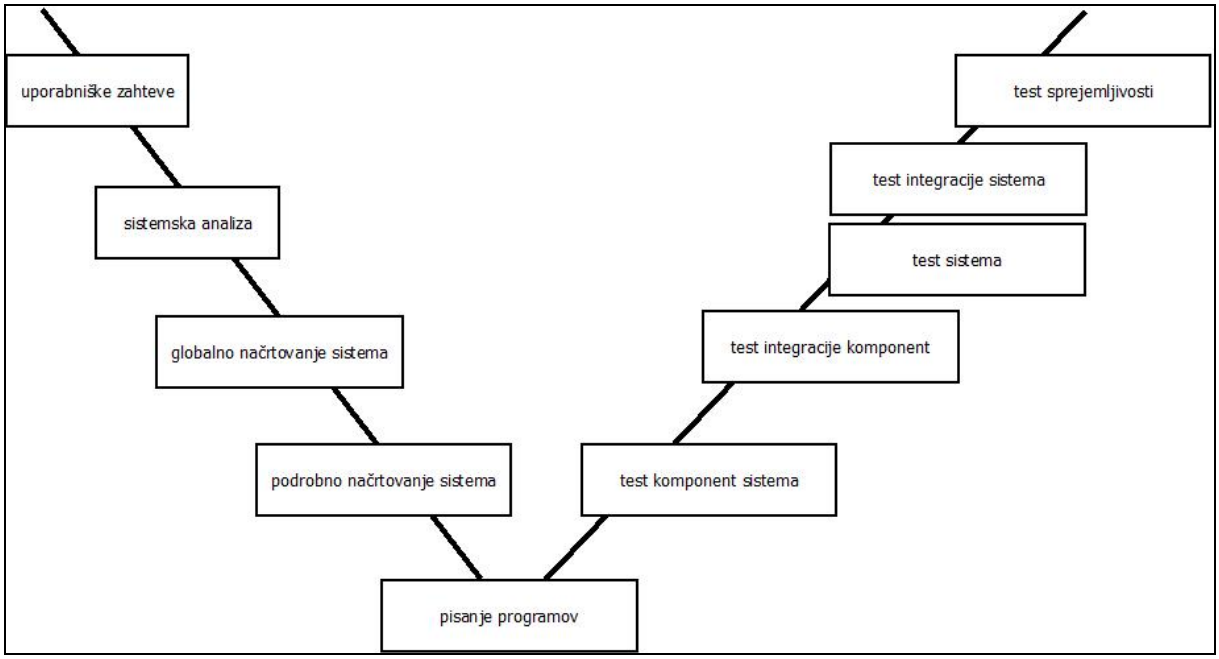
W-model je uvedel Paul Herzlich. Še vedno gre v osnovi za SDLC metodo, ob kateri se za vsako fazo načrtuje še testiranje. Metoda terja več testiranja kot pa pri V-modelu.

Slika 5: Metoda SDLC



Vir: J. W. Satzinger, R. B. Jackson, & S. D. Burd, Introduction to systems analysis and design: an agile, iterative approach, 2012, str. 211, slika 8-3.

Slika 6: V-model



Vir: Software testing times, V-model is a basis of structured testing, 2010.

1.5.3 Adaptivni razvojni pristopi

Pri iterativnem pristopu k razvoju informacijskega sistema le-ta raste postopoma. Najprej se razvijejo osnovne komponente sistema, nato pa se postopoma dodajajo še dodatne funkcionalnosti. Ta pristop imenujemo iterativni zato, ker se osnovnih šest korakov razvoja (iniciacija projekta, načrtovanje projekta, sistemska analiza, načrtovanje sistema, izdelava sistema, uvedba) ponavlja znova in znova, ob vsaki iteraciji se produktu doda nekaj nove funkcionalnosti. Delo na razvoju je organizirano kot en sam velik projekt, ki ga sestavlja veliko malih projektov. Vsaka iteracija je projekt zaradi tega, ker je njen rezultat delujoč končni izdelek in ima vnaprej omejen čas za realizacijo (Satzinger et al., 2012, str. 8). Primera iterativnega modela razvoja sta spiralni model in IBMjev RUP.

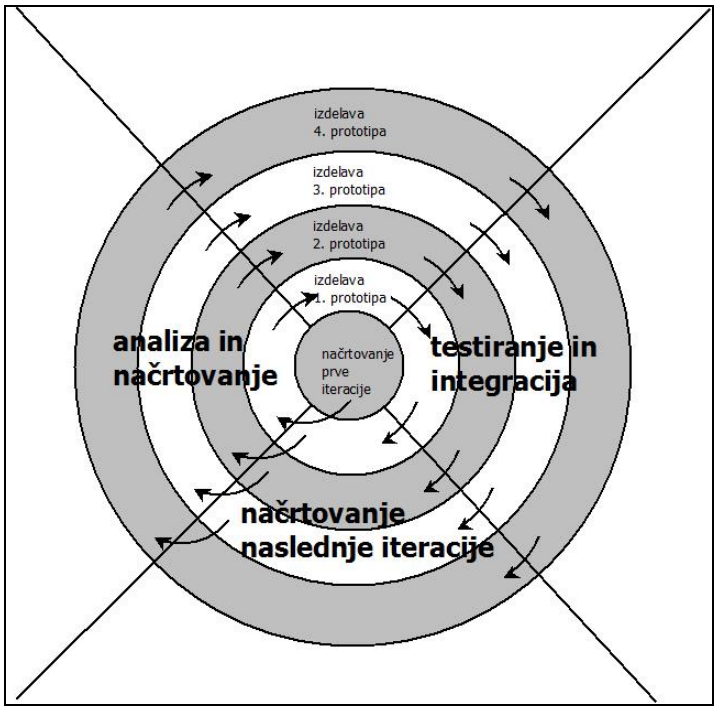
Spiralni model je utemeljil Barry Boehm leta 1986. V osnovi je to iterativni model, usmerjen k identifikaciji in zmanjšanju tveganj s kakršnimikoli primernimi pristopi. Tipičen primer pa je prototipiranje (Agile Alliance, 2016). Spiralni model velja za enega od prvih pristopov, kjer projekt prilagajamo glede na rezultate prejšnje iteracije projekta. Na življenjski cikel gledamo kot na spiralo, ki izhaja iz središča in se odvija navzven, vsak polni zavoj predstavlja eno iteracijo. V vsaki iteraciji izvajamo najprej načrtovanje iteracije (določimo cilje in načrt dela za to iteracijo), nato analizo in načrtovanje sistema, nato izdelavo programa, da dobimo delujoč izdelek, ki ga potem testiramo in integriramo (Satzinger et al., 2012, str. 212). Shema spiralnega modela je prikazana na Sliki 7.

Objektna metodologija IBM z imenom **Rational Unified Process** (v nadaljevanju **RUP**) je opredelitev načina razvoja programske opreme, do katere lahko dostopamo preko skupne baze znanja. Vsi člani razvojne ekipe imajo na voljo isto bazo znanja. Na voljo so priporočila, predloge in orodja. V bistvu gre za priporočila in navodila, kako učinovito uporabljati poenoteni jezik modeliranja (angl. *unified modeling language*, v nadaljevanju UML). UML je standardiziran jezik, ki omogoča jasno komunikacijo zahtev, arhitekture in načrtovanja sistema. Pri razvoju informacijskega sistema metodologija priporoča šest najboljših praks in zanje definira tudi načine, kako jih izvajati (Rational, 1998, str. 2) :

1. Programsko opremo razvijamo v iteracijah, rezultat vsake iteracije je delujoča programska oprema, ki pokriva del funkcionalnosti,
2. upravljamo zahteve, kar pomeni, da metoda predpisuje načine za evidentiranje in spremljanje definiranih zahtev, njihove realizacije in sprejetih odločitev v zvezi z njimi,
3. uporabljamo komponentno arhitekturo, komponente so netrivialni moduli, podsistemi, ki izvajajo določeno funkcijo,
4. vizualno modeliramo programsko opremo, osnova za uspešno vizualno modeliranje je standard UML,
5. preverjamo kvaliteto programske opreme in
6. nadziramo spremembe programske opreme.

Vsak cikel razvoja ima za rezultat novo generacijo programskega proizvoda. Razvojni cikel je razdeljen na štiri zaporedne faze. Vsaka od faz se zaključi z jasno opredeljenimi mejniki.

Slika 7: Spiralni model razvoja



Vir: J. W. Satzinger, R. B. Jackson, & S. D. Burd, Introduction to systems analysis and design: an agile, iterative approach, 2012, str. 212, slika 8-4.

1.5.4 Skupni in hitri razvoj aplikacij

Že precej pred uveljavitvijo adaptivnih pristopov k razvoju informacijskih sistemov so se informatiki zavedali, da je zelo tvegano razvijati informacijske sisteme brez zadostnega sodelovanja uporabnikov. Sčasoma so zato v razvojne ekipe začeli poleg tehničnega kadra vključevati tudi uporabnike in vodstvene delavce. Tako oblikovane ekipe so lahko delale bolj uspešno in hitreje prišle do rezultatov. Popularna sta postala dva načina (Shelly et al., 2008, str. 24):

1. skupni razvoj aplikacij (angl. *joined application development,* v nadaljevanju JAD) in
2. hitri razvoj aplikacij (angl. *rapid application development*, v nadaljevanju RAD).

Skupni razvoj aplikacij se osredotoča na skupno ugotavljanje dejstev. Združena ekipa uporabnikov, vodstvenih delavcev in informatikov deluje skupaj le v eni fazi razvoja, pri zbiranju uporabniških zahtev. To ni metodologija, je bolj tehnika dela. Uporabniki imajo

tu veliko bolj aktivno vlogo kot pa pri klasičnih pristopih, ko so večinoma le odgovarjali na vprašanja in pregledovali izdelane programe. Delo poteka v skupini, vsi člani sedijo za isto mizo. Skupina ima vodjo, ki usmerja delo, ter enega ali več članov, ki imajo nalogo, da zapisujejo ugotovitve in odločitve. Sistemski analitiki sodelujejo v razpravi, postavljajo vprašanja, zapisujejo ugotovitve in nudijo podporo delu skupine. Če je na voljo primerno CASE orodje, lahko sistemski analitiki neposredno ob delu v skupini izdelajo modele in dokumentacijo. Takšen način dela terja od vseh udeležencev intezivno sodelovanje in je zanje precej naporen (Shelly et al., 2008, str. 95).

Hitri razvoj aplikacij je za razliko od JAD metodologija. Tako kot JAD uporablja skupinsko delo, a gre precej dlje. Končni izdelek tu ni opredelitev uporabniških zahtev, ampak je nov informacijski sistem. Pristop RAD je razvit z namenom zmanjšanja stroškov in časa razvoja. S tem naj bi se povečala verjetnost za uspeh projekta. Metodologija intenzivno uporablja prototipiranje in aktivno vlogo uporabnikov. Spodbuja se čimprejšnja izgradnja prototipa, na osnovi katerega uporabniki podajo zahteve za potrebne spremembe. Z uporabo CASE orodij se gradijo prototipi in se kreira dokumentacija. Delo poteka v štirih fazah (Shelly et al., 2008, str. 97):

1. načrtovanje zahtev,
2. uporabniško načrtovanje sistema,
3. konstruiranje in
4. prehod.

Faza načrtovanja zahtev vsebuje elemente načrtovanja sistema in sistemske analize iz pristopa SDLC. Uporabniki, predstavniki vodstva in informatiki skupaj obravnavajo poslovne potrebe, obseg projekta, omejitve in sistemske zahteve. Skupaj sprejemajo odločitve. Faza je končana takrat, ko se člani razvojne ekipe strinjajo glede ključnih zadev in pridobijo soglasje vodstva za nadaljevanje dela.

Uporabniško načrtovanje sistema pomeni sodelovanje uporabnikov pri razvoju modelov in prototipov, ki predstavljajo vse procese sistema, njegove vhode in izhode. Običajno pri tem uporabljajo kombinacijo JAD pristopov in CASE orodij.

Konstruiranje pomeni izdelavo programov in aplikacij. Uporabniki tu še vedno sodelujejo in še vedno lahko predlagajo spremembe, tudi ko so že izdelani uporabniški ekrani in poročila.

Prehod pomeni vse dejavnosti, ki so potrebne, da se sistem uvede v uporabo. V tej fazi se izvaja prevedba podatkov, testiranje, prehod na nov sistem in usposabljanje uporabnikov. V primerjavi s klasičnimi pristopi k razvoju je tu postopek precej strnjen. Posledično je nov sistem zgrajen in predan v uporabo veliko hitreje.

Metodologija hitrega razvoja ima v primerjavi s klasičnimi pristopi strukturne analize svoje prednosti in slabosti. Najpomembnejša prednost je možnost, da sistem lahko razvijemo veliko hitreje in z znatnimi prihranki. Kot slabost viri navajajo dejstvo, da se ta pristop preveč posveča sami mehaniki bodočega sistema, torej načinu delovanja, premalo pa upošteva strateške potrebe organizacije, ki ga bo uporabljala. Zato obstaja večje tveganje, da bo sistem sicer kratkoročno dobro deloval, ne bo pa skladen z dolgoročnimi cilji organizacije. Druga možna slabost pristopa RAD je, da je zaradi pospešenega življenjskega cikla sistema manj časa za posvečanje kakovostnemu razvoju, konsistentnosti in upoštevanjem standardov razvoja. Če se organizacija zaveda možnih tveganj, je lahko RAD dobra izbira (Shelly et al., 2008, str. 99).

1.5.5 Težke in lahke metodologije

Bajec in Krisper (2003, str. 72) po agilnem zavezništvu (Agile Alliance, 2016) povzemata opredelitev **teže metodologije**. To je zmnožek obsega in gostote metodologije. **Obseg metodologije** pove, koliko različnih elementov, to je vlog, izdelkov, priporočil in podobno zaajema metodologija. **Gostota** pa se nanaša na raven podrobnosti, s katero so elementi opisani. RUP velja za težjo metodologijo, saj zajema in podrobno opisuje številne elemente. Tipični predstavniki lahkih metodologij pa so Extreme Programming (XP), Feature-Driven Development (FDD), Crystal Clear, ki jih bomo opisali v naslednjem razdelku.

Agilni razvojni pristopi so se pojavili kot odziv na iskanje ustreznega ravnovesja med formalno in neformalno metodologijo (Bajec & Krisper, 2003, str. 70).

Agilne metodologije razvoja programske opreme so nastale v devetdesetih letih prejšnjega stoletja kot odgovor na "težke metodologije", takšne kot je npr. slapovni model ali RUP. Kritiki so težke metodologije videli kot neživljenjske, počasne, birokratske in v nasprotju z dejanskim načinom dela programerjev. Predlagali so drugačen način dela, se organizirali v Agilno zavezništvo (angl. *Agile Alliance*) in sklepe objavili v manifestu (angl. *Agile Manifesto*), ki ostaja skupni imenovalec vseh agilnih metodologij.

1.6 Agilne metodologije razvoja informacijskih sistemov

1.6.1 Agilni razvoj

Enotna agilna metodologija razvoja informacijskih sistemov ne obstaja. Razvijalci programske opreme, ki so pri svojem delu skrenili stran od težkih metodologij in razvili vsak svoj lažji pristop, so se organizirali v agilno zavezništvo in definirali skupne značilnosti agilnih (gibkih) metodologij. Začetki tega zavezništva segajo v leto 2001, ko se je v kraju Snowbird v zvezni državi Utah v Združenih državah Amerike srečalo 17 razvijalcev. Namen srečanja je bil opredeliti skupne principe različnih lahkih metodologij,

ki so jih uporabljali pri svojih projektih razvoja programske opreme. Srečanje je prineslo štiri pomembne rezultate: besedo agilen, agilni manifest, dvanajst načel agilnega programiranja in agilno zavezništvo (Cimolini & Cannell, 2012, str. 1).

Od takrat beseda **agilen** na področju razvoja informacijskih sistemov pomeni premik od težkih k lahkemu načinu vodenja procesov projekta razvoja, kar vključuje tudi iterativni pristop k odkrivanju, kaj naj bi končni programski proizvod delal in kakšen naj bi bil videti. **Agilni razvoj** pomeni predvsem filozofijo in zbirko vodil pri razvoju informacijskih sistemov v neznanem, hitro spreminjajočem se okolju. Lahko se uporablja z vsako metodologijo razvoja informacijskih sistemov. Lahko pomeni dopolnitev adaptivnih pristopov k metodi življenjskega cikla in metodologij, ki jo podpirajo. Poudarek pa je na prevzemu adaptivnega pristopa in njegovi predelavi v agilnega, torej prilagodljivega, v vseh razvojnih aktivnostih. **Agilno modeliranje** je filozofija o izdelavi modelov. Ne gre za definicijo novih modelov, ampak za smernice za uporabo modelov, ki so sicer obstajali že pred uveljavitvijo agilnega pristopa. Nekateri modeli so zelo formalni in izdelani do podrobnosti, spet drugi so bolj skice in zelo minimalistični (Satzinger et al., 2012, str. 226).

Agilni manifest je objavljen na spletnih straneh agilnega zavezništva v številnih jezikih in ga tu povzemamo v slovenščini (Agile manifesto, 2016):

Člani zveze so kot svoj cilj zapisali iskanje boljših pristopov razvoja programske opreme, tako da pri tem sami sodelujejo in pomagajo drugim. Sledijo štirim načelom:

1. Posamezniki in njihova komunikacija so pomembnejši kot sam proces in orodja.
2. Delujoča programska oprema je pomembnejša kot popolna dokumentacija.
3. Vključevanje (sodelovanje) uporabnika je pomembnejše kot pogajanje na podlagi pogodb.
4. Upoštevanje sprememb je pomembnejše od sledenja načrtu.

V izjavi navajajo, da so proces, orodja, dokumentacija, pogodbe in plani vsekakor pomembni elementi metodologije, vendar pa večji pomen dajejo posameznikom, sodelovanju, delujoči programski opremi,vključevanju uporabnika in reagiranju na spremembe.

Agilni manifest je kratek in jedrnat. Kot tak je sam po sebi zgled osnovnih načel agilnosti. Je lahek, učinkovit in zadosten (Cimolini & Cannell, 2012, str. 2).

Prvi stavek manifesta poudarja nesebično predanost skupine panogi razvoja programske opreme. Štiri vrednote, ki jih naštejejo v nadaljevanju, opredelijo štiri temeljne aktivnosti razvoja programske opreme. Zaključni stavek je pomemben, ker eksplicitno pove, da čeprav je pristop k razvoju lahek, brez veliko formalnosti in dokumentiranja, pa razvoj programske opreme nikoli ne more potekati povsem brez upravljanja. Brez stvari, naštetih

na desni, pade razvoj programske opreme v brezno nedisciplinirane ga »hekanja« in kavbojskega pisanja kode (Cimolini & Cannell, 2012, str. 2).

Dvanajst načel agilnega programiranja (Agile Alliance, 2016):

1. Naša najvišja prioriteta je zadovoljstvo naročnika, kar dosežemo z zgodnjim in nenehnim izdajanjem vredne programske opreme.
2. Pozdravljamo zahteve po spremembah, celo pozno v procesu razvoja. Agilni procesi sprejemajo spremembe v prid naročnikove primerjalne prednosti.
3. Pogosto izdajamo delujočo programsko opremo, v času od nekaj tednov do nekaj mesecev. Stremimo k čim krajšim časom razvoja.
4. Osebje naročnika in razvijalci morajo sodelovati ves čas trajanja projekta.
5. Projekte gradimo okrog motiviranih posameznikov. Damo jim okolje in podporo, ki ju potrebujejo, in jim zaupamo, da bodo delo opravili.
6. Najbolj učinkovit način za izmenjavo informacij med člani razvojne ekipe je pogovor iz oči v oči.
7. Delujoča programska oprema je glavno merilo napredka.
8. Agilni procesi stremijo k trajnostnemu razvoju. Sponzorji, razvijalci in uporabniki naj bi bili zmožni vzdrževati stalen tempo dela za nedoločen čas.
9. Nenehno stremljenje k tehnični popolnosti in dobremu načrtovanju poveča gibkost (agilnost).
10. Enostavnost – to je umetnost zmanjševanja količine nepotrebnega dela – je bistvena.
11. Najboljša arhitektura, zahteve in načrtovanje zrastejo v samoorganiziranih ekipah.
12. V rednih intervalih ekipa preverja, kako bi lahko postala bolj učinkovita in v skladu s temi spoznanji sproti prilagaja svoj način dela.

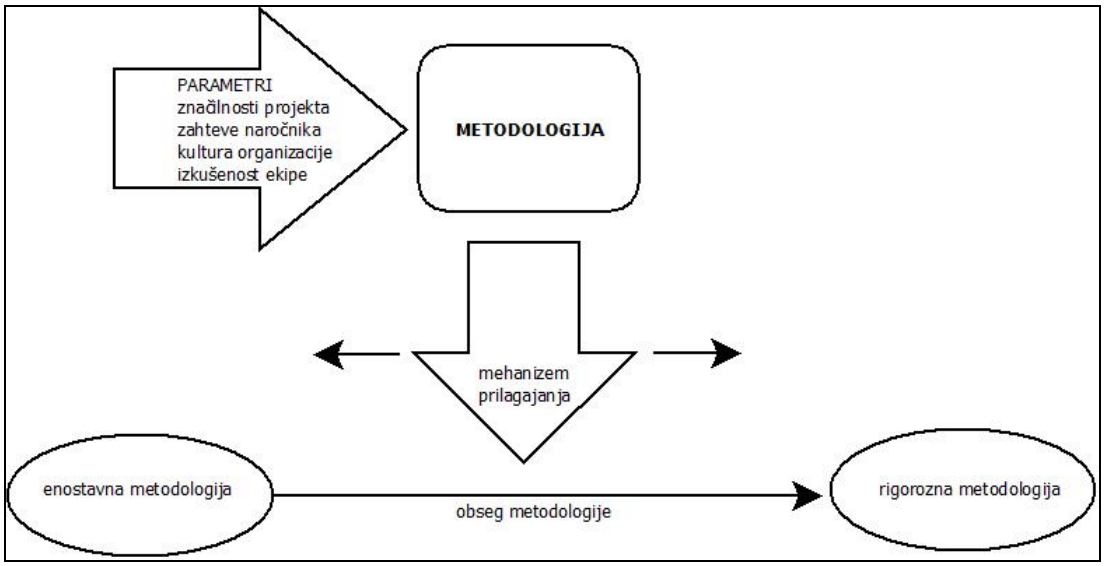
Pri tradicionalnih pristopih k razvoju informacijskega sistema sledimo razvojnim fazam po vrsti: načrtovanje sistema se začne takrat, ko je končana analiza. Programirati začnemo takrat, ko je končano načrtovanje. In ko je vse sprogramirano, povabimo naročnika, da si ogleda izdelek in upamo, da mu bo všeč. Pri takšnem načinu dela se vse uporabniške zahteve obravnavajo hkrati. Vse hkrati analiziramo, potem za vse zahteve hkrati načrtujemo informacijsko rešitev. Ko programiramo, programiramo kodo za celoten sistem. Na tak način izdelamo celotno rešitev naenkrat. S takšnim načinom dela lahko prihranimo veliko sredstev in truda. Vendar pa to drži le v primerih, ko se projekti oziroma uporabniške potrebe ne spreminjajo (Agile process, 2016).

Agilno zavezništvo samo sebe opisuje kot neprofitno organizacijo z globalnim članstvom, ki je zavezana k napredku tako agilnih načel razvoja kot tudi agilnih praks. Verjamejo, da agilni pristopi hitreje prinesejo večjo vrednost in tako pripomorejo, da je panoga razvoja programske opreme bolj produktivna, humana in vzdržna (Agile Alliance, 2016).

»Bistvo vseh agilnih metodologij je izdelava projekta v manjših iteracijah, katerih končni izdelek je 'oprijemljiv' in uporaben skoraj brez sprememb v naslednjih iteracijah.« (Korelič, 2011, str. 26)

Bajec in Krisper (2003, str. 72) poudarjata, da agilne metodologije niso nujno lahke metodologije. Prav tako velja, da vsaka lahka metodologija ni nujno agilna. Agilnost predvsem pomeni gibkost, torej veliko zmožnost sprotnega prilagajanja obsega metodologije dejanskim potrebam projekta. Avtorja trdita, da je prilagodljivost popolnoma neodvisna od teže metodologije, kot smo jo opredelili v prejšnjem razdelku. Agilna metodologija se s pomočjo mehanizma za prilagajanje v odvisnosti od parametrov, kot so značilnosti projekta, zahteve naročnika, kultura organizacije, izkušenost ekipe, pozicionira na ustrezno mesto na skali, ki opredeljuje obseg in s tem tudi težo metodologije (Bajec & Krisper, 2003, str, 72). Shematsko je takšno prilagajanje prikazano na Sliki 8.

Slika 8: Obseg metodologije



Vir: M. Bajec & M. Krisper, Agilne metodologije razvoja informacijskih sistemov, 2003, str. 72, Slika 2.

1.6.2 Pregled agilnih pristopov k razvoju informacijskih sistemov

Udeleženci srečanja v Snowbirdu so se uskladili glede vsebine agilnega manifesta in dvanajstih principov agilnega razvoja programske opreme. Strinjali pa so se, da se ne strinjajo o tem, katere metodologije so primerne za udejanjanje manifesta in dvanajstih načel v praksi, pri konkretnem razvoju informacijskih rešitev. To nestrinjanje kaže predvsem na spoznanje, da neka univerzalna metodologija, ki bi bila primerna za vse velikosti projektov in za vse situacije, ne obstaja (Cimolini & Cannell, 2012, str. 3).

Metodologije, ki so se razvile po agilnih načelih, se do določene mere med sabo prekrivajo, kar je razumljivo. Razlikujejo se v terminologiji in na poudarkih, ki jih dajejo posameznim agilnim načelom, orodjem in tehnikam. Najpogosteje omenjane agilne metodologije so (Cimolini & Cannell, 2012, str. 4):

1. prilagodljiv razvoj programske opreme (angl. *adaptive software development*, v nadaljevanju ASD),
2. ekstremno programiranje (angl. *extreme programming*, v nadaljevanju XP),
3. gruč ali skram (angl. *scrum*),
4. crystal clear in ostale crystal metode,
5. razvoj narekovan z lastnostmi (angl. *feature-driven development*, v nadaljevanju FDD)
6. metoda dinamičnega razvoja sistemov (angl. *dynamic system development method*, v nadaljevanju DSDM) in
7. agilno združen proces (angl. *agile unified process*, v nadaljevanju AUP).

V nadaljevanju povzemamo bistvene značilnosti nekaterih ključnih lahkih metodologij.

Prilagodljiv razvoj programske opreme (ASD) sprejema kot dejstvo, da deležniki ne vedo vsega o problemu in so zato naredili nekaj napačnih predpostavk. Pri tem pristopu delo izvajamo v iteracijah. Vsaka iteracija ima po tri cikle (Cimolini & Cannell, 2012, str. 4):

1. predvidevanje (angl. *speculation*),
2. sodelovanje (angl. *collaboration*) in
3. učenje (angl. *learning*).

Predvidevanje se uporablja namesto načrtovanja sistema. Sodelovanje poudarja odprto in pošteno komunikacijo med vsemi deležniki. Učenje tu pomeni kratke časovno omejene aktivnosti razvoja, ki vključujejo načrtovanje, programiranje in testiranje (Cimolini & Cannell, 2012, str. 4).

Ekstremno programiranje (XP) nekateri avtorji navajajo kot najbolj razširjeno agilno metodologijo. Nekatere dobre načrtovalske in programerske prakse je ta metodologija pripeljala do ekstremov, od tod tudi ime. Ta pristop ne dopušča izvajanja aktivnosti, ki ne dajejo takoj uporabnih rezultatov. Tako na primer v tabele ne dodajamo trenutno nepotrebnih stolpcev za morebitno prihodnjo rabo. Stolpec bomo dodali šele takrat, ko ga bomo zares potrebovali (Cimolini & Cannell, 2012, str. 4).

Kent Beck, utemeljitelj metode XP in soavtorica druge izdaje njegove knjige ugotavljata, da je osnovni problem pri razvoju programske opreme tveganje. Prepoznavata več vrst tveganj (Beck & Andres, 2004, str. 3):

1. Zamujanje dogovorjenih rokov.
2. Neuspešni zaključki projektov – projekti se opustijo, brez da bi dali uporaben rezultat.
3. Izdelani sistem po krajšem času uporabe zaradi šele med uporabo ugotovljenih napak ali zaradi potrebnih velikih sprememb postane neuporaben in ga je potrebno zamenjati.
4. Programska oprema ima toliko napak, da se je po uvedbi v uporabo ne da uporabljati.
5. Napačno razumevanje poslovnih zahtev ima za rezultat, da izdelana programska oprema ne rešuje problemov, zaradi katerih jo je naročnik naročil.
6. V času od naročila do časa uvedbe v uporabo se je poslovanje tako spremenilo, da izdelana programska oprema ne pomeni več rešitve problema.
7. Izdelani program ima veliko zanimivih rešitev, ki jih je bilo zabavno programirati, a nobena od njih naročniku ne prinaša velike koristi.
8. Menjava članov razvojne ekipe sredi dela.

Beck in soavtorica predlagata uporabo metode XP kot sredstvo za zmanjševanje teh tveganj (Beck & Andres, 2004, str. 5):

1. Kratki razvojni cikli naj bi zmanjševali tveganje nedoseganja zastavljenih ciljev in rokov, saj je vsa odstopanja možno ugotoviti kmalu. XP predlaga razvojne cikle dolge največ nekaj mesecev. Znotraj razvojnega cikla predlaga delitev dela na eno- do štiritedenske iteracije za delo na uporabniških zahtevah, znotraj teh iteracij pa naloge, ki trajajo od enega do treh dni. Tako se lahko nastali problemi zaznajo in rešujejo v trajanj teh iteracij. XP tudi poudarja načelo, da se najprej implementirajo zahteve z najvišjo prioriteto. Na ta način naj bi dosegli, da so zahteve, ki bi se ne implementirale v roku, zahteve z manjšo prioroteto.
2. Tveganje neuspešnega zaključka projekta poskuša zmanjšati s tem, ko pozove naročnika, da opredeli najmanjšo možno funkcionalnost, ki za poslovanje naročnika pomeni največ. To bo funkcionalnost, ki se bo razvijala prva in se prva uvedla v uporabo. S takšnim pristopom je manj stvari, ki lahko gredo narobe pred prvo uvedbo v uporabo in vrednost dobljene rešitve je za naročnika največja.
3. Z veliko sprotnega testiranja se poskuša čim bolj zmanjšati število napak. Napake se želi odkrivati sproti in s tem preprečiti, da bi se v programski opremi kopičile. Testira se po vsaki spremembi programa.
4. Po XP metodi je potrebno testirati tako s stališča pravilnosti delovanja programske opreme (ang. *functions*) kot tudi s stališča za uporabnika pomembnih lastnosti programa (angl. *features*). S tem se poskuša preprečiti neuporabnost programske opreme po uvedbi v uporabo.
5. Napačno razumevanje poslovnih zahtev se poskuša preseči s stalno prisotnostjo predstavnika naročnika v razvojni ekipi. Tekom razvoja se dopolnjuje specifikacija zahtev, kar se odraža tudi v razviti programski opremi.
6. S kratkimi razvojnimi cikli je verjetno, da bodo spremembe poslovnih zahtev manjše, kot bi bile pri daljših razvojnih ciklih. Ob koncu vsakega razvojnega cikla lahko

naročnik predlaga nove funkcionalnosti in odpove izvedbo še ne realiziranih funkcionalnosti.
7. Obilju lepo sprogramiranih nepomembnih funkcionalnosti se razvojna ekipa izogne s tem, ko vedno razvija najprej tiste funkcionalnosti, ki so naročniku najpomembnejše.
8. Odhajanje članov razvojne ekipe sredi projekta se skuša preprečiti na več načinov: Od članov se pričakuje, da bodo prevzeli odgovornost za ocenjevanje obsega svojega dela in za njegovo zaključevanje. Pravila, kdo lahko oceni obseg dela in spreminja to oceno, so jasna. Na ta način je manj možnosti, da bi člani postali frustrirani zaradi zahtev, da v kratkem času naredijo nemogoče. Spodbuja se medosebne stike med člani ekipe, na ta način se skuša preprečiti osamljenost, ki je pogosto temeljni vzrok za nezadovoljstvo pri delu. Poudarek je tudi na delu z novimi člani ekipe, ki se jim postopno dodeljuje več in več odgovornosti. Pri delu si medsebojno pomagajo in na voljo jim je pomoč po stažu starejših članov ekipe.

Avtorja metodo XP temeljita na naslednjih **vrednotah** (Beck & Andres, 2004, str. 15):

1. komunikacija,
2. preprostost,
3. povratna informacija,
4. pogum in
5. spoštovanje.

Iz teh za vsakdanje delo precej abstraktnih vrednot izpeljeta konkretnejša **načela** (Beck & Andres, 2004, str. 24):

1. humanost: programsko opremo razvijajo ljudje in treba je upoštevati njihove potrebe,
2. ekonomičnost: kar delamo, mora imeti vrednost za naročnika,
3. koristnost za vse: najpomembnejše načelo ekstremnega programiranja, ki ga je najtežje upoštevati,
4. samo podobnost: strukturo obstoječe rešitve poskušamo uporabiti v novem kontekstu,
5. izboljševanje: prizadevamo si doseči odličnost skozi nenehne izboljšave svojega dela,
6. raznolikost: člani ekipe naj bodo različnih sposobnosti in pogledov, da bomo lahko problematiko osvetlili z različnih vidikov,
7. refleksija ali pogled nazaj: svoje preteklo delo analiziramo in se na napakah učimo,
8. stalen tok izdelkov,
9. priložnosti: na probleme gledamo kot na priložnosti za spremembe,
10. redundanca: za težke probleme si prizadevamo najti več poti do rešitve,
11. neuspeh: če imamo težave, je dobro preizkusiti več poti do rešitev, tudi neuspešne poti nas vodijo do novih izkušenj in spoznanj,
12. kakovost: prizadevanje za kakovost ne podaljšuje časa za izvedbo,
13. majhni koraki in

14. prevzemanje odgovornosti: odgovornost ne more biti podeljena, torej vsiljena, lahko je samo sprejeta.

Na osnovi načel opredelita metodo XP kot izvajanje naslednjih **praks ali tehnik dela** (Beck & Andres, 2004, str. 35):

1. sedenje skupaj: delo poteka v prostoru, ki je dovolj velik za celotno razvojno ekipo,
2. celovita ekipa: v razvojno ekipo vključimo vse posameznike, ki obvladajo specifična znanja, ki jih potrebujemo za napredek,
3. informativno delovno okolje: že vstop v prostor naj da vpogled v stanje projekta, pomagamo si lahko s tablami in s karticami, ki govorijo o nalogah in napredku,
4. delajmo polni energije: ne opravljamo nadurnega dela, skrbimo za odmore, le spočita glava ima dobre zamisli, zato nikar ne prihajajmo na delo bolni,
5. programiranje v paru: vso programsko opremo naj izdelujeta po dva programerja za enim računalnikom,
6. zgodbe: povedo, kaj je treba izdelati, opisujejo funkcionalnost, ki jo bo uporabnik opazil,
7. tedenski cikli: delo planiramo za teden dni vnaprej, cilj za konec tedna je delujoči program,
8. četrtletni cikli: delo planiramo za četrt leta vnaprej,
9. ohlapnost (avtorja uporabljata angleški izraz *slack*): pri načrtovanju si puščamo nekaj časovne rezerve, v načrt vključujemo manjše naloge, ki jih lahko opustimo,
10. desetminutne verzije: celotno avtomatsko prevajanje programa in testiranje naj ne traja več kot 10 minut,
11. stalna integracija,
12. stalna priprava testov in
13. postopno načrtovanje.

Programiranje v paru je dialog med dvema sodelavcema, ki si podajata tipkovnico in drug drugemu gledata na ekran in pod prste. Ves čas poskušata izboljšati svoje programiranje. Prednosti programiranja v paru so predvsem:

1. partnerja držita drug drugega osredotočenega na nalogo,
2. s sprotnim pogovorom o delu izboljšujeta svoj izdelek,
3. razčiščujeta ideje,
4. ko eden obtiči, lahko drugi prevzame iniciativo in je tako delo manj stresno in
5. drug drugega opominjata na vodila pri delu, to so prakse ekstremnega programiranja, ki se jim je zavezala njuna ekipa.

Gruč je na novo skovana slovenska beseda za agilno metodologijo z angleškim imenom *scrum*. Besedo uvaja David Pahor, ob izidu prevoda knjige »V gruču do uspeha – Umetnost vodenja projektov z metodo SCRUM« avtorja Jeffa Sutherlanda. Nadomestila

naj bi iz angleščine izpeljano in nič slovensko zvenečo besedo skram, ki se jo sicer dostikrat uporablja namesto angleške *scrum*. Jeff Sutherland je utemeljitelj metode, ki se je s področja razvoja informacijske tehnologije razširila tudi na druga delovna področja, ki terjajo projektni pristop. Metodologija je poimenovana po delu igre ragbija, kjer ekipa skupaj poskuša spraviti žogo do konca igrišča. Igralci so pozorni predvsem na skrbno poravnavo, enotnost namena in jasnost cilja. Isto želimo doseči tudi pri delu v skupini. (Sutherland, 2016, str. 9).

Z besedo gruč v svetu vodenja projektov poimenujemo vsakodnevni kratki sestanek, na katerem se člani ekipe pogovorijo o delu, načrtovanem za ta dan. Vsak dan je ta sestanek v istem prostoru in ob istem času. Začne se točno in se konča natanko čez petnajst minut. Sestanku lahko prisostvuje kdorkoli, pravico do besede pa imajo le člani ekipe. Člani na kratko predstavijo, kaj so naredili prejšnji dan in kaj nameravajo narediti ta dan. Če pričakujejo problem, izobesijo zastavico. Problemi se rešujejo po koncu sestanka. Delo poteka v omejenih časovnih okvirih, ki trajajo od dva do štiri tedne. Imenujejo jih šprinti. Roki za zaključke posameznih šprintov se zelo resno upoštevajo. Natančno ob roku izdajajo nove verzije delujoče programske opreme za testiranje ali za izvedbo prevzemnih testov. Kadar razvoj posameznih funkcij zaostaja, se jih izloči iz šprinta in se jih uvrsti na čakalni seznam. S tega seznama se jih lahko vključi v enega od kasnejših šprintov (Cimolini & Cannell, 2012, str. 5).

Šprinte se vodi s pomočjo treh sestankov (Cimolini & Cannell, 2012, str. 5):

1. načrtovanje,
2. tehnični pregled in
3. pregled procesa.

Načrtovanje je sestanek ekipe z lastnikom produkta, torej z naročnikom. Rezultat sestanka je seznam funkcionalnosti, ki bodo vključene v šprint. Tehnični pregled in pregled procesa sta izvedena ob koncu šprinta. Prvi je namenjen pregledu realizacije in odloženih funkcionalnosti. Na drugem člani ekipe pregledajo svoj način dela in ugotavljajo, kdaj so delali dobro, kaj ni delovalo in kako bi lahko v naslednjem šprintu delovali bolje. Na vseh sestankih po tej metodi se zahteva velika discipliniranost. Določen je strog časovni okvir in treba se je držati dnevnega reda.

Vse delo je organizirano okrog dveh seznamov: čakalni seznam produkta in čakalni seznam šprinta. **Čakalni seznam produkta** je seznam funkcionalnosti na visokem nivoju, ki jih je treba realizirati. **Čakalni seznam šprinta** je seznam nalog, ki jih je potrebno opraviti, da bi realizirali fukcionalnosti, naštete na čakalnem seznamu produkta. Na čakalnem seznamu šprinta razvijalci izbirajo, kaj bodo realizirali in označujejo, kaj so realizirali. Izbira in označevanje potekata na vsakodnevnem gruču. Tak način dela naj bi med člani ekipe spodbujal timskega duha in tovarištvo. Projekt ima plitvo vodstveno

strukturo. Vodi ga mojster gruča (angl. *scrum master*), ekipo pa sestavlja največ osem članov. Deležnike naročnika predstavlja lastnik produkta (Cimolini & Cannell, 2012, str. 5).

Crystal je družina sorodnih metodologij. Metodologije so označene z barvami: prozorna, rumena, oranžna in rdeča. Večji kot je projekt, bolj temne barve je metodologija. To pomeni, da več kot je v sklopu projekta treba narediti in več kot ima ekipa članov, več artefaktov je z metodologijo predpisanih. Velja tudi, da večji kot so stroški neuspeha projekta, več mehanizmov za zagotavljanje kakovosti in več formalnosti je treba izvajati v okviru projekta. Družina metodologij je tolerantna do izbiranja med različnimi agilnimi orodji in procesi. Dve pravili pa sta skupni vsem barvam: projekti morajo potekati po principih inkrementalnega razvoja, s prirastki (inkrementi) ne daljšimi od štirih mesecev. Pred in po vsakem prirastku mora imeti ekipa refleksivno delavnico (Cimolini & Cannell, 2012, str. 5).

Razvoj narekovan z lastnostmi (FDD) je pristop, osredotočen na modele. Preden se začne pisanje kode, je treba izdelati model rešitve na visokem nivoju. Model sestavlja seznam lastnosti (oziroma funkcionalnosti), ki so vidne naročniku in so zanj pomembne. Te lastnosti potem razgradimo na manjše kose, ki jih lahko razvijemo in uvedemo v kratkih razvojnih ciklih. Tipičen razvojni cikel pri tem pristopu traja dva tedna. Metodo sestavlja pet ključnih aktivnosti (Cimolini & Cannell, 2012, str. 6):

1. izdelava preglednega modela,
2. izdelava seznama lastnosti,
3. načrtovanje dela po lastnostih,
4. načrtovanje sistema po lastnostih in
5. izgradnja sistema po lastnostih.

Pri tej metodi napredek spremljamo glede na doseganje mejnikov in ocenjujemo odstotek realizacije za posamezne lastnosti kot tudi odstotek realizacije za celoten projekt (Cimolini & Cannell, 2012, str. 6).

Metoda dinamičnega razvoja sistemov (DSDM) je bolj zrela in disciplinirana verzija hitrega razvoja aplikacij (RAD). Razvoj metode vodi DSDM konzorcij. Metoda kombinira formalne strukture projektnega vodenja s principi iterativnega razvoja. Projekti so razdeljeni v tri faze. Druga faza se še naprej deli v pet stopenj (Cimolini & Cannell, 2012, str. 6):

1. faza: predprojekt,
2. faza: življenjski cikel projekta:
    1. študija izvedljivosti,
    2. študija poslovanja,

3. iteracija modela funkcij,
4. iteracija načrtovanja in gradnje sistema in
5. implementacija,

3. faza: poprojektne aktivnosti.

Tri glavne faze so prispevek klasičnih metod vodenja projektov k tej metodi. Zadnje tri stopnje druge faze pa predstavljajo faze iteracij, kar je agilni del te metodologije (Cimolini & Cannell, 2012, str. 6).

Agilno združen proces je ena od peresno lahkih izvedenk IBMjeve metodologije RUP. To je metodologija, ki se nahaja nekje med zelo formalno IBMjevo RUP metodologijo na eni strani in med neformalno XP metodologijo na drugi strani (Cimolini & Cannell, 2012, str. 7).

Včasih srečamo izraz **pragmatično programiranje,** ki pa ne označuje posebne metodologije. Nanaša se bolj na miselnost izkušenih agilnih programerjev, ki združujejo posamezne agilne tehnike, da zgradijo svoj lasten agilni pristop za posamezen projekt (Cimolini & Cannell, 2012, str. 7).

1.6.3 Prehod na uporabo agilnega pristopa pri razvoju programske opreme

Kadar se odločamo za prehod na agilen način dela pri razvoju programske opreme, imamo na voljo predvsem dve možnosti (Cimolini & Cannell, 2012, str. 10):

1. uvedba ene od definiranih agilnih metodologij ali
2. izgradnja lastne metodologije z upoštevanjem nekaterih agilnih praks.

Pri izbiranju prave metodologije za organizacijo projekta je treba odgovoriti na tri glavna vprašanja (Cimolini & Cannell, 2012, str. 8):

1. Koliko nas bo stal neuspeh? Dražji kot je neuspeh projekta, bolj formalna in težka mora biti metodologija, ki nas bo vodila do cilja.
2. Kako velik je naš projekt? Večji kot je projekt, težje je učinkovito neformalno komunicirati. Vedno pa lahko velik projekt poskušamo razdeliti na manjše podprojekte in vsakega od njih vodimo po agilnih načelih.
3. Kakšna je naša trenutna kultura, naš trenutni pristop k delu? Kako neformalno smo pripravljeni delati?

Pri oblikovanju svojega lastnega agilnega pristopa izbiramo iz nabora praks in tehnik, ki so del ene ali večih agilnih metodologij. Včasih se pojavljajo pod različnimi imeni, a pomenijo isto (Cimolini & Cannell, 2012, str. 10):

1. tesni časovni okviri,
2. pričakovanje zahtev za spremembe funkcionalnosti,
3. pogosta komunikacija s strankami in kolegi razvijalci,
4. dnevni kratki sestanki,
5. programiranje v paru,
6. delo v skupnem prostoru,
7. ploska upravljalska struktura projekta,
8. enostaven dostop do strokovnjakov,
9. osredotočen čas,
10. refleksivno učenje,
11. razvoj z neprestanim testiranjem,
12. programiramo le tisto, kar je res potrebno in
13. enostavnost in jasnost programske kode.

Vedno se pojavlja vprašanje, koliko aktivnosti vodenja projekta je potrebno izvajati. Izkušnje kažejo, da čisto brez načrtovanja in spremljanja izvajanja projekta nikoli ne gre. Tudi zelo majhni projekti, na katerih dela en sam programer, se lahko hitro izrodijo v zelo drag nered, če ni nad potekom dela nobenega nadzora. Zato si tudi zelo izkušeni programerji z dobrim spominom pomagajo vsaj s seznamom nalog, ki si ga napišejo na papir. Pisanje seznama nas že samo po sebi prisili v premislek o nalogah in problemu. Prav tako je seznam odlično vodilo pri delu po morebitni prekinitvi (Cimolini & Cannell, 2012, str. 11).

V zadnjem času je na voljo veliko bolj ali manj kompleksnih orodij za računalniško podprto organizacijo dela in spremljanje realizacije nalog. Orodja za projektno vodenje so lahko samostojni računalniški programi ali pa so deli orodij za računalniško podprto načrtovanje in gradnjo programske opreme. Na voljo so tudi čedalje bolj domiselni osebni organizatorji v obliki aplikacij za mobilne naprave, ki si jih lahko namestimo na pametni telefon in jih imamo tako vedno pri sebi. Izkušnje in pogled na delovne mize sodelavcev pa kažejo, da se za osebno organizacijo dnevnih ali tedenskih nalog tudi na področju razvoja programske opreme še vedno veliko uporabljata papir in svinčnik. Ugotavljamo, da črtanje nalog s seznama na papirju zelo prispeva k zadovoljstvu človeka ob opravljenem delu in tako poveča zanos in pripravljenost za nadaljevanje.

1.7 Stroški razvoja informacijske rešitve

1.7.1 Dilema med lastnim razvojem in vključitvijo zunanjih izvajalcev

»Vedno bolj se uveljavlja prepričanje, da je učinkovit menedžment v javnem sektorju in državni upravi javna dobrina, katerega temeljna naloga je smotrna poraba javnih virov.« (Žurga, 2015, str. 23) Tudi v delovanje organizacij državne uprave je treba uvesti mehanizme zagotavljanja učinkovitosti in nenehnega izboljševanja. Stranke ponavadi

nimajo izbire, kje bodo posamezne storitve opravile, a kljub temu so njihove zahteve in pričakovanja vedno večji, organizacije državne uprave pa se nanje poskušajo odzvati. Od organov državne uprave se čedalje bolj pričakuje večja učinkovitost in nižanje stroškov (Žurga, 2015, str. 7).

Tudi pri razvoju informacijskih rešitev v naši organizaciji se srečujemo z omejenimi sredstvi na eni strani in čedalje večjimi potrebami oziroma zahtevami na drugi strani. Zato je pomembno, da si prizadevamo za najučinkovitejše poti razvoja rešitev. Na voljo imamo nekaj lastnih kadrovskih virov, ne pa dovolj. Tako nekatere rešitve razvijamo sami, nekatere pa smo pridobili z nakupom na trgu.

Še pred nekaj leti so imele organizacije za pridobitev nove programske opreme na voljo tri glavne možnosti (Shelly et al., 2008, str. 234):

1. lasten razvoj v organizaciji z lastnimi viri,
2. nakup komercialnega proizvoda na trgu in morebitne prilagoditve in
3. najem svetovalcev ali zunanjih virov za izvedbo dela po lastnih zahtevah.

Danes se z uveljavitvijo interneta in elektronskega poslovanja med organizacijami tem opcijam pridružujejo še (Shelly, 2008, str. 234):

4. ponudniki aplikativnih storitev,
5. programska oprema z gostovanjem na spletu in
6. razni ponudniki s široko ponudbo celovitih rešitev.

Izkušnje v naši organizaciji kažejo, da je lahko učinkovita tudi opcija najema zunanjega strokovnjaka za določeno tehnično področje razvoja in njegova vključitev v domačo razvojno ekipo po potrebi, torej takrat, ko potrebujemo njegova specifična tehnična znanja. Prizadevamo si, da v razvojni ekipi sodeluje vsaj en tehnično vešč domač razvijalec, ki pozna tudi vsebino področja, ki se informatizira. Domač razvijalec spremlja delo zunanjega strokovnjaka na način, da razume zgradbo in logiko izdelka zunanjega strokovnjaka in da je kasneje sposoben njegove izdelke tudi vzdrževati. Na tak način prihaja tudi do prenosa znanja. Primer, kjer lahko učinkovito uporabimo zunanjega strokovnjaka s specifičnimi tehničnimi znanji, je izgradnja spletne storitve za izmenjavo podatkov v XML formatu.

Ocena stroškov razvoja programske opreme pred začetkom razvoja je bila od nekdaj nehvaležno opravilo. Zaradi mnogih neznank je ocena nenatančna in običajno prenizka. V primeru, ko imamo možnost izdelati rešitev z lastnimi viri, v delo pa po potrebi vključiti zunanjega strokovnjaka, je ocena nekaj lažja, saj je odgovornost za razvoj sistema na notranjih virih, zunanjega strokovnjaka pa plačujemo običajno po urah, ki jih je opravil na naši nalogi. Pri takšnem načinu sodelovanja se moramo pred začetkom dela z zunanjim

izvajalcem dogovoriti za način dela in za ceno ure, ni nam potrebno pred začetkom dela podrobno opredeliti celotne vsebine nalog zunanjega izvajalca. Seveda pa je odgovornost naročnika, da zna izbrati zunanjega strokovnjaka, ki resnično poseduje znanja, ki jih naročnik potrebuje, in ki je pri svojem delu tudi učinkovit.

1.7.2 Prispevčna ali kontributivna analiza

Pri presojanju ustreznosti načina izvedbe projekta in njegove ekonomičnosti si lahko pomagamo s prispevčno analizo. Prispevčna ali kontributivna analiza poslovnih odločitev temelji na prirastnih prihodkih in prirastnih stroških (Tajnikar, Brščič, Bukvič, & Ponikvar, 2004, str. 143).

Prirastni prihodek (angl. *incremental revenue*) je prihodek organizacije, ki nastane kot posledica določene poslovne odločitve. Prirastni prihodek ima eksplicitno sedanjo komponento, morebitno oportunitetno komponento in možno bodočo komponento. Eksplicitna sedanja komponenta pomeni prirastni prihodek, ki ga organizacija lahko neposredno izmeri po sprejetju odločitve. Oportunitetni prihodki pomenijo stroške, ki se jim organizacija s sprejetjem poslovne odločitve izogne, bi pa nastali, če te odločitve ne bi sprejela. Bodoča komponenta pomeni prihodke, ki jih organizacija pričakuje v prihodnosti in ne bi nastali, če organizacija te poslovne odločitve ne bi sprejela. Za potrebe izračuna vrednosti prirastnega prihodka je treba bodoče prihodke najprej diskontirati na sedanji čas s pomočjo oportunitetne diskontne stopnje in jih nato pomnožiti s stopnjo verjetnosti njihovega nastanka.

Prirastni ali inkrementalni stroški nastanejo kot posledica poslovne odločitve. Lahko so fiksni ali variabilni.

Prispevek poslovne odločitve je opredeljen kot prirastni prihodek te odločitve, zmanjšan za prirastne stroške, ki so z odločitvijo nastali. Smiselno je sprejeti le tiste poslovne odločitve, ki imajo pozitiven prispevek. Če pa izviramo med dvema izključujočim se odločitvama, pa je smiselno izbrati tisto, ki ima večji prispevek (Tajnikar et al., str. 143).

2 ELEKTRONSKO POSLOVANJE ZA IZMENJAVO PODATKOV MED VLADNIMI ORGANIZACIJAMI

2.1 Osnovni pojmi

»Vse, kar danes delamo v sklopu svoje poslovne dejavnosti s pomočjo računalniških aplikacij in računalniških omrežij, imenujemo elektronsko poslovanje.« (Jerman-Blažič, Klobučar, Perše, & Nedeljković, 2001, str. 11)

Elektronsko poslovanje pomeni poslovati v elektronski obliki z uporabo informacijske in komunikacijske tehnologije (Kovačič, Groznik, & Ribič, 2009, str. 55).

S pojmom **elektronsko poslovanje,** okrajšano **e-poslovanje** (angl. *e-commerce*) označujemo proces nakupa, prodaje ali izmenjave proizvodov, storitev ali informacij preko računalniških omrežij, vključno z internetom (Turban & King, 2003, str. 3).

V angleščini se za elektronsko poslovanje uporabljata dva izraza, *e-business* in *e-commerce*. Včasih se uporabljata kot sopomenki, velikokrat pa izraz *e-commerce* pomeni transakcije med poslovnimi partnerji, izraz *e-business* pa obsega več: ne samo prodaje in nakupov, ampak poleg tega tudi podporo uporabnikom, sodelovanje s poslovnimi partnerji, kot tudi izvedbo elektronskih transakcij znotraj organizacije (Turban & King, 2003, str. 3).

Elektronsko poslovanje se je razvijalo hkrati z razvojem tehnologije, ki ga omogoča. Razvoj elektronskega poslovanja se je začel z razvojem računalniških omrežij in interneta ter standardom za računalniško izmenjavo podatkov konec šestdesetih let dvajsetega stoletja. S pojavom elektronskih bančnih prenosov po varnih omrežjih v sedemdesetih letih dvajsetega stoletja se je spremenil način poslovanja na bančnem trgu. V poznih sedemdesetih in v osemdesetih letih se je e-poslovanje med podjetji razširilo v obliki sistemov za prenos datotek, računalniške izmenjave podatkov (angl. *electronic data interchange*, v nadaljevanju EDI) in elektronske pošte. Podjetja so s tem zmanjšala obseg papirnega poslovanja. Povečala se je avtomatizacija pisarniškega dela. Namesto izmenjave papirnih dokumentov se je pojavil računalniški prenos listin v standardizirani elektronski obliki. V devetdesetih letih dvajsetega stoletja so sistemi za izmenjavo sporočil postali sestavni del računalniških sistemov in omrežij. Z razvojem in z veliko razširjenostjo interneta in s pojavom svetovnega spleta pa je elektronsko poslovanje doživelo velikanski razmah, ki smo mu priča danes (Jerman-Blažič et al., 2001, str. 13).

Začetek elektronskega poslovanja pomeni z današnjega vidika enostavna izmenjava podatkov v elektronski obliki med dvema ali več posameznimi organizacijami. Prva organizacija je svoje podatke pripravila in zapisala v datoteko in to datoteko dostavila drugi organizaciji, ki je podatke iz datoteke prebrala v svoj informacijski sistem. Kljub navidezni enostavnosti opisanega postopka pa nam že kratek razmislek pove, da je za uspešno izvedbo potrebno veliko pripraviti in uskladiti:

1. Organizaciji se morata dogovoriti za obliko (format) datoteke, v kateri si bosta pošiljali podatke.
2. Dogovoriti se morata o komunikacijskem kanalu, po katerem si bosta datoteko poslali.
3. Dogovoriti se morata o trenutku, v katerem se bodo podatki pripravili iz informacijskega sistema organizacije pošiljateljice podatkov, saj se podatki običajno v času spreminjajo.

4. Dogovoriti se morata o frekvenci pošiljanja podatkov, če gre za periodično in ne za enkratno izmenjavo podatkov.
5. Dogovoriti se morata o načinu preverjanja pravilnosti prenešenih podatkov.
6. Dogovoriti se morata o postopkih v primeru, ko prenos podatkov ne uspe, ali pa ko prenos podatkov uspe in so pri tem ugotovljene napake.
7. Ena ali pa obe organizaciji morata svoj informacijski sistem prilagoditi sprejetim dogovorom.

Laudon in Traver navajata, da je dandanes fenomen elektronskega poslovanja tako širok, da se ga obravnava multidisciplinarno, predvsem z dveh vidikov: tehnološko in behavioristično (Laudon & Traver, 2012, str. 83).

Tehnološko je elektronsko poslovanje zanimivo kot zgleden primer uporabe internetne tehnologije. Znanstveniki s tega področja preučujejo razvoj računalniške strojne in programske opreme, telekomunikacijske sisteme, standarde, enkripcijo, torej kodiranje podatkov, načrtovanje podatkovnih baz in operacij za namen elektronskega poslovanja (Laudon & Traver, 2012, str. 83).

Behavioristično se preučuje vpliv elektronskega poslovanja na vrednostne verige podjetij in gospodarskih panog, na strategijo podjetij. Tesno v povezavi z razvojem elektronskega poslovanja se razvijata tudi podatkovno rudarjenje in umetna inteligenca. Ekonomisti preučujejo obnašanje potrošnikov na spletnih straneh, zakonitosti oblikovanja cen v spletnih prodajalnah in posebnosti elektronskih trgov. Predmet preučevanja so marketing, razvoj blagovnih znamk in zmožnost spletnih strani, da segmentirajo skupine kupcev in jih različno obravnavajo. Vse seveda s ciljem, da bi dosegli nadpovprečno donosnost naložbe (angl. *return on investment*, v nadaljevanju ROI) (Laudon & Traver, 2012, str. 83).

Glede na interakcijo subjektov v elektronskem poslovanju ločimo tri glavne vrste e-poslovanja (Jerman-Blažič et al., 2001, str.17):

1. podjetje – podjetje (B2B): povezave med prodajalci na drobno in dobavitelji, elektronsko bančništvo, sodelovanje na skupnih projektih,
2. podjetje – potrošnik (B2C): elektronsko poslovanje podjetij s končnimi uporabniki in
3. javna in državna uprava - javnost in ljudstvo (tako podjetja kot posamezniki): zajema tako poslovanje javne uprave s podjetji kot tudi poslovanje javne uprave s prebivalci.

Pri drugem viru najdemo delitev na šest vrst e-poslovanja (Kovačič, Groznik, & Ribič, 2009, str. 55):

1. med podjetji (B2B),
2. med podjetji in potrošniki (B2C),
3. med potrošniki (C2C),

4. med podjetji in javno oziroma državno upravo (B2G),
5. med državljani in javno oziroma državno upravo (C2G) in
6. znotraj javne oziroma državne uprave (G2G).

Tretji vir glede na naravo transakcij loči več vrst e-poslovanja (Turban, 2012, str. 7):

1. medsebojno poslovanje podjetij (angl. *Business to business*, v nadaljevanju B2B),
2. poslovanje podjetij s potrošniki oziroma končnimi uporabniki (angl. *Business to consumer*, v nadaljevanju B2C),
3. poslovanje podjetij z drugimi podjetji, ki poslujejo s končnimi potrošniki (angl. *Business to business to consumer*, v nadaljevanju B2B2C),
4. posamezniki prodajajo svoje storitve ali izdelke podjetjem (angl. *Consumer to business* v nadaljevanju C2B),
5. poslovanje med posamezniki, primer je prodaja rabljenih stvari (angl. *Consumer to consumer*, v nadaljevanju C2C),
6. Mobilno poslovanje: transakcije potekajo v brezžičnem okolju,
7. poslovanje med različnimi deli iste organizacije, običajno se odvija na intranetu (angl. *Intrabusiness (organizational)* EC),
8. poslovanje organizacije s svojimi zaposlenimi Business to employees (v nadaljevanju B2E),
9. sodelovanje posameznikov ali organizacij pri skupnem opravilu s pomočjo interneta, na različnih lokacijah (angl. *Collaborative commerce*),
10. elektronsko poslovanje v negospodarstvu (angl. *Nonbusiness EC*): naraščajoče število negospodarskih organizacij, kot so akademske ustanove, neprofitne organizacije, verske organizacije, socialne ustanove, vladne agencije, uporablja e-poslovanje, da bi zmanjšale stroške ali izboljšale svoje delovanje in ponudile boljše storitve svojim uporabnikom,
11. E-vlada (angl *E-government*): poslovanje države z državljani (G2C) in z drugimi in
12. povezovanje javnih elektronskih trgov (angl. *Exchange to exchange,* v nadaljevanju E2E).

2.2 Tehnološki gradniki

Dandanes je za veliko večino elektronskega poslovanja podlaga internetna tehnologija. V tem razdelku bomo v nadaljevanju po avtorjih Laudon in Traver (2012) povzeli opredelitve nekaterih pojmov s področja internetne tehnologije, povezanih z elektronskim poslovanjem.

S pojmom **internet** (v angl. skrajšano iz *internetwork*) ali **medmrežje** označujemo medsebojno povezano omrežje tisočev omrežij in milijonov računalnikov strežnikov, ki povezuje podjetja, izobraževalne ustanove, vladne agencije in posameznike. Internet oskrbuje približno dve milijardi ljudi po vsem svetu z elektronsko pošto, aplikacijami,

novičarskimi skupinami, nakupovanjem, takojšnjim sporočanjem, glasbo, video posnetki in novicami, povzemata avtorja podatke po spletni strani Internetworldstats iz leta 2011. Konec junija 2016 je bila ta številka že tri milijarde in pol (Internetworldstats, 2016).

Splet (angl. *web*) je ena najpopularnejših storitev interneta. Omogoča dostop do ogromnega števila spletnih strani. Spletne strani so dokumenti, izdelani z jezikom za označevanje nadbesedila (angl. *hyper text markup language*, v nadaljevanju HTML). Ti dokumenti vsebujejo besedilo, slike, zvočna in filmska gradiva in druge objekte, vsebujejo pa tudi nadpovezave, ki omogočajo uporabnikom hitro premikanje med spletnimi stranmi. Po spletnih straneh se prestavljamo z uporabo programov, imenovanih spletni brskalniki. Svetovni splet (angl. *world wide web*, v nadaljevanju WWW) je informacijski prostor, kjer so dokumenti in drugi viri informacij identificirani z enoličnimi krajevniki vira (angl. *uniform resource locators*, v nadaljevanju URL), medsebojno povezani z nadpovezavami in so dosegljivi na internetu.

Čeprav imamo morda občutek, da doživlja internet velik razmah ravno v zadnjih letih, pa internet ni od včeraj, njegovi začetki segajo 50 let nazaj. Njegov razvoj lahko v grobem razdelimo na tri faze (Laudon & Traver, 2012, str. 103):

1. faza inovacij v letih od 1961 do 1974: to je čas, ko so nastali osnovni gradniki interneta, najprej samo kot koncepti, nato pa še realizirani kot strojna in programska oprema,
2. druga faza v času od 1974 do 1995: internet so uporabljale le velike ustanove in
3. tretja faza, ki še traja: prišlo je do komercializacije, internet se je razširil v gospodarstvo in vse do gospodinjstev.

Med osnovne gradnike interneta prištevamo (Laudon & Traver, 2012, str. 108):

1. strojno opremo za izmenjavo paketov, ki zna razbiti binarna sporočila v majhne pakete, jih prenašati po različnih komunikacijskih poteh do naslovnika in jih tam spet sestaviti v sporočilo,
2. komunikacijski protokol TCP/IP, ki je postal osnovni komunikacijski protokol interneta in
3. računalniško tehnologijo odjemalec/strežnik.

Z besedo **protokol** označujemo množico pravil in standardov za prenos podatkov (Laudon & Traver, 2012, str. 110). TCP/IP je standardiziran protokolni sklad, ki poskrbi za komunikacijo v heterogenem okolju. To je kombinacija internetnega protokola (angl. *Internet Protocol*, v nadaljevanju IP) in protokola nadzora nad prenosom (angl. *Transmission Control Protocol*, v nadaljevanju TCP). Zaradi svoje popularnosti je TCP/IP protokol postal standard, ki je poznan kot medomreževanje, komunikacija v omrežju, sestavljenem iz več manjših omrežij (TCP/IP, 2016). TCP/IP je razdeljen na štiri sloje,

vsak od slojev skrbi za en vidik komunikacije: sloj omrežnega vmesnika, internetni sloj, transportni (prenosni) sloj in aplikacijski sloj. Stortive v aplikacijskem sloju so splet, elektronska pošta, FTP in veliko drugih.

Delovanje spleta sloni na treh standardih, to so URL, HTML in protokol za prenos nadbesedila (angl. *hyper-text transfer protocol*, v nadaljevanju HTTP), ki določa način, kako se sporazumevata spletni strežnik in brskalnik. Protokol za prenos datotek (angl. *file transfer protocol*, v nadaljevanju FTP), je programski standard za prenos datotek med računalniki z različnimi operacijskimi sistemi.

WebDAV (angl. *Web Distributing Authoring and Versioning)* je odprtokodni standard, v katerem so opisane razširitve HTTP protokola, ki omogočajo urejanje datotek preko oddaljenega strežnika. To je omrežni protokol za ustvarjanje aplikacij, ki omogočajo večim uporabnikom manipuliranje s podatki na oddaljenem strežniku.

2.3 Standard XML

V elektronskem poslovanju se je uveljavila izmenjava podatkov v XML formatu. XML je razširljivi označevalni jezik (angl. *extensible markup language*), ki omogoča univerzalno obliko strukturiranih dokumentov in podatkov na spletu. Dokumenti v XML formatu so na videz podobni dokumentom v HTML formatu, a je podobnost samo navidezna, saj je namen drugačen. HTML je označevalni jezik, ki je namenjen za formatiranje in prikazovanje besedila in slik v brskalniku. XML pa je označevalni jezik za strukturiranje podatkov (Introduction to XML, 2016).

XML uporabljamo, da z njim kreiramo dokument, ki vsebuje strukturirane podatke. Te potem lahko uporabljajo ali interpretirajo druge aplikacije. Format je lahko razumljiv tako računalnikom kot tudi ljudem.

Za razliko od značk v HTML jeziku so značke v XML razširljive, kar pomeni, da lahko definiramo svoje značke. XML je standardiziran, standard vzdržuje WWW konzorcij (angl. *WWW consortium*, v nadaljevanju W3C).

Tudi slovenska javna uprava v zadnjih letih pospešeno uvaja elektronsko poslovanje tako s podjetji, z državljani kot tudi med samimi inštitucijami javne uprave. V okviru širitve e-poslovanja se čedalje bolj uveljavlja tudi izmenjava podatkov v XML obliki.

3 RAZVOJ INFORMACIJSKIH REŠITEV V APEX RAZVOJNEM OKOLJU

3.1 Osnovne značilnosti orodja APEX za razvoj spletnih rešitev

Podjetje Oracle je razvilo orodje Application Express (v nadaljevanju APEX), ki se je sprva imenovalo HTML DB, kot dodatek k svojemu sistemu za upravljanje podatkovne baze. To je orodje za hiter razvoj aplikacij (RAD). APEX je primarno Oraclovo razvojno orodje za izdelavo profesionalnih spletnih aplikacij, ki do baze dostopajo z uporabo jezikov SQL in PL/SQL. Samo z uporabo spletnega brskalnika in nekaj programerskimi izkušnjami lahko razvijamo in uvajamo v uporabo hitre in varne aplikacije. Razvijamo lahko aplikacije za namizne računalnike kot tudi za mobilno okolje (Koratamaddi, 2015).

SQL je strukturirani poizvedovalni ali povpraševalni jezik za delo s podatkovnimi bazami (SQL, 2016). Določen je z ANSI/ISO SQL standardom. Programski stavki posnemajo ukaze v naravnem jeziku. Najbolj pogosta operacija v SQL-u je poizvedba, ki se izvede s SELECT stavkom. Stavek vrne podatke iz ene ali več tabel. Standardni SELECT stavek nima nobenega vpliva na podatke v bazi in jih ne spreminja. Velja pa omeniti, da kljub temu, da podatkov v bazi ne spreminja, lahko zelo zahtevna SQL poizvedba med svojim izvajanjem delovanje baze precej upočasni. Zato moramo biti pri sestavljanju poizvedb pazljivi in pri njihovem izvajanju dobro načrtovati čas izvajanja, da ne oviramo dela uporabnikov ali izvajanja paketnih obdelav podatkov.

PL/SQL je kratica za Proceduralni jezik /Strukturirani poizvedovalni jezik. To je jezik, ki ga je razvilo podjetje Oracle in pomeni proceduralno razširitev poizvedovalnega jezika SQL za Oracle relacijsko bazo. Vsebuje tudi proceduralne elemente jezika, kot so zanke in pogoji. Omogoča deklaracijo konstant in spremenljivk, procedur in funkcij, podatkovnih tipov in spremenljivk teh tipov in prožilcev. PL/SQL programske enote so lahko shranjene v bazi in so tako dostopne vsem aplikacijam, ki uporabljajo katerega od programskih vmesnikov za Oracle bazo.

Orodje APEX je samo po sebi brezplačno orodje, hkrati pa podjetje Oracle nudi razvijalcem, ki delajo s tem orodjem, vso podporo na profesionalnem nivoju. V vseh trenutno podprtih različicah verzij Oracle baze (oziroma Oracle SUPB) je orodje APEX v privzetem načinu že nameščeno.

Obstaja tudi brezplačna različica Oracle baze oziroma Oracle SUPB z imenom Oracle Database Express (v nadaljevanju XE). Glede na plačljivo različico ima sicer nekaj omejitev, tu jih povzemamo za verzijo 11g izdaja 2 (Oracle , 2016c):

1. količina shranjenih uporabniških podatkov je omejena na 11 GB,

2. uporablja največ 1 GB spomina,
3. uporablja le eno centralno procesno enoto na strežniku in
4. podpora proizvajalca je na voljo le preko Oracle spletnega foruma.

Podpora preko spletnega foruma pomeni, da bomo odgovore na svoje probleme in težave morali poiskati med temami, ki so se na forumu v preteklosti že obravnavale. Če odgovora tam ne bomo našli, bomo svoje vprašanje lahko zastavili tako, da bomo na forumu odprli novo temo za razpravo. Na forumu poleg tehničnega kadra organizacij uporabnikov opreme Oracle sodelujejo tudi uslužbenci podjetja Oracle. Tako da z veliko verjetnostjo lahko pričakujemo uporaben odgovor.

V kolikor uporabljamo plačljivo verzijo Oracle SUPB, pa imamo na voljo več možnosti za pomoč v težavah. Vprašanja lahko zastavljamo neposredno oddelku za podporo podjetja Oracle in lahko pričakujemo tudi hiter odgovor. Običajno probleme in vprašanja javljamo po elektronski poti in tako tudi prejemamo odgovore in sledimo napredku reševanja našega problema. Podjetje Oracle je globalno in zaposluje ljudi s celega sveta, zato reševanje problemov ni lokalno omejeno, ampak problem načeloma rešuje tisti sodelavec podjetja Oracle, ki se na obravnavano področje najbolj spozna. Tako lahko en dan dobimo odogovor od nekoga iz Romunije, spet drugič pa iz Indije.

Kljub navedenim omejitvam brezplačne različice baze pa je le-ta primerna rešitev za (Oracle, 2016c):

1. razvijalce, ki izdelujejo programe v jezikih PHP, Javi, .NET okolju, XML in odprtokodne programe,
2. administratorje baze, ki potrebujejo brezplačno začetno bazo za učenje,
3. neodvisne dobavitelje programske in strojne opreme, ki želijo ob svoji opremi distribuirati še začetno bazo brez dodatnih stroškov in
4. izobraževalne ustanove in študente, ki potrebujejo brezplačno bazo za izvajanje učnega načrta.

V primeru, ko si na računalnik namestimo brezplačno različico baze Oracle XE, je torej možno vzpostaviti razvojno in produkcijsko okolje APEX brez stroškov za nakup baze oziroma SUPB in razvojnega orodja.

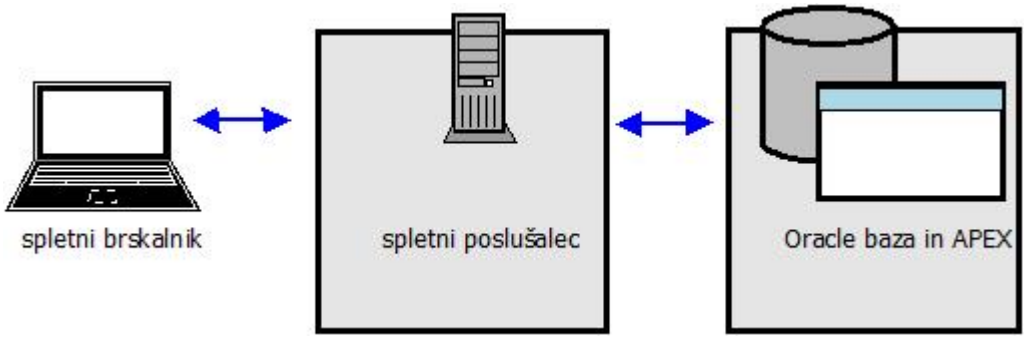
3.2 Način delovanja

Orodje Oracle APEX uporablja spletni brskalnik na uporabnikovem računalniku in preko spletnega poslušalca (angl. *web listener*) komunicira z Oracle bazo oziroma Oracle SUPB. Shematsko je način delovanja prikazan na Sliki 9. Pod pojmom uporabnik tu razumemo tako končnega uporabnika izdelanih programskih rešitev kot tudi razvijalca teh rešitev. Noben od njiju na svojem računalniku ne potrebuje nobenih dodatno nameščenih

programov ali datotek, samo brskalnik. APEX spletne strani se v uporabnikovem brskalniku tvorijo z uporabo HTML (Koratamaddi, 2015).

Namestitev orodja Oracle APEX v bazi pravzaprav pomeni, da imamo v bazi nameščeno še eno dodatno shemo, ki je sicer navadnim uporabnikom nedosegljiva. V shemi so postavljene namenske tabele, imenujemo jih lahko APEXove tabele in namenske PL/SQL procedure, imenujemo jih lahko APEXove PL/SQL procedure. To vse skupaj lahko imenujemo APEXov mehanizem (angl. *APEX engine*). V APEX shemi za APEX verzijo 4.2 je preko 300 tabel in 200 PL/SQL objektov, ki vsebujejo več kot 3000.000 vrstic kode. Spletne strani, ki jih vidi uprabnik, niso shranjene v bazi, ampak se ob klicu, prejetem iz brskalnika preko spletnega poslušalca, generirajo dinamično v realnem času iz metapodatkov, ki so shranjeni v APEX repozitoriju (Oracle, 2016b).

Slika 9: Arhitektura APEX



Vir: Oracle, Oracle Application Express Learn More, b.l..

Ob kreiranju ali spreminjanju uporabniške aplikacije APEX kreira ali spreminja metapodatke v svojih tabelah v bazi. Ko uporabnik požene aplikacijo, kar izvede s klikom na spletno povezavo ali gumb na spletni strani, APEXov mehanizem prebere metapodatke in skreira in pošlje brskalniku zahtevano spletno stran za prikaz ali pa izvede zahtevano akcijo.

Običajno pod pojmom trislojna tehnologija spletnih aplikacij razumemo naslednjo razmestitev:

1. predstavitveni nivo, kar pomeni spletni brskalnik na računalniku uporabnika,
2. aplikacijski nivo, kar pomeni poslovno logiko in
3. podatkovni nivo: podatki na podatkovnem strežniku.

Pri takšni razmestitvi se poslovna logika, to je drugi nivo, procesira na neke vrste aplikacijskem strežniku. Pri APEXU pa je drugače: posebnega aplikacijskega strežnika v

navedenem smislu ni več, vse kar je potrebno za izvajanje poslovne logike, je zapisano v bazi, v metapodatkih v APEXovih tabelah in se izvaja s pomočjo APEXovih PL/SQL procedur. Na srednjem nivoju je v verziji APEX 4.2 ostal le še spletni poslušalec, ki zahteve brskalnika prevaja v klice procedur, shranjenih v bazi. Poslovna logika pa na srednjem nivoju ni shranjena.

3.3 Sodelujoči pri razvoju programskih rešitev v APEXu

V delovnem okolju naše organizacije imamo delo z APEXom razdeljeno med tri različne profile informatikov:

1. administrator baze podatkov,
2. administrator razvojnega okolja APEX in
3. razvijalec informacijskih rešitev.

Administrator baze podatkov poleg siceršnjih nalog administriranja podatkovne baze za razvoj na APEXu kreira bazne sheme, namenjene postavitvi podatkovnih tabel in drugih baznih objektov nove programske rešitve. V sodelovanju z razvijalcem oceni potreben prostor za vsako bazno shemo. Ker imamo več baznih strežnikov in več podatkovnih baz, se odloči, v katero bazo bo shemo umestil. Odločitev je precej odvisna od tega, na katere že obstoječe tabele se bo nova programska rešitev navezovala. Izkušnje kažejo, da je priporočljivo, da je shema v isti bazi, kot so vse ali pa vsaj večina tabel z zunanjimi podatki, od katerih je odvisno delovanje nove programske rešitve. Administrator baze tudi namešča APEX v bazo, skrbi za namestitev posodobitev in morebitnih popravkov.

Administrator razvojnega okolja APEX dodeljuje razvijalcem delovne prostore (angl. *workspace)*. **Delovni prostor** je navidezna zasebna baza podatkov (angl. *virtual private database*). Koncept delovnih prostorov omogoča delo večih razvijalcev na isti bazi, pri tem pa vsakdo od njih vidi le svoje objekte, podatke in aplikacije. Vsak delovni prostor je povezan z eno ali več baznimi shemami. Povezava delovnega prostora z bazno shemo omogoča, da lahko razvijalci v tem delovnem prostoru razvijajo aplikacije, ki delujejo nad tabelami te sheme in kreirajo nove bazne objekte v povezani shemi.

APEX omogoča, da ima vsak delovni prostor svojega administratorja. V naši organizaciji imamo delo organizirano tako, da opravlja delo **administratorja delovnega prostora** kar razvijalec sam. V primerih, ko dela v istem delovnem prostoru več razvijalcev, pa je smiselno, da je kot administrator delovnega prostora določen en sam, da niso administratorji kar vsi. Tako je lažje vzdrževati določeno doslednost pri delu.

Razvijalec v APEXu predvsem razvija programske rešitve za končne uporabnike. Kreira bazne objekte, zaslonske maske, poročila. Programske rešitve pripravlja za uporabo in jih uvaja v uporabo. Pri svojem delu se posvetuje in dogovarja z administratorjem baze, z

administratorjem APEXa, s predstavnikom naročnika, to je glavnim uporabnikom in z uporabniki. Pri manjših projektih opravlja hkrati vlogo analitika in programerja, pogosto pa tudi neformalnega vodje projekta. Po končanem razvoju in uvedbi je na voljo za podporo uproabnikom.

Uporabnik opravlja svoje delo z uporabo zanj razvitih programskih rešitev v APEXu. Pred začetkom razvoja mora opredeliti svoje potrebe. Med razvojem sodeluje pri načrtovanju programske rešitve predvsem s svojimi pripombami in dopolnitvami prej opredeljenih zahtev. Pregleduje in testno uporablja izdelano programsko rešitev. Sodeluje pri uvedbi rešitve v uporabo.

3.4 Začetek dela razvijalca

Pred začetkom dela razvijalec potrebuje:

1. spletni brskalnik,
2. dostop do orodja APEX,
3. shemo v bazi podatkov, v kateri bo kreiral bazne objekte za svojo programsko rešitev,
4. delovni prostor na APEXu,
5. uporabniško prijavno ime za APEX in
6. pravice za razvoj aplikacij.

Dostop do orodja APEX je omogočen s spletnim brskalnikom preko URL naslova, ki ga razvijalcu sporoči administrator baze ali administrator APEX okolja. Shemo v bazi podatkov razvijalcu odpre administrator baze pod nekim baznim prijavnim imenom in geslom. Administrator APEX orodja razvijalcu dodeli delovni prostor, APEX prijavno ime in poveže njegov delovni prostor z dodeljeno bazno shemo. Ob kliku na povezavo, torej URL naslov, se odpre prijavno okno APEX orodja, ki zahteva vpis imena delovnega prostora, prijavnega imena razvijalca in njegovega gesla.

3.5 Delovno okolje razvijalca v APEXu

Ob prijavi v delovno okolje za razvijalca se prikažejo štirje glavni sklopi orodja:

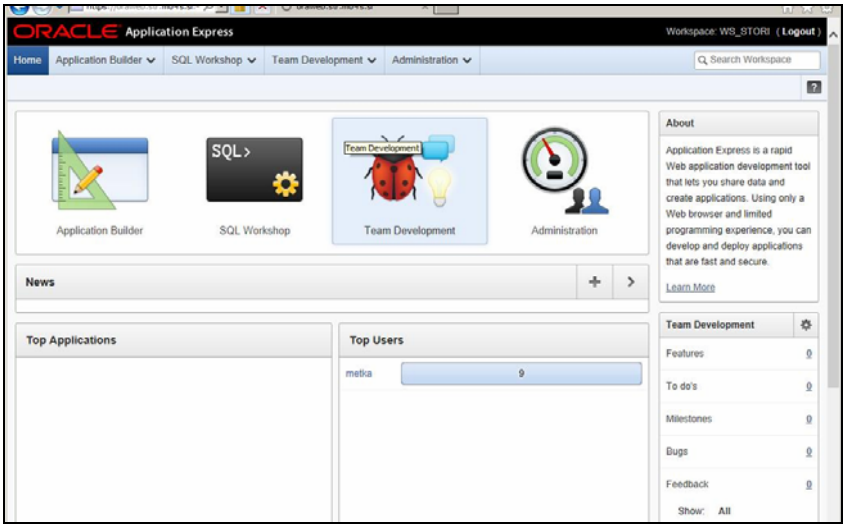
1. graditelj aplikacij (angl. *application builder*),
2. SQL delavnica (angl. *SQL workshop*),
3. skupinski razvoj (angl. *team development*) in
4. administracija (angl. *administration*).

Začetna slika delovnega okolja razvijalca je prikazana na Sliki 10.

Graditelj aplikacij je kot že ime pove namenjen razvoju uporabniških rešitev, torej:

1. oblikovanju zaslonskih obrazcev,
2. definiranju poslovne logike aplikacije,
3. oblikovanju poročil in
4. urejanju skupnih komponent aplikacije.

Slika 10: APEX - delovno okolje razvijalca



Skupne komponente aplikacije so tisti elementi, ki niso določeni za posamezno spletno stran aplikacije, ampak so skupni vsem spletnim stranem, ki sestavljajo aplikacijo. Primeri skupnih komponent so nastavitve avtentikacije, nastavitve avtorizacije, predloge za uporabniški vmesnik in elementi navigacije, kot so zavihki in drobtinice.

Če nismo preveč zahtevni, lahko preko graditelja aplikacij hkrati s kreiranjem zaslonskih mask postavimo tudi bazne objekte, torej tabele s podatki in nekatere pridružene objekte (ključe, sekvence). Delamo z uporabo čarovnikov, ki nas vodijo skozi postopek. Aplikacija, izdelana z APEXom, je zbirka spletnih strani, ki so medsebojno povezane preko zavihkov, gumbov ali hiperpovezav. Vse spletne strani, povezane v eno aplikacijo, imajo skupno metodo avtentikacije uporabnikov.

SQL delavnica omogoča:

1. pregledovanje baznih objektov,
2. pisanje in izvajanje SQL ukazov in pregledovanje rezultatov poizvedb,
3. pisanje in izvajanje SQL skript,
4. gradnjo baznih poizvedb,
5. uporabo raznih pripomočkov za delo s podatki in baznimi objekti in
6. kreiranje in upravljanje Oracle REST podatkovnih storitev.

Oracle REST (angl. *representational state transfer*) podatkovne storitve so javanska različica http strežnika, ki omogoča tudi realizacijo spletnih storitev.

Sklop **skupinski razvoj** je podpora načrtovanju in spremljanju realizacije projekta izgradnje informacijske rešitve. Omogoča evidentiranje funkcionalnosti sistema, ki jih je potrebno razviti, dodelitev nalog razvoja teh funkcionalnosti posameznih razvijalcem, določitev mejnikov razvoja, povezovanje funkcionalnosti z mejniki. Omogočeno je verzioniranje.

Sklop **administracija** omogoča upravljanje računa uporabnika APEXa. Glede na dodeljene pravice omogoča kreiranje novih uporabnikov, dodeljevanje pravic uporabnikom, kreiranje skupin uporabnikov in urejanje članstva v skupinah, spremljanje aktivnosti uporabnikov.

3.6 Oblikovanje baze v APEXu

Na klasičen način kreiramo bazne objekte z uporabo ukazov jezika za definiranje podatkov (angl. *data definition language*, v nadaljevanju DDL). Orodje APEX pa ima na voljo razvijalcu prijazen vmesnik za kreiranje tabel v bazi. Ni nam potrebno pisati DDL ukazov in jih izvajati, ampak si lahko pomagamo s čarovnikom, ki nas vodi po korakih. V ekranskem obrazcu lahko naštejemo polja tabele in določimo njihov tip, dolžino in druge lastnosti. Prav tako s pomočjo čarovnika lahko ustvarimo ključe in prožilce, pa tudi bazno zaporedje za avtomatsko generiranje vrednosti primarnega ključa. V zadnjem koraku pred dejansko izvedbo nam čarovnik pokaže tudi DDL stavek, kot ga je sestavil iz naših odločitev, ki smo jih zabeležili s pomočjo čarovnika.

Podobno lahko s čarovnikom oziroma grafičnim vmesnikom oblikujemo tudi razne poglede na bazne tabele. Enostavnost uporabe grafičnega in prijaznega vmesnika za kreiranje tabel v bazi pa nikakor ne zmanjšuje pomena razmisleka o entitetnih tipih in njihovih povezavah, kar je predpogoj za smiselno kreiranje tabel v bazi.

3.7 Alternativa ali dopolnitev pri oblikovanju baze: orodje SQL Developer

Pri postavitvi baznih objektov si lahko pomagamo tudi z orodjem SQL Developer. **Oracle SQL Developer** je brezplačno grafično orodje, prosto dostopno pod OTN licenčnimi pogoji (Oracle, 2016d). Oraclovo tehnološko omrežje (angl. *Oracle Technology Network*, v nadaljevanju OTN) je obsežna zbirka spletnih strani, ki jo je podjetje vzpostavilo z namenom promocije svojih proizvodov in podpore zlasti tehničnemu kadru pri svojih odjemalcih. OTN licenčni pogoji v glavnem od nas zahtevajo, da programske opreme ne

izvažamo v določene države, da ne omogočamo uporabe programske opreme državljanom teh držav in osebam, ki so na seznamih kriminalcev in drugih seznamih prepovedi, ki jih vodijo organi Združenih držav Amerike in da z uporabo programske opreme ne kršimo zakonodaje Združenih držav Amerike (Oracle, 2016f).

Orodje je namenjeno izboljšanju učinkovitosti in poenostavitvi opravljanja vsakodnevnih opravil na bazi podatkov. Z njim lahko lažje izvajamo naslednja opravila (Serhal, Srivastava, Koratamaddi, & Clement, 2011b, str. B-3):

1. pregledovanje in upravljanje baznih objektov,
2. izvajanje SQL stavkov in skript,
3. urejanje in razhroščevanje PL/SQL stavkov in
4. izdelavo poročil.

Z orodjem SQL Developer se lahko prijavimo v katerokoli standardno oracle bazno shemo. Za prijavo potrebujemo oracle prijavno ime in geslo, ki nam ju dodeli administrator baze. Od pravic, ki jih imamo dodeljene k temu prijavnemu imenu, je odvisno, do katerih baznih objektov bomo lahko dostopali in katere operacije bomo lahko izvajali z orodjem.

Orodje si na računalnik namestimo preprosto tako, da v izbrano datotečno mapo namestimo programske datoteke, ki smo jih naložili s spletne strani proizvajalca (Oracle, 2016e). Za bolj udobno delo si kreiramo še bližnjico za zagon programa na namizju. Za namestitev na osebni računalnik ne potrebujemo administratorskih pravic. Moramo pa poznati parametre za dostop do baze podatkov. Te nam lahko priskrbi administrator baze.

3.8 Razvoj uporabniškega vmesnika v APEXu

Ločimo imperativno in deklarativno programiranje. Imperativno ali postopkovno programiranje je način, ko računalniku korak za korakom povemo, kaj naj naredi. Deklarativno ali nepostopkovno programiranje je način, kjer opišemo, kaj želimo, ne opisujemo pa korakov, kako do tega priti. Razvoj programske rešitve v APEX okolju poteka z uporabo čarovnikov. Torej programiramo deklarativno. Na voljo so čarovniki za (Mandya & Mani, 2013):

1. izdelavo zaslonskih obrazcev,
2. izdelavo klasičnih poročil,
3. izdelavo interaktivnih poročil, ki si jih uporabnik lahko prilagaja,
4. izdelavo elementov za navigacijo po spletnih straneh, kot so zavihki in drobtinice
5. in še mnogo drugih.

Uporabniški vmesnik v APEX aplikaciji je zbirka medsebojno povezanih spletnih strani. Spletne strani je treba tudi vizuelno lepo in uporabno izdelati. V pomoč za to opravilo

imamo v APEXu na voljo zbirko vnaprej pripravljenih predlog, ki jih lahko takoj uporabimo.

3.9 Zagotavljanje varnosti

3.9.1 Informacijska varnost

Informacijska varnost pomeni varstvo podatkov in informacijskih sistemov pred nezakonitim dostopom, uporabo, razkritjem, ločitvijo, spremembo in uničenjem. Pri gradnji informacijskega sistema z APEX razvojnim orodjem proizvajalec priporoča upoštevanje naslednjih varnostnih priporočil (Oracle, 2016a):

1. varnostna priporočila za administratorja,
2. varnostna priporočila za razvijalca,
3. varno nalaganje datotek,
4. ugotavljanje identitete uporabnika preko avtentikacije in
5. zagotavljanje varstva podatkov preko avtorizacij.

Varnostna priporočila za administratorja se nanašajo na varno uporabo različnih tehnoloških rešitev, kot so povezave z drugimi produkti in vzpostavitev mrežnih storitev. Varnostna priporočila za razvijalca so namenjena predvsem preprečevanju poskusov različnih vrst varnostnih vdorov. APEX razvijalcu omogoča zelo hiter razvoj aplikacije, s katero uporabnik lahko shranjuje v bazo datoteke različnih vrst in objavlja na spletni strani dostop do njih in omogoča tudi drugim uporabnikom prevzemanje teh datotek s spletne strani na lokalni računalnik. Da bi preprečili dostop do datotek uporabnikom, ki jim te niso namenjene, je pri gradnji aplikacij treba upoštevati priporočila za zagotavljanje varnega nameščanja datotek.

3.9.2 Avtentikacija ali preverjanje istovetnosti

Avtentikacija je proces preverjanja identitete uporabnika preden lahko dostopa v sistem. Uporabnik se lahko identificira s svojim geslom, z digitalnim potrdilom ali pa z varnim ključem.

Končnim uporabnikom lahko uredimo prijavo v APEX aplikacijo na več načinov. Lahko jim kreiramo uporabniško prijavno ime v APEX okolju in jim tu urejamo gesla, lahko pa se prijavljajo v aplikacije s prijavnim imenom, ki smo jim ga skreirali v bazi. Možna je tudi prijava z uporabo internetnega protokola za dostop do imenikov na osnovi arhitekture odjemalec strežnik (angl. *lightweight directory access protocol,* v nadaljevanju LDAP).

Za vsako od aplikacij določimo, katero od preddefiniranih vrst avtentikacijskih shem naj uporablja:

1. brez avtentikacije – za uporabo aplikacije je dovolj že, da poznamo njen URL naslov, uporabniki ne uporabljajo prijavnih imen,
2. APEX uporabniško prijavno ime,
3. bazni uporabniški račun in
4. LDAP avtentikacija.

Vsak od navedenih načinov ima svoje prednosti in slabosti. Aplikacije, pri katerih ne zahtevamo avtentikacije uporabnikov, omogočajo takojšen dostop do vsebine vsakomur, nad uporabo nimamo nadzora. Pri avtentikaciji z APEX prijavnim imenom je treba za vsakega novega uporabnika v APEXu kreirati prijavno ime in mu dodeliti geslo in ga o tem na zaupen način obvestiti. Uporabnik si mora prijavno ime in geslo zapomniti in ju varovati.

V primerih, ko imajo uporabniki aplikacije že dodeljena prijavna imena in gesla za prijavo na podatkovno bazo, je smiselna odločitev za avtentikacijo s temi prijavnimi imeni in gesli tudi v APEX aplikacijo. Na ta način administratorju APEXa ni treba kreirati novih prijavnih imen, jih razpošiljati in o njih voditi evidence, uporabnikom pa ni treba skrbeti še za eno dodatno prijavno ime in geslo. Podobno lahko uporabimo LDAP avtentikacijo v primeru, ko imajo uporabniki že dodeljeno prijavno ime in geslo za uporabo omrežja. Omogočimo jim lahko, da se s tem prijavnim imenom in geslom prijavljajo tudi v APEX aplikacijo.

3.9.3 Avtorizacija ali pooblastila za uporabo

Avtorizacija pomeni prepoznavanje upravičenosti uporabe določenih storitev ali podatkov. APEX glede na dodeljene pravice pozna tri vrste uporabnikov okolja APEX: administratorje delovnega prostora, razvijalce in končne uporabnike. Administratorji delovnega prostora lahko kreirajo in spreminjajo aplikacije in bazne objekte, upravljajo uporabniške račune in skupine. Razvijalci lahko kreirajo in spreminjajo aplikacije in bazne objekte. Končni uporabniki lahko samo uporabljajo aplikacije, nimajo možnosti razvoja. Naloga razvijalca aplikacije je, da poskrbi za to, da ima vsak uporabnik dostop do natančno tistih delov aplikacije in do tistih podatkov, ki jih potrebuje za opravljanje svojega dela.

3.10 Orodje Oracle XML DB

Orodje Oracle XML DB je zbirka Oraclovih baznih tehnoloških rešitev, namenjenih za učinkovito upravljanje s podatki v XML obliki. Orodje omogoča (Serhal, Srivastava, Koratamaddi & Clement, 2011a):

1. shranjevanje podatkov v XML obliki,
2. kreiranje podatkov v XML obliki,

3. dostop do podatkov, shranjenih v XML obliki,
4. iskanje po podatkih,
5. preverjanje,
6. transformacijo in
7. indeksiranje podatkov v XML obliki.

Orodje je kot njen del vključeno v Oracle bazo oziroma njen SUBP. Ključna elementa orodja sta:

1. podatkovni tip XMLType: v stolpce in tabele tega tipa lahko shranjujemo podatke v XML formatu in
2. Oracle XML repozitorij, ki omogoča organizacijo in upravljanje vsebine baze na način datotek in datotečnih map, ki jih imenujemo viri (angl. *resources*). Dostop do repozitorija je možen z uporabo uveljavljenih spletnih protokolov, kot so FTP, HTTP in odprtokodni WebDAV. Na tak način realiziran dostop do podatkov je hiter in zanesljiv enako kot do kateregakoli datotečnega sistema.

3.11 Uvedba izdelane rešitve v uporabo

Za uvedbo izdelane rešitve v uporabo je treba postoriti vsaj naslednje:

1. vzpostavitev začetnega stanja podatkov v bazi,
2. vzpostavitev nadpovezave za dostop do programske rešitve.

Začetno stanje podatkov običajno obsega polnjenje vsebine šifrantov, morebitne prevedbe podatkov in polnjenje podatkov iz starega v nov sistem. Nadpovezavo do programske rešitve (URL naslov) objavimo na uporabniku dostopni spletni strani. V skrajnem primeru, kadar v internem omrežju organizacije nimamo vzpostavljenih primernih internih spletnih strani za objavo povezave, lahko uporabniku povezavo pošljemo po elektronski pošti in si jo lahko pripne na namizje. Na uporabnikov računalnik običajno ni potrebno nameščati ničesar, če ima le kolikor toliko sodoben spletni brskalnik. Začetek dela uporabnika z novo programsko rešitvijo je preprost:

1. klikne na povezavo,
2. v prijavno masko vnese prijavno ime in geslo,
3. odpre se mu prva spletna stran in
4. lahko začne delati s programom.

3.12 Agilne metodologije in APEX

Če beremo Agilni manifest, takoj opazimo, da je kratek, enostaven in osredotočen na bistvo. Podobno lahko gledamo tudi na APEX kot razvojno orodje. Deklarativen način razvoja in preprosta arhitektura orodja pripomoreta, da nam je prihranjeno veliko rutinskega dela. Razvoj je zato hitrejši in preprostejši, hkrati pa robusten (Cimolini & Cannell, 2012, str. 15).

APEXovo deklarativno okolje za hiter razvoj aplikacij spodbuja hitro izdajanje nove programske kode in omogoča hiter odziv na spremembe zahtev. Z modulom za timsko delo spodbuja duh sodelovanja, sooodvisnosti in timskega dela med vsemi deležniki na projektu. S tme orodjem lahko hitro razvijemo progmrakse rešitve, ki dobro realizirajo tako poslovne zahteve kot tudi tehnične zahteve (Cimolini & Cannell, 2012, str. 25).

Praksa kaže, da so programske rešitve, ki jih izdelamo z APEXom, zanesljivi in varni programi. Enostavno povedano, ko je program enkrat v logičnem smislu pravilno izdelan, z njim ni več težav, deluje zanesljivo. To zanesljivost najbrž da lahko pripišemo dejstvu, da so z APEXom izdelane programske rešitve shranjene v bazi v obliki PL/SQL programskih enot. PL/SQL je Oraclov skriptni jezik, ki ga je podjetje dodalo k svoji bazi leta 1991, torej je star 25 let. Po tolikšnem času uporabe, izboljšav in razširitev je to zrela tehnologija, ki je v svoji zgodovini prestala številne cikle testiranja, razhroščevanja in optimizacije (Cimolini & Cannell, 2012, str. 17).

Tudi spletne tehnologije, na katerih sloni APEX, so že dosegle nek dovolj visok nivo zrelosti, saj že dolgo niso nekaj novega. Z izredno razmahnitvijo interneta so tudi spletne tehnologije prestale ogromno testiranja v živo, pri sami množični uporabi.

Z zrelostjo infrastrukture in razvojnih orodij tu mislimo na dejstvo, da nam uporaba te infrastrukture in orodij z veliko verjetnostjo omogoča uspešen zaključek oziroma konvergenco projekta.

Z orodjem APEX smo razvijalci dobili način, da lahko hitro delamo po osnovnih principih izgradnje baze in aplikativnih rešitev. Ni se nam treba toliko ukvarjati z zmožnostjo baze, saj je tudi Oracle SUPB po skoraj 30 letih uporabe, nadgradenj in izboljšav zaradi svoje zrelosti stabilna tehnologija.

Pred prihodom APEX orodja v našo organizacijo smo manjše programske rešitve nad relacijsko bazo izdelovali z orodjem Microsoft access. Orodje ima možnost izdelave lastne baze podatkov ali pa povezavo programske rešitve preko ODBC povezljivosti na drugo podatkovno zbirko, tudi na Oracle bazo. V praksi se nam je dogajalo, da smo tudi z upoštevanjem vseh znanih principov izdelave programskih rešitev s tem orodjem naleteli na situacijo, ko je rešitev v nekem trenutku kar nehala delovati. MS Access ima tudi

številne omejitve, kot je na primer majhno dovoljeno število hkratnih uporabnikov. Ko smo prešli na izdelavo rešitev z orodjem APEX, se nam je bistveno poenostavilo delo. Ko so bile enkrat rešitve izdelane in preizkušene, so delovale in z njimi ni bilo več dela. Vsaka novejša verzija orodja pa prinaša s sabo še večjo pomoč razvijalcu.

4 PROCES ZAGOTAVLJANJA PODPORE DRŽAVE GOSTITELJICE

4.1 Podpora države gostiteljice

S pojmom **podpora države gostiteljice** (angl. *host nation support*, v nadaljevanju HNS) označujemo vojaško in civilno pomoč, ki jo zagotavlja Republika Slovenija silam in organizacijam, ki se zadržujejo na njenem ozemlju ali se gibljejo prek njega, so v njenih teritorialnih vodah ali v zračnem prostoru in tam delujejo ali se usposabljajo (Vlada Republike Slovenije, 2015, v nadaljevanju Vlada RS, 2015).

Republika Slovenija je kot članica zveze NATO dolžna zagotavljati podporo države gostiteljice zavezniškim silam. Vlada Republike Slovenije je v letu 2003 sprejela prvi Načrt zagotavljanja podpore države gostiteljice, s katerim je to področje na načelni ravni prvič celovito uredila. Dejavnost podpore države gostiteljice je v skladu s tem načrtom potekala v okviru logistike in civilno-vojaškega sodelovanja.

Zagotavljanje podpore države gostiteljice pomeni kontinuiran proces mednarodnega dogovarjanja, dopolnjevanja načrtov in usposabljanja izvajalcev, sprotnega prilagajanja infrastrukture, namestitvenih in oskrbnih zmogljivosti, kot tudi potrebam zagotavljanja podpore države gostiteljice ter nenazadnje zbiranja, obdelave in vnosa spremenjenih podatkov v kataloge zmogljivosti in računalniško vodene baze podatkov držav članic Nata (Pipenbaher, 2006).

V letu 2015 je Vlada Republike Slovenije sprejela nov Načrt zagotavljanja podpore države gostiteljice, ki ureja področje gostovanja vojaških misij na območju Republike Slovenije. V njem je opredelila, kaj obsega področje načrtovanja in podpore države gostiteljice, baze podatkov, ki se v tem procesu uporabljajo, obseg in zahtevnost načrtovanja in definirala korake pri načrtovanju podpore. Opredeljeno je tudi izvajanje podpore. V dokumentu so upoštevana določila Zakona o obrambi, Uredbe o obrambnem načrtovanju, določila Resolucije o strategiji nacionalne varnosti Republike Slovenije iz leta 2010 in Obrambne strategije Republike Slovenije iz leta 2012.

Načrt zagotavljanja podpore opredeljuje štiri cilje na področju zagotavljanja podpore države gostiteljice (Vlada RS, 2015):

1. v okviru zmožnosti zagotoviti celovito podporo silam in organizacijam, ki prehajajo ozemlje ali so razmeščene na ozemlju, v teritorialnih vodah ali zračnem prostoru Republike Slovenije,
2. izpolnjevati del mednarodnih obveznosti na obrambnem področju,
3. izvajati priprave za podporo države gostiteljice v obsegu, določenem s sporazumi, tako da se v čim krajšem času zagotovijo vojaški in civilni viri in zmogljivosti in
4. zagotavljati in vzdrževati baze podatkov za podporo države gostiteljice.

Zagotavljanje podpore države gostiteljice se začne z vstopom sil držav pošiljateljic ali organizacij pošiljateljic na območje Republike Slovenije, se izvaja v dogovorjenem obsegu in se konča z odhodom sil pošiljateljic iz Republike Slovenije (Vlada RS, 2015).

Postopki in dokumenti načrtovanja podpore temeljijo na pravnih podlagah Republike Slovenije ter zavezniških doktrinah in standardih oziroma upoštevajo rešitve, ki jih določajo NATOva politika in načela logistike ter NATOva doktrina in načela podpore države gostiteljice (Vlada RS, 2015).

4.2 Načrtovanje podpore države gostiteljice

4.2.1 Nosilci podpore

Načrtovanje podpore države gostiteljice usklajuje Ministrstvo za obrambo kot upravitelj obrambnega načrta države. Vojaške zmogljivosti za zagotavljanje podpore načrtuje Generalštab Slovenske vojske s podrejenimi poveljstvi in enotami. Civilne zmogljivosti načrtuje in usklajuje organizacijska enota Ministrstva za obrambo, pristojna za obrambne zadeve. V načrtovanje in zagotavljanje podpore se skladno s svojimi pristojnostmi in delovnimi področji vključujejo tudi drugi organi državne uprave, gospodarske družbe, zavodi in druge organizacije, katerih dejavnost je posebnega pomena za obrambo. Te organizacije imenujemo **nosilci podpore** države gostiteljice. Nosilci podpore sodelujejo predvsem pri pripravi dokumentov, izdaji ustreznih dovoljenj in zagotavljanju oskrbe. Opozarjajo na posebnosti in morebitne omejitve pri izvedbi posameznih aktivnosti (Vlada RS, 2015).

4.2.2 Področja načrtovanja

Načrtovanje in zagotavljanje podpore se nanaša na naslednja področja (Vlada RS, 2015):

1. materialna sredstva, storitve in usluge,
2. kadrovska zagotovitev,
3. infrastrukturne zmogljivosti,
4. zdravstvena oskrba in
5. finančni in pravni vidiki.

Materialna sredstva, storitve in usluge v okviru svojih zmožnosti zagotavlja Slovenska vojska skladno z implementacijskimi sporazumi in dogovori. Obseg, ki presega zmožnosti Slovenske vojske, zagotavljajo nosilci podpore države gostiteljice. Materialna sredstva so po Slovenskem vojaškem standardu razdeljena v pet razredov oskrbe (Vlada RS, 2015):

1. potrošno blago,
2. orožje, vojaška in druga oprema,
3. gorivo in maziva,
4. blago, ki ni določeno v formaciji in
5. strelivo, minskoeksplozivna sredstva in rakete.

Kadrovska zagotovitev pomeni zagotovitev dodatnih kadrov pošiljateljicam, v skladu z dogovorom in ob upoštevanju usmeritev ministrstva, pristojnega za področje dela.

Pri podpori gostovanju vojaških sil se prednostno uporablja infrastruktura Slovenske vojske in infrastruktura, ki je v lasti države. Slovenska vojska zagotavlja zmogljivosti za namestitev, bivanje in usposabljanje. Zagotovitev dodatnih zmogljivosti usklajuje organizacijska enota, pristojna za obrambne zadeve.

Zdravstveno oskrbo izvaja Slovenska vojska v skladu s svojimi zmožnostmi. Zdravstveno oskrbo izvajajo tudi izvajalci zdravstvene dejavnosti v Republiki Sloveniji skladno z usmeritvami ministrstva, pristojnega za zdravje, in zdravstveni zavodi, katerih dejavnost je po odločitvi Vlade Republike Slovenije posebnega pomena za obrambo (Vlada RS, 2015).

V izvedbenih dokumentih in sporazumih se načrtujejo in opredelijo tudi finančni in pravni vidiki zagotavljanja podpore države gostiteljice. Dokumenti s področja financ so osnova za izhodiščno oceno stroškov podpore države gostiteljice. Določijo se postopki in višina povračila stroškov, splošna finančna načela, morebitna davčna oprostitev, uvozne dajatve, takse in duge pristojbine. Vsi stroški, ki jih krije Republika Slovenija, morajo biti vključeni v proračun Republike Slovenije (Vlada RS, 2015).

Za nabavo blaga in izvedbo storitev je mogoče sklepati pogodbe z gospodarskimi družbami, zavodi, drugimi organizacijami in fizičnimi osebami. Za izpolnjevanje pogodb so odgovorne izključno države pošiljateljice oziroma organizacije pošiljateljice ali zavezniška poveljstva. Če pošiljateljice to zahtevajo, se jim zagotovi seznam dobaviteljev blaga in izvajalcev storitev, s katerimi ima Republika Slovenija v postopkih javnega naročanja že sklenjene pogodbe. Plačevanje poteka med pogodbenima strankama. Na nacionalni ravni se v primeru potrebe oblikujejo svetovalne skupine za pomoč pri sklepanju pogodb, sestavljene iz predstavnikov Slovenske vojske in organizacijskih enot Ministrstva za obrambo oziroma uprav za obrambo na regijski ravni (Vlada RS, 2015).

4.2.3 Obseg načrtovanja

Načrtovanje podpore se lahko izvaja na dva načina: načrtovanje v celotnem obsegu ali načrtovanje v delnem obsegu.

Načrtovanje v celotnem obsegu se izvaja v naslednjih primerih (Vlada RS, 2015):

1. ko zahtevo posreduje zavezniško poveljstvo strateške ravni in predvideva razmestitev zavezniških sil v Republiki Sloveniji ali tranzit teh sil prek njenega ozemlja,
2. med združenimi vojaškimi operacijami pod vodstvom zavezniškega poveljstva,
3. med humanitarno pomočjo ali nalogami kriznega odzivanja širših razsežnosti in
4. med vojaškimi vajami pod vodstvom zavezniškega poveljstva.

Načrtovanje v celotnem obsegu poteka po naslednjih korakih (Vlada RS, 2015):

1. korak: uskladitev memoranduma o soglasju,
2. korak: priprava koncepta potreb,
3. korak: priprava izjav o potrebah,
4. korak: priprava tehničnega sporazuma in
5. korak: priprava skupnega izvedbenega dogovora.

Načrtovanje v delnem obsegu s prilagojenimi koraki se uporablja predvsem pri podpori silam držav ali organizacij pošiljateljic, ki se gibljejo preko ozemlja Republike Slovenije. To je redna dejavnost glavnih nosilcev podpore oziroma nosilcev zagotavljanja tranzita. Osrednjo vlogo pri tem ima organizacijska enota, ki opravlja nalogo Nacionalnega centra za koordinacijo vojaških premikov. Načrtovanje v delnem obsegu je v Načrtu predvideno za dve različni situaciji (Vlada RS, 2015):

1. tranzit preko ozemlja Republike Slovenije in
2. dvo ali večstranske vojaške vaje v organizaciji Slovenske vojske.

4.3 Zagotavljanje podpore države gostiteljice

Po izvedbi petih korakov načrtovanja, katerih rezultat je skupni izvedbeni dogovor, sledijo koraki izvedbe podpore (Vlada RS, 2015):

1. pridobivanje in izdajanje dovoljenj za prehod državne meje,
2. sprejem sil držav ali organizacij pošiljateljic,
3. zagotavljanje podpore v dogovorjenem obsegu, ki se uresničuje z oskrbo in svetovanjem in
4. odhod tujih oboroženih sil z ozemlja Republike Slovenije.

4.4 Zbirki podatkov za zagotavljanje podpore države gostiteljice

Slovenska vojska za načrtovanje podpore države gostiteljice uporablja zavezniški programski produkt, imenovan Informacijski sistem logistike (angl. *Logistics Functional Area Services*, v nadaljevanju LOGFAS).

Za načrtovanje civilnih zmogljivosti se podatki zagotavljajo in posodabljajo v bazi, imenovani EHONAS, katere skrbnik je organizacijska enota, pristojna za obrambne zadeve. Za vzdrževanje podatkov v bazi in za pripravo informacij iz baze se uporablja programska rešitev z istim imenom, ki smo jo razvili pred petimi leti in ki jo želimo zdaj posodobiti in prilagoditi vlogi, ki jo ima v skladu z novim Načrtom zagotavljanja podpore države gostiteljice.

Novi Načrt zagotavljanja podpore države gostiteljice predvideva naslednjo funkcionalnost programske rešitve EHONAS (Vlada RS, 2015):

1. vodenje podatkov o zmogljivostih letališč,
2. vodenje podatkov o cestnih vstopnih točkah in nekaterih deponijah ob avtocestnem križu,
3. vodenje podatkov o železniških vstopnih točkah, železniških tovornih postajah,
4. vodenje podatkov o pomorskem pristanišču,
5. vodenje podatkov o izvajalcih zdravstvene dejavnosti,
6. vodenje podatkov o bencinskih servisih,
7. vodenje podatkov o nastanitvenih objektih oziroma objektih in območjih, primernih za podporo države gostiteljice,
8. vodenje podatkov o krajih in osebah za stike in povezave nosilcev podpore države gostiteljice,
9. vzdrževanje in objava seznama nacionalnih normativnih aktov in dokumentov, ki določajo postopke zagotavljanja podpore države gostiteljice,
10. vzdrževanje in objava seznama skupnih politik, načel, doktrin, postopkov in standardov Nata, povezanih z zagotavljanjem podpore države gostiteljice,
11. objavljanje vzorcev obrazcev za zagotavljanje podpore države gostiteljice in
12. objavljanje vzorcev dokumentov, namenjenih zagotavljanju podpore države gostiteljice.

Programska rešitev mora v skladu z Načrtom omogočati, da nosilci podpore dokumente in zbirke podatkov dopolnjujejo neposredno v bazi podatkov EHONAS. Dopolnitve oziroma spremembe morajo evidentirati ob njihovem nastanku. Nosilci podpore vzdržujejo dokumente in podatke v skladu s svojimi pristojnostmi, torej v skladu s svojim delovnim resorjem oziroma področjem dela (Vlada RS, 2015).

Dostop do baze podatkov naj bo načrtovalcem podpore države gostiteljice iz Slovenske vojske omogočen na internem omrežju Ministrstva za obrambo. Nosilci podpore države gostiteljice naj imajo dostop omogočen na omrežju Nacionalnega centra za krizno upravljanje (Vlada RS, 2015).

5 ANALIZA TRENUTNEGA STANJA INFORMACIJSKE PODPORE

5.1 Dva programska proizvoda

Načrtovanje podpore gostovanju vojaških misij je v skladu z Načrtom zagotavljanja podpore države gostiteljice razdeljeno na načrtovanje vojaških zmogljivosti in na načrtovanje civilnih zmogljivosti. Načrtovanje vojaških zmogljivosti je informacijsko podprto z zavezniškim programskim proizvodom LOGFAS. Za načrtovanje civilnih zmogljivosti organizacijska enota, pristojna za obrambne zadeve, uporablja manjši programski proizvod, ki smo ga z lastnimi viri izdelali pred približno šestimi leti. Čeprav gre za manjši program, pa je nastajal več let.

5.2 Prvi poskus informatizacije

Prvi poskus vzpostavitve nekega informacijskega sistema za področje načrtovanja civilnih zmogljivosti se je pričel z idejo o podatkovno precej obsežnem informacijskem sistemu. V njem bi vodili veliko vrst različnih podatkov, ki bi lahko bili v pomoč pri načrtovanju podpore obiska tuje vojaške enote. Ideja je prišla s strani nosilca procesa nudenja podpore. Po nekaj začetnih sestankih vodstva službe za informatiko in vodstva organizacijske enote – nosilca procesa smo ustanovili manjšo delovno skupino, ki je imela nalogo izdelati tak sistem. V delovni skupini smo bili glavni uporabnik in dva informatika. Imeli smo nekaj začetnih sestankov z namenom prve analize zahtev, da bi izdelali nek grobi okvir bodočega informacijskega sistema. Drugače povedano, da bi sploh ugotovili, kaj konkretno je potrebno informatizirati in kako bi to lahko storili.

Na delovnih sestankih se je izkazalo, da ima glavni uporabnik zbranih veliko različnih dokumentov v papirni obliki, ki jih je pridobil od raznih državnih inštitucij. Vsebina na dokumentih je bila z različnih področij (gospodarstvo, infrastruktura, zdravstvo,..). Želja je bila, da bi vsebino teh dokumentov prelili v nek informacijski sistem, iz katerega bi se dalo v vsakem trenutku pridobiti uporabne informacije za načrtovanje namestitev tujih vojaških enot. Že na prvih sestankih smo skupaj ugotovili, da je za realizacijo te ideje treba premagati več ovir:

1. Pridobljeni dokumenti so bili večinoma izdelani na papirju, potrebno bi bilo vse podatke z dokumentov ročno prenesti v informacijski sistem, če jih želimo potem

obdelovati ali prikazovati. Kot vedno pri ročnem vnašanju podatkov je tudi tu obstajala velika možnost nastanka napak pri prepisovanju.

2. Dokumente so pripravile različne inštitucije, vsaka v svoji obliki oziroma formatu. Da bi vse podatke z dokumentov organizirali v neko logično zbirko podatkov, bi se bilo treba lotiti obsežne analize strukture teh dokumentov: kaj sploh vsebujejo, podatke o katerih entitetah, katere atribute entitet. Predvidoma bi za vsako entiteto bilo treba v bazi podatkov izdelati vsaj eno tabelo in ustrezno vnosno formo. Takoj v naslednjem koraku smo slutili veliko množico novih šifrantov, ki bi jih bilo treba vzpostaviti in po možnosti poenotiti z že obstoječimi v našem informacijskem sistemu.
3. Vsi prejeti dokumenti so bili posnetek trenutnega stanja (na primer seznam bencinskih črpalk po občinah) in ker ni bilo zagotovljenega posodabljanja podatkov, niso bili zanesljivi oziroma jih je bilo že takoj treba obravnavati kot zastarele.

Skupna ugotovitev je bila, da je ob vzpostavitvi podatkovno tako obsežnega informacijskega sistema nujno zagotoviti bolj zanesljiv zajem podatkov in vzpostaviti mehanizme za vzdrževanje ažurnosti podatkov. Najbolje bi bilo, da uredimo vnos podatkov tam, kjer nastajajo oziroma, da jih z zanesljivo izmenjavo datotek prenašamo iz informacijskih sistemov, v katerih so nastali, v naš sistem. Za realizacijo teh načinov bi bilo potrebno skleniti veliko tehnično podrobnih medresorskih dogovorov in jih tudi realizirati v obstoječih in v našem novem informacijskem sistemu.

Podrobnejši razmislek začetne ideje o neki manjši priročni evidenci nas je hitro pripeljal do razvijalcem informacijskih sistemov znane ugotovitve, da »Ni majhnih aplikacij«. Drugače povedano, če želimo narediti neko uporabno informacijsko rešitev, je v to vedno treba vložiti konkretno količino dela, običajno bistveno več, kot pa je videti v prvem trenutku. Člani delovne skupine smo v tistem času dobili zadolžitve na takrat bolj prioritetnem projektu in tako je naša naloga za nekaj časa obmirovala.

5.3 Drugi poskus

5.3.1 Omejitev zahtev naročnika

Potreba po informacijski podpori gostovanju vojaških misij na področju civilnih zmogljivosti je še vedno obstajala in uporabnik še vedno ni imel rešitve. V želji po neki uporabni rešitvi, ki bi v doglednem času tudi delovala, je glavni uporabnik (naročnik) skrčil in konkretiziral svoje informacijske potrebe. Opustil je idejo o polnjenju podatkovne baze s podatki, zbranimi z dokumenti, ki bi jih pošiljale različne državne inštitucije. Namesto tega se je odločil za zajem podatkov po regionalnem principu in za vnos podatkov na lokacijah, kjer s temi podatki razpolagajo, to je v lokalnem okolju, na upravah za obrambo. Z novim, manjšim naborom svojih zahtev se je nekajkrat sestal s sistemskim analitikom, s katerim sta do atributov natančno opredelila podatke, ki jih je potrebno voditi v novem informacijskem sistemu. Rezultat tako skrčenih zahtev naročnika je ideja o

evidenci civilnih nastanitvenih zmogljivosti, kot prvi korak do bodočega informacijskega sistema civilnih zmogljivosti za podporo gostovanju vojaških misij.

5.3.2 Izbira razvojnega orodja

V tem času je bilo razvojno orodje Oracle APEX v naši organizaciji še nepreizkušeno v praksi. Ničesar še nismo razvili sami s tem orodjem. Glede na to, da je šlo za razvoj manjše evidence in da smo imeli predstavnika naročnika, ki je znal dobro opredeliti svoje potrebe in je bil hkrati dovolj potrpežljiv, da rezultatov ni terjal od nas že »jutri zjutraj«, smo se odločili, da evidenco razvijemo s tem za nas takrat novim orodjem. Orodje smo pridobili hkrati z nakupom licenc za Oracle SUPB in zato odločitev zanj ni pomenila dodatnih stroškov za nakup. Takrat dostopne informacije o orodju so dale slutiti, da je primerno za razvoj spletnih aplikacij nad podatki v relacijski Oracle bazi, da razvoj poteka hitro in da je delovanje izdelkov bolj zanesljivo od delovanja primerljivih rešitev, ki smo jih do tedaj razvijali z orodjem Microsoft Access in so za delovanje zahtevale namestitev na delovnih postajah uporabnikov. Na strategijo razvoja informatike naše organizacije se takrat pri izbiri orodja za razvoj spletnih aplikacij nismo mogli opreti, saj tega ni določala. Smo pa menili, da bodo izkušnje, pridobljene pri uporabi orodja, lahko v veliko pomoč pri pripravi strategije na tem področju.

Na začetku je delo potekalo počasi, kot vedno pri uporabi novega orodja. Predvsem se je izkazalo, da lahko osnovne stvari naredimo z orodjem APEX presenetljivo hitro, če pa želimo nekaj točno določenega v tehničnem smislu, pa to lahko zelo podaljša čas razvoja. Ko smo se spoznali z orodjem, razumeli njegovo »filozofijo« in z njo uskladili svoja pričakovanja, smo lahko hitro izdelali lepo uporabniško rešitev.

V bistvu smo z uporabo orodja, ki ga proizvajalec oglašuje kot orodje primerno za hiter razvoj aplikacij (RAD), razvili informacijsko rešitev po klasični slapovni metodi. V začetnih aktivnostih smo identificirali praktično vse uporabnikove potrebe.

5.3.3 Analiza in načrtovanje

Sodelavec, ki je opravil nalogo sistemske analize in mi predal njene rezultate, je razen nekaj podrobnosti identificiral že vse tabele in atribute, pa tudi glavne ključe. Ko sem pregledala njegovo delo, sem imela pogovor s predstavnikom naročnika, torej glavnim uporabnikom, kjer sva razčistila le še manjše detajle in opredelila način vzpostavitve šifrantov. Sprva smo imeli namen vzpostaviti od informacijskega sistema naše organizacije povsem avtonomen informacijski sistem, v katerem bi sami vodili tudi vse potrebne šifrante. Po premisleku smo ugotovili, da bodo naše tabele v bazi realizacija entitetnih tipov, ki se sicer nekateri informatizirajo prvič, nekateri pa v informacijskem sistemu organizacije že imajo realizacijo v obliki tabel. Zato jih ni smiselno ponovno uvajati z

novim načinom šifriranja, ampak se je potrebno navezati na obstoječe tabele, v našem primeru šifrante. Takšne že obstoječe šifrante smo našli v registru prostorskih enot.

Ugotovitev o potrebni navezavi na register prostorskih enot nam je bila tudi vodilo pri izbiri baze, v katero bomo umestili bazne objekte nove programske rešitve. Skupaj s sistemsko administratorko smo se odločili, da je najbolje, da je nova bazna shema EHONAS v isti bazi kot je register prostorskih enot. Tako se bomo lahko na šifrante iz tega registra navezali kar z dodelitvijo ustreznih pravic in kreiranjem pogledov na šifrante. Sicer bi pa morali kreirati in uporabljati medbazne povezave.

5.3.4 Razvoj sistema

Po dodeljenem prostoru za bazne objekte sem se lahko kar lotila dela na razvoju podatkovne strukture in uporabniškega vmesnika. Z orodjem SQL delavnica, ki je del APEX razvojnega okolja, sem kreirala bazne tabele, postavila glavne ključe, prožilce in tabele povezala s tujimi ključi.

Prožilce na tabelah smo potrebovali predvsem za sledenje sprememb na podatkih. Glavni uporabnik je želel sledenje sprememb v smislu beleženja zadnjega posega v vrstico tabele. Ni pa zahteval beleženja zgodovine vseh sprememb na podatkih, kar nam je delo precej olajšalo. Uporabili smo bazne prožilce za naslednje dogodke:

1. pred vpisom nove vrstice v tabelo (angl. *before insert trigger,* v nadaljevanju BI),
2. pred spreminjanjem podatkov v vrstici tabele (angl. *before update trigger,* v nadaljevanju BU) in
3. po brisanju vrstice tabele (angl. *after delete trigger*, v nadaljevanju AD).

V vseh tabelah, za katere se je zahtevalo sledenje, smo vzpostavili štiri dodatna polja za zapise sledenja:

P1: polje za zapis prijavnega imena uporabnika, ki je kreiral novo vrstico,
P2: polje za zapis časa kreiranja nove vrstice (datum, ura, minute, sekunde),
P3: polje za zapis prijavnega imena uporabnika, ki je izvedel spremembo in
P4: polje za zapis časa spremembe (datum, ura, minute, sekunde).

Vzpostavili smo **dnevnik brisanj**, torej tabelo, v katero so se vpisovali podatki o pobrisanih zapisih. Beležili smo le nekaj bistvenih atributov brisane vrstice, ne pa vseh. Skupaj z glavnim uporabnikom smo opredelili, da je dovolj, če v namen sledenja brisanju beležimo ime tabele, od koder se je brisal zapis, vrednost ključa brisanega zapisa, šifro objekta, na katerega se je nanašal zapis, in še vrednosti dveh atributov. Tako vzpostavljen dnevnik brisanj ne omogoča rekonstrukcije prejšnjega stanja, je le pomoč pri sledenju

preteklega dogajanja na podatkih. Prožilec BI je tako napolnil polji P1 in P2, prožilec BU pa polji P3 in P4. Prožilec AD je napolnil dnevnik brisanj.

Kot smo se dogovorili v koraku načrtovanja, smo za šifrante, ki so že obstajali v registru prostorskih enot, vzpostavili samo poglede na te šifrante in tabel nismo podvajali. Po kreiranju tabel v relacijski podatkovni bazi sem z orodjem graditelj aplikacij in čarovniki, ki jih ponuja, oblikovala še zaslonske obrazce, povezane v aplikacijo. Najprej sem izdelala preprostejše zaslonske obrazce, to so obrazci za vzdrževanje vsebine tabel z matičnimi podatki. Z glavnim uporabnikom sva skupaj pregledala narejeno in ko mi je potrdil ustreznost, sem izdelala še transakcijski del aplikacije, to je podporo vzdrževanju podatkov o konkretnih nastanitvenih zmogljivostih. Po potrditvi ustreznosti so prišla na vrsto poročila in izpisi podatkov.

Izdelani program EHONAS je tako kot vsi APEX izdelki spletna aplikacija. Oblikovanje podobe spletnih aplikacij zahteva od razvijalca določene spretnosti in znanja. Teh znanj nimamo vsi razvijalci. Kadar naročnik nima posebnih zahtev glede videza aplikacije, si lahko pomagamo kar z eno od standardnih predlog, ki jih vsebuje APEX. Tudi sama se z videzom programa nisem posebej ukvarjala, skupaj z glavnim uporabnikom sva izbrala eno od že narejenih predlog. Glede na to, da ne gre za programsko rešitev, ki bi morala s svojo podobo privabiti čim več potencialnih kupcev in nam z ustvarjenim prometom prinesti čim večji zaslužek, ampak gre za program, ki ga uporabljamo zaposleni v organizaciji po službeni dolžnosti, sva presodila, da je uporaba ene od standardnih predlog dovolj dobra rešitev. Ta odločitev je pomenila velik prihranek časa in tudi denarja, saj bi za lepo oblikovano lastno podobo programa najbrž da morali najeti strokovnjaka z ustreznimi znanji.

5.3.5 Uvedba v uporabo

Po zaključku razvoja baznih objektov in spletnih strani programske rešitve je bilo treba najprej napolniti tabele z začetnimi podatki. Glavni uporabnik je v preglednicah pripravil začetno vsebino šifrantov, ki sem jo napolnila v tabele v bazi.

Po potrditvi ustreznosti programa je glavni uporabnik organiziral usposabljanje uporabnikov novega programa. Vsebino dela in predpise na tem področju so uporabniki poznali, treba jim je bilo samo omogočiti seznanitev s programom in jih naučiti, kako z njim delati. Pripravili smo jim navodilo za uporabo programa in organizirali dvourno skupno delo v testnem okolju. Tak način je bil izvedljiv, ker uporabnikov ni bilo veliko in smo jih lahko vse hkrati sprejeli v eno računalniško učilnico.

Spletno nadpovezavo na prijavno masko programa smo objavili na spletni strani na internem omrežju. Na spletni strani smo objavili tudi kratka navodila in kontaktne podatke oseb, ki so uporabnikom na voljo za vsebinsko in tehnično pomoč pri delu. Po končanem

usposabljanju so se uporabniki vrnili v svoje delovno okolje in so lahko samostojno začeli z delom.

5.4 Trenutno stanje

5.4.1 Program v uporabi

Program, ki je trenutno v uporabi, omogoča vodenje evidence civilnih nastanitvenih objektov, namenjenih za nudenje podpore države gostiteljice. Uporabnikom je omogočen vpogled v seznam objektov, dodajanje novih objektov na seznam, vzdrževanje podatkov o posameznem objektu in vzdrževanje podatkov o storitvah v okolici objekta. Omogočeno je tudi vzdrževanje vsebine lastnih šifrantov in vpogled v tuje šifrante, na katere se navezujemo (Zupančič, 2010, str. 3).

5.4.2 Podatki

Osnovni podatki o civilnih nastanitvenih zmogljivostih so zbrani v tabeli OBJEKTI. Za vsak objekt iz tabele OBJEKTI vodimo v posebnih tabelah še podatke o priključkih za elektriko, telekomunikacijskih priključkih, vodnih priključkih, načinih varovanja in storitvah v bližini objekta. Tako imamo v bazi poleg tabele OBJEKTI še pet tabel s prometnimi podatki, ki so s podatki v tabeli OBJEKTI povezani preko tujih ključev.

V okviru našega sistema smo za potrebe evidentiranja podatkov na objektih vzpostavili šifrante za šifriranje:

1. vrst objektov,
2. namembnosti objektov,
3. vrst električnih priključkov,
4. vrst telekomunikacijskih priključkov,
5. vrst priključkov za vodo,
6. vrst varovanja objektov,
7. vrst storitev,
8. vrst vojaških enot.

Iz registra prostorskih enot uporabljamo šifrante občin, naselij, pošt, ulic in naslovov. Uporabljamo tudi šifrant uprav za obrambo. Sicer entitetnega tipa uprava za obrambo v naš informacijski sistem ne uvajamo prvi. Zaradi izjemne statičnosti in majhnosti (vsebuje manj kot deset zapisov) pa smo se odločili, da ga realiziramo kar lokalno. Specifično za ta šifrant je, da se dodajanje novih zapisov ne pričakuje.

Za namen ureditve pravic dostopa do podatkov smo vzpostavili lastne tabele s podatki o uporabniških prijavnih imenih in nanje vezanih pravicah, ki se sicer programsko realizirajo preko avtorizacijskih shem.

5.4.3 Uporabniški vmesnik

Uporabniški vmesnik je realiziran kot spletna stran z več zavihki, iz katere se s kliki na povezave odpirajo nove spletne strani. Vmesnik ima štiri zavihke:

1. evidenca objektov,
2. matični podatki,
3. zunanji šifranti in
4. administracija.

Po odločitvi glavnega uporabnika je vse delo organizirano okrog seznama objektov, ki predstavlja prvi zavihek. Uporabniku se tako po prijavi v program najprej prikaže spletna stran s seznamom objektov. Vsaka vrstica v seznamu predstavlja en objekt oziroma civilno nastanitveno zmogljivost. Če ima uporabnik dodeljene pravice, lahko doda nov objekt ali spreminja podatke o obstoječem.

Po kliku na ime objekta se uporabniku odpre spletna stran s podatki o tem objektu. Tu ima na voljo možnost izpisa podatkov o objektu v dokument, ki ga lahko shrani v obliki datoteke v pdf formatu ali pa si dokument izpiše na papir. V odvisnosti od dodeljenih pravic lahko dodaja ali spreminja podatke o objektu. Drugi zavihek, matični podatki, je namenjen vpogledu v šifrante in vzdrževanju njihove vsebine. Na zavihku zunanji šifranti si uporabniki lahko ogledajo vsebino šifrantov, ki jih povzemamo iz drugih informacijskih rešitev in vanje ne posegamo. Zadnji zavihek administracija je namenjen tehničnemu skrbniku programske rešitve. Tu ima možnost urejanja pravic uporabnikov in parametrov za pravilno delovanje avtorizacijskih shem.

5.4.4 Informacijska varnost

V naši organizaciji področje informacijske varnosti ureja Pravilnik o varovanju komunikacijskega in informacijskega sistema (Ministrstvo za obrambo RS, 2014). Po tem pravilniku informacijska varnost zajema določanje in uporabo ukrepov varovanja tajnih podatkov, ki se obravnavajo s pomočjo komunikacijskih, informacijskih in drugih elektronskih sistemov, pred naključno ali namerno izgubo tajnosti, celovitosti ali razpoložljivosti ter ukrepov za preprečevanje izgube celovitosti in razpoložljivosti samih sistemov. Pravilnik določa, da je strojno, programsko in komunikacijsko opremo komunikacijskega in informacijskega sistema (v nadaljevanju KIS) MO dovoljeno uporabljati le osebam, ki imajo za to ustrezno pooblastilo, ki ga je mogoče preveriti s

postopki identifikacije, avtentikacije in avtorizacije. Pri izgradnji naše aplikacije smo sledili določilom navedenega pravilnika in iz tega pravilnika izhajajočim navodilom.

V razdelku 3.9.1 smo opisali štiri možne načine dostopanja uporabnikov v APEX aplikacijo. Prvi način, to je dostop brez avtentikacije, v našem primeru ni bil primeren, saj moramo spremljati, kdo vnaša podatke in kdo ima do njih dostop. Ker je večina uporabnikov naše aplikacije že imela dodeljeno prijavno ime in geslo za dostop do podatkovne baze za opravljanje drugih svojih delovnih nalog, smo se odločili za bazno avtentikacijo. To pomeni, da se uporabniki v program prijavljajo kar z baznim prijavnim imenom in se identificirajo z geslom, dodeljenim za bazo podatkov. Za večino uporabnikov zato ni bilo treba kreirati in dodeljevati novih prijavnih imen in gesel posebej za APEX. Smo jim pa morali dodeliti pravice za delo v aplikaciji.

Vse spletne strani programske rešitve EHONAS imajo omejen dostop. To pomeni, da je za vstop obvezna avtentikacija in dodeljene morajo biti ustrezne pravice. Spletnim stranem in njihovim elementom (gumbom, povezavam, zavihkom,...) so pridružene avtorizacijske sheme. Avtorizacijska shema je v našem primeru logična funkcija, katere izračunani rezultat je odgovor »da« ali »ne«. Na osnovi zapisov v tabelah pravic uporabnikov avtorizacijska shema odloči, ali ima konkreten uporabnik pravico za dostop ali je nima. Naloga razvijalca programa je bila, da premisli in pametno postavi tabele, v katerih vodi podatke o pravicah posameznih uporabnikov.

5.5 Ustreznost trenutne rešitve

Trenutno delujoča rešitev je po besedah uporabnikov nekaj let dobro služila namenu, za katerega je bila izdelana. Prav tako so uporabniki pohvalili enostavnost uporabe. Pri uporabi programa nismo opažali težav v tehničnem smislu, kar je tudi pozitivno. Res pa je, da sistem ni doživljal velikih obremenitev, količina podatkov ni bila velika in tudi uporabnikov ni veliko.

Sedanje potrebe po informatizaciji načrtovanja civilnih zmogljivosti pa so bistveno večje od tega, kar trenutno delujoča rešitev ponuja. Zato želi predstavnik naročnika čimprejšnjo razširitev oziroma prenovo obstoječega sistema v skladu s potrebami, opredeljenimi v Načrtu zagotavljanja podpore države gostiteljice, ki ga je Vlada sprejela v letu 2015 in smo ga na kratko povzeli v poglavju 4.

6 ANALIZA ŽELENEGA STANJA INFORMACIJSKE PODPORE

6.1 Potrebe naročnika

6.1.1 Popis potreb v obliki zgodb

Opisali bomo trenutno znane potrebe naročnika sistema glede delovanja novega informacijskega sistema za potrebe vodenja civilnih zmogljivosti za podporo gostovanju vojaških misij. Trenutno znane potrebe naročnika smo pridobili z začetno analizo zahtev, ki smo jo izvedli s pogovori z glavnim uporabnikom in s preučitvijo razpoložljive dokumentacije. Predvidevamo, da se bodo v prihodnosti zahteve spreminjale in da se bodo pojavljale nove. Zato pričakujemo, da se bo tudi informacijski sistem v prihodnje še spreminjal in širil. Tu opisana slika želenega stanja je tako le odraz trenutno znanih potreb naročnika in po vsej verjetnosti ne pomeni končnega stanja sistema.

Iz pogovorov z glavnim uporabnikom in na osnovi pregleda razpoložljive dokumentacije, predvsem Načrta zagotavljanja podore države gostiteljice v Republiki Sloveniji, smo razbrali sedem zgodb, ki jih je treba realizirati z novim informacijskim sistemom.

6.1.2 Vključitev uporabnikov iz drugih organizacij

Sedanji sistem je nameščen v internem omrežju naše organizacije. Nov sistem pa naj bi uporabljali tudi novi uporabniki, ki dostopa do našega internega omrežja nimajo. To so uporabniki v organizacijah nosilcih podpore, torej v organizacijah, ki so posebnega pomena za obrambo.

Uporabniki novega sistema bodo svoje delo opravljali v dveh fizično ločenih omrežjih, zato bo treba vzpostaviti bazo podatkov informacijskega sistema v vsakem od omrežij in poskrbeti za usklajenost podatkov v obeh bazah.

6.1.3 Spletna stran s seznamom predpisov in dokumentov

Vzpostaviti je treba spletno stran, na kateri bo objavljen seznam slovenskih predpisov in dokumentov, ki določajo postopke zagotavljanja podpore države gostiteljice. Prav tako je treba vzpostaviti spletno stran s seznamom NATOvih dokumentov za obravnavano področje. Omogočiti in izvajati je treba vzdrževanje obeh seznamov. Vpogled v oba seznama naj bo omogočen vsem uporabnikom brez preverjanja dodeljenih pravic dostopa.

6.1.4 Spletna stran z obrazci

Vzpostaviti je treba spletno stran za objavo obrazcev za zagotavljanje podpore države gostiteljice. Omogočiti in izvajati je treba vzdrževanje vsebine spletne strani določenim

uporabnikom. Vsem uporabnikom pa je treba omogočiti vpogled v obrazce in shranjevanje obrazcev s spletne strani na njihov računalnik.

Vzorci obrazcev bodo objavljeni z namenom, da si jih uporabniki shranijo in jih potem lokalno lahko izpolnijo in odpošljejo. Med samo analizo zahtev je glavni uporabnik zagotovil, da je dovolj, če bomo v okviru nove programske rešitve omogočili le objavo obrazcev. Ni treba zagotavljati možnosti spletnega izpolnjevanja obrazcev in avtomatske oddaje tako izpolnjenih obrazcev pravim naslovnikom. Zato se bomo zaenkrat omejili na to funkcionalnost. Z morebitno povezavo z dokumentnim informacijskim sistemom organizacije se bomo ukvarjali kasneje, če se bo ta zahteva pojavila.

6.1.5 Spletna stran z vzorci dokumentov

Vzpostaviti je treba spletno stran za objavo vzorcev dokumentov v procesu zagotavljanja podpore države gostiteljice. Spletno stran naj vidijo vsi uporabniki. Omogočiti in izvajati je treba vzdrževanje vsebine spletne strani določenim uporabnikom. Vsem uporabnikom pa je treba omogočiti vpogled v vzorce dokumentov in shranjevanje vzorcev s spletne strani na njihov računalnik.

Vzorci bodo objavljeni z namenom, da uporabnikom olajšajo sestavljanje lastnih dokumentov v procesu načrtovanja in izvajanja podpore države gostiteljice. Tako kot velja za objavo obrazcev, se tudi tu zaenkrat ne bomo ukvarjali s povezavo z dokumentnim informacijskim sistemom organizacije. Zavedamo pa se, da se takšna zahteva zna kdaj v prihodnje pojaviti.

6.1.6 Vodenje podatkov o osebah za stike

Organizacije nosilci podpore bodo v novem informacijskem sistemu objavili kontaktne podatke za svoje osebe za stike za namen organizacije načrtovanja in izvajanja podpore. Sistem naj omogoča objavo in vzdrževanje teh podatkov in navezavo teh podatkov na podatke o objektih zmogljivosti.

6.1.7 Vodenje podatkov o zmogljivostih različnih tipov

Trenutna funkcionalnost programa EHONAS omogoča vodenje podatkov o namestitvenih zmogljivostih in infrastrukturi, vezani na te zmogljivosti. Zmogljivosti se vodijo po upravah za obrambo (vsaka uprava za obrambo vzdržuje podatke o objektih, ki jih je evidentirala).

Nova funkcionalnost mora omogočati vodenje podatkov o zmogljivostih različnih tipov (letališča, luka…). Načeloma bo vsak tip zmogljivosti imel svoj nabor atributov. Podatkov ne bodo vzdrževali samo uporabniki v internem omrežju ministrstva, ampak tudi drugi

skrbniki podatkov, zaposleni v drugih ministrstvih in gospodarskih družbah posebnega pomena za obrambo.

6.1.8 Prenos podatkov v informacijski sistem zavezniške organizacije

Vzpostaviti je treba avtomatiziran prenos podatkov iz baze podatkov EHONAS v informacijski sistem LOGFAS, ki ga uporablja Slovenska vojska. Razvoj sistema LOGFAS poteka po naročilu Nata, zato nanj nimamo neposrednega vpliva. Način izvoza podatkov za prenos bo treba maksimalno prilagoditi tehničnim specifikam sistema LOGFAS.

Sistem LOGFAS se trenutno izdeluje na novo. Uvedba novega sistema LOGFAS v uporabo se pričakuje v kratkem. Odločili smo se, da bomo z realizacijo izvoza podatkov počakali do takrat, ko bodo znane specifikacije za uvoz podatkov v nov sistem LOGFAS. Vsebinsko pa moramo vseeno zagotoviti, da bo nov sistem EHONAS pripravljen za zbiranje vseh podatkov, ki jih bo treba izvažati za LOGFAS.

6.2 Uporabniki novega informacijskega sistema

Po do sedaj znanih zgodbah bo imel novi sistem v primerjavi s sedanjim več uporabnikov z več različnimi vlogami. Svoje delo bodo opravljali v svoji organizaciji, kjer bodo potrebovali dostop do novega sistema EHONAS. Zaenkrat lahko identificiramo:

1. Uporabnike v Slovenski vojski, ki bodo do sistema lahko dostopali tako kot do sedaj v internem omrežju ministrstva. Kot do sedaj bodo imeli dodeljene pravice za vpogled v podatke, ne bodo pa vnašali podatkov ali jih spreminjali. Za opravljanje njihovega dela jim bomo dodelili ustrezne pravice v programu APEX.
2. Uporabnike v ministrstvu za obrambo, ki bodo do sistema lahko dostopali tako kot do sedaj v internem omrežju ministrstva. Nekateri bodo skrbeli za matične podatke, torej za vsebino šifrantov. Nekateri bodo skrbeli za podatke o zmogljivostih. Za opravljanje njihovega dela jim bomo dodelili ustrezne pravice v programu APEX.
3. Uporabnike na drugih ministrstvih in v organizacijah posebnega pomena za obrambo, ki nimajo dostopa do internega omrežja ministrstva za obrambo. Zanje bo delo treba omogočiti v omrežju, do katerega imajo dostop.
4. Bodoči sistem bo imel tudi uporabnike brez posebej dodeljenih pravic za uporabo programa. V praksi to pomeni, da bomo del programa, torej nekaj njegovih spletnih strani, izdelali in dali v uporabo na tak način, da za njihov ogled ne bo potrebna posebna prijava v program. Strani bodo na vpogled vsem, ki imajo dostop do omrežja, v katerem bo deloval novi program.

6.3 Podatki in informacije

Iz povedanih zgodb lahko razberemo, da bo v novem programu potrebno voditi vsaj naslednje podatke:

1. podatki o organizacijah, ki so nosilci podpore,
2. podatki o zmogljivostih različnih tipov,
3. podatki o osebah za stike,
4. podatki za seznam slovenskih predpisov in dokumentov obravnavanega področja,
5. podatki za seznam NATO predpisov in dokumentov obravnavanega področja,
6. shranjevati bo treba obrazce,
7. shranjevati bo treba vzorce dokumentov.

Poleg naštetih podatkov bo treba zagotoviti tudi šifrante za šifriranje atributov pri vnosu podatkov o zmogljivostih. Nekateri šifranti bodo ostali isti kot do sedaj, ker pa bomo vodili podatke o zmogljivostih različnih tipov, o katerih bo treba voditi različne atribute, pa pričakujemo še pojav novih šifrantov. Pred informatizacijo vodenja podatkov o zmogljivostih bo treba narediti podrobno analizo zahtev, ki bo te šifrante natančno identificirala.

Iz zbranih podatkov je treba uporabnikom pripravljati naslednje informacije:

1. seznam slovenskih predpisov za obravnavano področje,
2. seznam NATO predpisov za obravnavano področje,
3. prikaz podatkov o zmogljivostih po različnih kriterijih,
4. izvoz podatkov o zmogljivostih v XML formatu za prenos v sistem LOGFAS,
5. seznam obrazcev s priklicem posameznega obrazca in z možnostjo lokalnega shranjevanja obrazca in
6. seznam vzorcev dokumentov s priklicem posameznega vzorca dokumenta in z možnostjo lokalnega shranjevanja vzorca dokumenta.

6.4 Informacijska varnost novega sistema

Za namen dobro urejene kontrole dostopa do programa in podatkov in za sledenje posegov v podatke bo tako kot v programu, ki je že v uporabi, treba vzpostaviti tabele za vodenje podatkov o dodeljenih uporabniških baznih prijavnih imenih in nanje vezanih pravicah. Ustrezno bo treba sprogramirati nove avtorizacijske sheme.

Dodatno bo potrebno preučiti vsa vprašanja varovanja informacijskega sistema, ki se na novo pojavljajo ob širitvi sistema v drugo fizično ločeno omrežje in ob izvozu podatkov v zavezniški informacijski sistem. Gre za vprašanja, povezana z odpiranjem internega

informacijskega sistema navzven in za prenos podatkov med posameznimi omrežji. Tu nas Pravilnik o varovanju KIS MO zavezuje k določenim postopkom in k upoštevanju določenih varnostnih zahtev (Ministrstvo za obrambo, 2014).

7 IZVEDBENI NAČRT ZA DOSEGO ŽELENEGA STANJA

7.1 Postopna izgradnja

Iz popisa trenutno znanih zahtev za novo informacijsko rešitev je razvidno, da vse funkcionalnosti novega sistema ta hip še niso znane. Iz preteklih izkušenj sklepamo, da se bodo tekom razvoja tudi trenutno znane zahteve še spreminjale. Zato menimo, da klasični slapovni pristop za to nalogo ni primeren.

Izgradnje novega informacijskega sistema se nameravamo lotiti postopoma. Pri tem bomo upoštevali prioritete, ki jih je postavil glavni uporabnik. Sistem bomo gradili iterativno, pri delu bomo upoštevali izbrana agilna načela. Razvojno orodje bo tako kot do sedaj orodje Oracle APEX. Trudili se bomo sistem razvijati z lastnimi viri, po potrebi pa bomo v razvojno ekipo vključevali zunanje strokovnjake za posamezne tehnične izzive. Delo bomo zunanjim izvajalcem oddajali po pravilih naročanja storitev, ki veljajo za neposredne proračunske uporabnike. V ta namen smo tudi predlagali vključitev stroškov dela zunanjega strokovnjaka v proračun ministrstva za prihodnjih nekaj let.

7.2 Način dela

Kot vodilo pri delu smo izbrali agilna načela, ki smo jih preučili v Poglavju 1. Trudili se bomo za zadovoljstvo naročnika, sprejemali zahteve po spremembah v vseh korakih razvoja, rezultate razvoja bomo dajali v uporabo kakor hitro je le mogoče, razvijalci in uporabniki bodoče rešitve bomo pri razvoju čim tesneje sodelovali, svoj napredek bomo merili s količino delujoče programske opreme, stremeli bomo k enostavnosti izdelanih rešitev, delali bomo v zmernem tempu, ki ga je možno dalj časa vzdrževati, redno bomo preverjali svoj način dela in ga po potrebi prilagodili.

Prvo verzijo programskega proizvoda smo razvijali po klasični slapovni metodi. Zaradi majhnosti projekta smo ocenili, da vodenje obsežne projektne dokumentacije ni potrebno. Rezultat je bil, da smo delali brez nekih čvrstih okvirov. Res da smo se uporabe novega orodja šele učili in smo bili pri delu zato počasnejši, a dostikrat so bile zamude tudi rezultat zahajanja z glavne smeri v samostojno izdelovanje lepe programske kode, ki pa morda tisti hip ni bila nujno potrebna. Pri izgradnji novega sistema pričakujemo od sebe več discipline. Sicer še vedno ne želimo voditi obsežne projektne dokumentacije, ki bi nas pri delu upočasnjevala, želimo pa neka vodila, ki bi nas disciplinirala in preprečevala zahajanje na stranpoti ter nas tako hitreje peljala do cilja.

S tem namenom smo si glede na našo dejansko situacijo iz nabora agilnih pristopov, opisanih v prvem poglavju, izbrali naslednje prakse in tehnike dela in tako zgradili svoj lasten osnutek metodologije agilnega razvoja:

1. zgodbe: glavni uporabnik je svojo vizijo novega informacijskega sistema podal v obliki zgodb in iz teh zgodb bomo izhajali pri podrobni analizi zahtev in pri načrtovanju realizacije,
2. najprej bomo razvili izdelke, ki jim uporabnik pripisuje najvišjo prioriteto,
3. celovita ekipa: po potrebi bomo v ekipo vključevali poznavalce vsebine in procesov in tehnične strokovnjake za določena področja,
4. informativno delovno okolje: pripravili bomo stensko tablo, iz katere bo razviden napredek projekta,
5. delajmo polni energije: delo bomo skrbno načrtovali in ga opravljali v okviru rednega delovnega časa, ne bomo podaljševali v nadurno delo,
6. štirinajstdnevni cikli – šprinti,
7. ohlapnost pri načrtovanju: puščali si bomo nekaj rezerve pri načrtovanju dela,
8. stalna integracija: APEX že sam po sebi omogoča, da izdelke zelo hitro dajemo na voljo uporabnikom, saj ni lokalnega razvoja, vse razvijamo na strežniku, vsi razvijalci na istem projektu imajo takojšen vpogled v vse izdelke,
9. stalna priprava testov: hkrati z razvojem izdelkov bomo pripravljali testne primere, s katerimi bomo izdelke čim prej preizkusili,
10. čakalni seznam produkta: vodili bomo seznam vsega, kar je potrebno realizirati,
11. čakalni seznam šprinta je seznam vsega, kar imamo namen realizirati v štirinajstih dneh in
12. stremljenje k preprostim rešitvam.

Med agilnimi praksami in tehnikami, ki jih ocenjujemo kot koristne in bi jih želeli izbrati, a jih zaradi različnih omejitev nismo izbrali, so tudi:

1. sedenje skupaj: za takšen način dela moramo imeti na voljo dovolj velik in udoben prostor, v katerem člani ekipe preživijo večino delovnega dne; trenutno takšnega prostora ni na voljo, hkrati pa smo tudi člani ekipe vsi angažirani še na drugih nalogah in bomo lahko našemu projektu posvetili le del svojega časa, kar sicer ni najboljše, je pa realnost,
2. programiranje v paru: večinoma bom na nalogi delala kot edina razvijalka, zato za delo v paru ne bo veliko priložnosti; poskusila ga bom uvesti takrat, ko bomo v ekipo vključili še kakšnega tehničnega strokovnjaka za reševanje specifičnih problemov in
3. avtomatsko testiranje: v naši organizaciji nimamo uveljavljenega načina avtomatskega testiranja programerskih izdelkov, zato ga tudi pri našem projektu ne bomo uporabili, bomo pa sproti izvajali teste z ročnim vnosom podatkov in proženjem delov programa.

7.3 Dokumentacija

Pri vsakem projektu se pojavi vprašanje količine dokumentacije, ki bi jo bilo treba voditi. Pri razvoju prve verzije programa smo vodili le malo projektne dokumentacije, večinoma smo zapisali dogovore, ki smo jih sprejeli na delovnih srečanjih in si jih sodelujoči poslali po elektronski pošti. Pomembnejše dogovore smo člani ekipe poslali v vednost tudi drugim deležnikom. Tak način obveščanja je vsem zadoščal in nam hkrati ni jemal veliko časa. Ocenili smo, da je sistem dovolj dober in ne potrebuje sprememb. Če se bo pojavila potreba, pa bomo način komunikacije na projektu spremenili.

V primeru vključevanja dela zunanjega strokovnjaka bo potrebno v dokumentacijo izvajanja projekta vložiti več truda, saj mora biti iz nje natančno razvidno, kaj je delal zunanji strokovnjak, kako obsežno je bilo njegovo delo, merjeno v številu ur, in koliko sredstev smo v ta namen porabili. Skrbeti je treba za vso dokumentacijo, ki je predpisana v primerih, ko porabljamo proračunska sredstva z oddajo del zunanjemu izvajalcu.

Pri samem razvoju prve verzije programske rešitve smo bazne objekte kreirali neposredno v bazi z uporabo APEXovega modula SQL delavnica. Ker tabel ni bilo veliko, predhodno nismo posebej risali modela entitet-povezav, zadoščal nam je seznam tabel. Za tako majhen sistem sva imela tako sistemski analitik kot razvijalka za sabo dovolj praktičnih izkušenj, da sva lahko učinkovito komunicirala kar s seznami tabel in atributov, povezave so nama bile očitne. Nov sistem bo po naših predvidevanjih večji. Sproti se bomo odločali, katere modele je treba risati. Zagotovo pa bo osnova za pogovor o entitetah vsaj na papir skicirani E-R model.

7.4 Prva iteracija razvoja nove rešitve

V prvi iteraciji bomo informatizirali tri zgodbe, ki so za glavnega uporabnika ta hip najpomembnejše: objavo seznama predpisov in dokumentov, objavo obrazcev in objavo vzorcev dokumentov. Pripravili bomo prostor za shranjevanje obrazcev in dokumentov v bazi podatkov in izdelali spletne strani za uporabnike. Poskrbeli bomo, da bodo novo rešitev lahko uporabljali uporabniki v internem omrežju organizacije kot tudi uporabniki, ki nimajo dostopa do internega omrežja.

Za realizacijo teh treh zgodb smo identificirali naslednje naloge:

1. V internem omrežju je treba vzpostaviti razvojno okolje za novo programsko rešitev. Potrebujemo bazno shemo in delovni prostor v APEXu.
2. Zgradili bomo spletno stran za objavo seznama slovenskih predpisov in dokumentov in jo postavili v novo okolje, da jo bodo uporabniki v internem omrežju lahko takoj začeli uporabljati. Z uporabo čarovnika bo spletna stran izdelana hitro. Nekaj uporabnikov bo

imelo pravice za urejanje seznama, vsi ostali pa le pravice za vpogled. Da bi to omogočili, moramo že v tem koraku izdelati tudi tabele za urejanje pravic uporabnikov in vsaj dve avtorizacijski shemi. Način realizacije tabel za urejanje pravic uporabnikov in način izdelave avtorizacijskih shem bomo povzeli po prvi verziji programske rešitve, a si bomo prizadevali za njegovo poenostavitev, kar bo prispevalo k večji preglednosti.

3. Na enak način bomo izdelali še spletno stran za objavo seznama NATOvih predpisov in dokumentov. Spletni strani bosta v aplikacijo povezani z navigacijo z zavihki in drobtinicami. Z uporabo čarovnika to ne bo predstavljalo veliko dela. Uredili bomo pravice za dostop v internem omrežju.
4. Izdelali bomo spletno stran za objavo obrazcev. Z uporabo čarovnika bomo spletno stran izdelali zelo hitro, pričakujemo pa, da bo nekaj dodatnega dela z dodajanjem omejitev na velikost datotek, ki jih bodo uporabniki lahko objavljali na spletni strani. Uredili bomo pravice za dostop v internem omrežju.
5. Na enak način kot za objavo obrazcev bomo izdelali še spletno stran za objavo vzorcev dokumentov in uredili dostop.
6. Omogočili bomo dostop do izdelane programske rešitve uporabnikom, ki nimajo dostopa do internega omrežja naše oganizacije. Vsi ti uporabniki pa imajo dostop do omrežja krizno upravljanje. Skupaj s sistemskim administratorjem bomo poiskali način, kako bomo bazne objekte in aplikacijo prenesli še v to omrežje in kako bomo zagotovili usklajeno stanje podatkov v obeh omrežjih. Ker sta omrežji fizično ločeni, bo treba prenos podatkov izvajati z uporabo izmenljivih nosilcev podatkov. Ta izziv ima dva vidika, tehnični vidik in organizacijski. V tehničnem smislu bo treba pripraviti procedure za izvoz podatkov v datoteko na izmenljivi nosilec in za uvoz podatkov iz datoteke na izmenljivem nosilcu v bazo v drugem omrežju. Uporabili bomo orodje XML DB. V organizacijskem smislu bo treba izdelati zanesljiv protokol prenosa podatkov iz enega v drugo omrežje. To sicer terja nekaj premisleka in modeliranja tokov podatkov, a v primeru, ko se bo za vsak podatek vedelo, v katerem od obeh omrežij se ga sme dodajati ali spreminjati in je v drugem omrežju na voljo le za prikaz, ne pričakujemo nepremostljivih težav.

7.5 Druga iteracija

V drugi iteraciji bomo realizirali zgodbo o vodenju podatkov o zmogljivostih različnih tipov in vodenju podatkov o osebah za stike. Brez poglobljene analize lahko ugotovimo, da bomo tu informatizirali delo z več povezanimi entitetami. Ne bo šlo le za izdelavo spletnih strani z nekaj vsebine, ampak bomo gradili pravo aplikacijo nad podatkovno strukturo, čeprav v obliki spletnih strani. Zato predvidevamo, da bo potrebno opraviti naslednje naloge:

1. Podrobna analiza zahtev in izdelava E-R modela in prevedba tega v tabele v relacijski bazi. Tabele bomo v bazi izdelali z uporabo čarovnikov, ki jih ponuja APEX.

2. Ko bodo tabele vzpostavljene in povezane, bomo z uporabo čarovnikov za izdelavo spletnih strani izdelali spletne strani za vzdrževanje podatkov v teh tabelah.
3. Z uporabo tabel za urejanje pravic uporabnikov, ki smo jih izdelali v prvi iteraciji, bomo uredili pravice za dostop uporabnikov. Izdelali bomo dodatne avtorizacijske sheme.
4. Procedure za prenos podatkov iz enega v drugo omrežje, ki smo jih izdelali v prvi iteraciji, bomo dopolnili tako, da se bo ustrezno prenašala tudi vsebina novih tabel. Tu pričakujemo, da bo potreben temeljit razmislek in risanje modela tokov podatkov. Kompleksnost rešitve bo odvisna od tega, ali bo takrat še vedno veljala predpostavka, da se bo podatke vnašalo le v enem omrežju, v drugem omrežju pa bodo imeli uporabniki vanje le vpogled.
5. Ko bo uporabnik potrdil ustreznost izdelane rešitve in jo bomo iz preizkusne faze vpeljali v redno uporabo, bomo lahko upokojili prvo verzijo programa EHONAS.

7.6 Tretja iteracija

V tretji iteraciji bomo realizirali še prenos podatkov v informacijski sistem zavezniške organizacije.

1. Pridobiti bo treba natančne vsebinske in tehnične specifikacije podatkov, ki jih bo treba izvažati za zavezniški program. Pričakujemo, da si bomo do takrat priskrbeli definicijo strukture XML datoteke z vsebinskimi opisi.
2. Preverili bomo, kako bomo iz pri nas zbranih podatkov o zmogljivostih lahko pridobili podatke za izvoz. Po potrebi bomo morali razširiti obstoječi podatkovni model in popraviti spletne strani za vnos podatkov.
3. Z uporabo orodja XML DB bomo izdelali procedure za izvoz podatkov v ustrezno datoteko.

Informacijska rešitev, ki jo moramo izdelati, ni velika v smislu količine podatkov, števila uporabnikov ali števila uporabniških ekranov. Po teh kriterijih je pravzaprav zelo majhna rešitev. Prav zaradi svoje majhnosti pa je zelo primerna za preizkus agilnega pristopa k razvoju.

Kljub majhnosti pa izdelava te rešitve prinaša kar nekaj tehničnih izzivov. Realizirati moramo informacijski sistem, ki ima bazo podatkov distribuirano v dveh fizično ločenih omrežjih in kjer obstaja velika možnost, da se bo vsebina tabel s podatki ažurirala v obeh omrežjih in bo zato treba podatke prenašati v obeh smereh. Prav tako moramo sistem prilagoditi za izvoz podatkov v tuj sistem, na razvoj katerega zaenkrat nimamo kaj dosti vpliva.

Izdelava takšnega informacijskega sistema s klasičnimi razvojnimi orodji za imperativno programiranje je kljub majhnosti zamudno in zahtevno opravilo. Napisati bi bilo treba

veliko programske kode. Z uporabo čarovnikov, ki nam jih ponuja APEX, pa bomo rešitev izdelali bistveno hitreje in z manj možnosti napak. Prav tako predvidevamo, da nam bo uporaba Oraclove rešitve XML DB zelo olajšala prenose podatkov med sistemi.

Pri reševanju tehničnih vprašanj si bomo lahko pomagali z bogato dokumentacijo, ki je na spletu na voljo uporabnikom teh orodij in jo proizvajalec stalno dopolnjuje. Prav tako si bomo lahko pomagali z iskanjem odgovorov na naša vprašanja na spletnih forumih, kjer sodelujejo razvijalci z vsega sveta. Tak način dela je bistveno drugačen in zagotovo lažji od načina dela, ki smo ga bili razvijalci navajeni pred desetletji.

8 EKONOMIČNOST PROJEKTA IZGRADNJE NOVEGA SISTEMA

8.1 Prirastni stroški

Pripravili smo načrt za izvedbo prenove informacijskega sistema. Ob tem pa je treba preučiti tudi smiselnost izvedbe te prenove: koliko stroškov bo z izvedbo projekta in koliko koristi bomo od projekta imeli. Zanima nas, ali se izvedba tega projekta splača. Sprašujemo se, če so koristi res večje od stroškov. Pri oceni ekonomske upravičenosti projekta si pomagamo s prispevčno analizo, opredeljeno v razdelku 1.7.2.

Stroške pri razvoju novega informacijskega sistema je težavno oceniti iz več razlogov. V obravnavanem primeru bodo neposredni in v denarnih enotah izraženi stroški zaradi razvoja novega sistema nastali le takrat, ko bomo najeli zunanjega izvajalca in ga vključili v delo projektne skupine za specifična tehnološka opravila. Ker ga bomo plačevali po urah, lahko ta strošek izračunamo tako, da ocenimo število ur, ki jih bo zunanji izvajalec opravil za naš projekt. Vendar pa je vnaprej to število ur težko realno oceniti, če gre za delo, ki še ni utečeno. Ceno ure lahko dokaj dobro ocenimo na osnovi preteklih dogovorjenih pogodbenih cen ure za podobna dela.

Pretekla praksa kaže, da smo dostikrat ves denar, ki smo ga načrtovali za izgradnjo kakšnega informacijskega sistema, tudi v resnici porabili in večkrat nam ga je še zmanjkalo. Tako da si za oceno stroškov zunanjega izvajalca lahko pomagamo kar zneskom, ki ga imamo v proračunu na voljo za projekt. Več kot imamo, zagotovo ne bomo zapravili. To je sicer pesimistična ocena stroškov, a najbrž da kar blizu realnosti.

Na projektu bomo delali tudi domači razvijalci, drug tehnični kader in uporabniki, tako člani projektne skupine kot tudi tisti, ki bodo nov sistem samo uporabljali. Zaradi tega projekta bomo vsi našteti imeli dodatno delo. Ena možnost za oceno stroškov projekta bi bila, da ocenimo tudi število delovnih ur, ki jih bodo posamezni deležniki vložili, jih pomnožimo z njihovimi bruto urnimi postavkami in zneske prištejemo k stroškom projekta. Zagotovo bi dobili visok znesek.

Vendar se na strošek delovne sile v državni upravi običajno ne gleda na ta način. Običajno se privzame, da so plače zaposlenih fiksen strošek, ki bremeni proračun v vsakem primeru. S stališča proračuna je javne uslužbence vedno treba plačati. Zato stroška lastne delovne sile ne pripisujemo stroškom projekta. V ozadju takšnega razmišljanja je predpostavka, da so zaposleni v državni upravi dovolj neobremenjeni s svojim delom, da lahko brez škode za svoje siceršnje naloge prevzamejo še sodelovanje v eni ali več projektnih skupinah. Takšno razmišljanje je morda v posameznih primerih upravičeno, dostikrat pa se nam tudi maščuje. Zgodi se lahko, da pri načrtovanju projekta preveč računamo na proste notranje vire, ki so »zastonj«, potem pa je projekt neuspešen ravno zaradi tega, ker so člani projektnih skupin angažirani na preveč nalogah in niso sposobni dovolj intenzivno sodelovati z drago plačanimi zunanjimi izvajalci.

V praksi stroškovno primernost projekta dostikrat ocenjujemo kar na način, da ocenimo, ali lahko načrtujemo dovolj denarja za dejanska izplačila, ki jih bomo imeli zaradi zunanjih izvajalcev, in če imamo na voljo zaposlenega s primernimi znanji, ki bo na projektu lahko delal. Kadar je na voljo dovolj denarja, je njegovo delo to, da sodeluje z zunanjimi izvajalci, ko pa denarja zmanjka, pa običajno pričakujemo, da bo delo na razvoju nadaljeval sam, seveda s precej manjšim tempom.

Pri presoji smiselnosti izvedbe projekta informatizacije in njegovih stroškov je treba razmisliti tudi o morebitnih stroških, ki bi nastali zaradi potreb po dodatnem prostoru v podatkovni bazi, zaradi potreb po nakupu dodatne strojne opreme, bodisi osebnih računalnikov, komunikacijske opreme ali celo novega strežnika. V našem primeru smo zaenkrat ocenili, da ti dodatni stroški ne bodo nastali. V bazi podatkov bo nekaj novih tabel, a količina podatkov v njih ne bo velika. Bazno shemo bomo sicer na novo postavili še v drugo omrežje, a tudi v tem omrežju je na baznem strežniku še dovolj prostora, da s tako majhno količino podatkov ne bo problema. Za fizičen prenos podatkov med obema omrežjema si bomo morali priskrbeti kakovosten izmenljivi nosilec podatkov. Ta strošek nikakor ne bi smel zanašati več kot 100 EUR, saj toliko stanejo dražji zunanji diski. Povsem možno pa je, da bo za naše prenose podatkov dovolj dober kar USB izmenljivi ključek, ki je bistveno cenejši.

8.2 Koristi in prirastni prihodki

Prepričani smo, da bo uspešno zaključen projekt prinesel velike koristi vsem deležnikom. Koristi vidimo predvsem v naslednjem:

1. Izboljšala se bo obveščenost sodelujočih v procesu zagotavljanja podpore države gostiteljice.
2. Naročniku bo nov sistem zelo olajšal zbiranje podatkov iz drugih ministrstev in organizacij posebnega pomena za obrambo.

3. Prav tako bo sistem, vsaj ko bo delo utečeno, olajšal pripravo podatkov uporabnikom, ki morajo podatke zagotavljati.
4. Olajšano bo delo vsem, ki izpolnjujejo obrazce in pripravljajo dokumente, saj bodo na spletnih straneh imeli na voljo pripravljene predloge za te obrazce in dokumente.
5. Olajšano bo delo pri poročanju v zavezniški informacijski sistem, saj podatkov ne bo treba več pretipkavati, ampak jih bomo avtomatsko prenašali.
6. Z avtomatiziranimi prenosi podatkov se bo precej zmanjšala možnost nastanka napak pri prenosih podatkov iz enega v drug sistem.
7. Z razvojem distribuiranega sistema v APEX okolju v dveh fizično ločenih omrežjih se bomo tehnični delavci zagotovo veliko naučili. Pridobljena znanja nam bodo v korist pri načrtovanju in izvedbi bodočih v tehnološkem smislu podobnih projektov. Morda zaradi teh novih znanj v prihodnje ne bomo več potrebovali sodelovanja zunanjih izvajalcev in bomo prihranili nekaj neposrednih stroškov.
8. Nenazadnje, z oddajo dela zunanjemu izvajalcu bomo v imenu davkoplačevalcev nekaj denarja vrnili gospodarstvu.

Tako utemeljene koristi in velika pripravljenost glavnega uporabnika za delo na projektu so bile vodilo, da smo projekt uvrstili na seznam nalog za izvedbo. Z glavnim uporabnikom smo dobro sodelovali že v prvi iteraciji izgradnje programa. Vedno je imel konkretne odgovore na vprašanja. Ko je bilo potrebno, si je vzel čas za sodelovanje kljub drugim nalogam, v svojih zahtevah je bil zmeren v smislu, da so bile vedno v okvirih izvedljivega.

Resnično težko pa bi tu navedene koristi realno ovrednotili v konkretnih denarnih zneskih. Z veliko špekuliranja lahko naredimo le neko grobo oceno prirastnih prihodkov. Če koristi, ocenjene za prihodnja leta, diskontiramo z obrestno mero in upoštevamo le njihovo neto sedanjo vrednost, dobimo manjšo oceno prirastnih prihodkov od te, ki jo bomo prikazali v nadaljevanju. A pri trenutnih zelo nizkih obrestnih merah ne naredimo velike napake, če namesto neto sedanje vrednosti upoštevamo kar nediskontirane zneske. Napaka, ki smo jo naredili zaradi zelo grobih ocen posameznih postavk stroškov in prihodkov, je bistveno večja od napake zaradi opustitve diskontiranja.

Koristi izboljšav delovnega procesa zaradi novega sistema lahko ovrednotimo v denarju tako, da ocenimo število prihranjenih delovnih ur. Če je pri starem načinu dela uporabnik potreboval 4 ure, da je sestavil kompleksen dokument, z uporabo informacijskega sistema pa to delo opravi v eni uri, je prihranek pri izdelavi vsakega takega dokumenta 3 ure. S podobno grobo oceno lahko ovrednotimo tudi nekatere druge koristi.

Tabela 1: Ocena vrednosti prirastnih stroškov in prihodkov

Vrsta stroška ali prihodka	Ocena vrednosti stroška v EUR	Ocena vrednosti prihodka v EUR na leto
Sodelovanje zunanjega strokovnjaka v obsegu največ 100 delovnih ur s ceno ure približno 50 EUR	5.000	
Nakup dodatne strojne opreme (samo izmenljivi nosilec podatkov)	100	
Nakup dodatne programske opreme ni potreben	0	
Sodelovanje domačih strokovnjakov pri razvoju in uvedbi (100 ur po ceni 20 EUR)	2.000	
Sodelovanje uporabnikov pri razvoju in uvedbi (100 ur po ceni 20 EUR)	2.000	
Boljša obveščenost		Ni ocene
Lažje zbiranje podatkov (prihranek 20 delovnih ur po 20 EUR)		400
Lažja priprava podatkov (prihranek 20 delovnih ur po 20 EUR)		400
Lažje izpolnjevanje obrazcev in priprava dokumentov (100 dokumentov na leto, pri vsakem prihranek 3 ure po 20 EUR)		6.000
Avtomatiziran prenos podatkov v zavezniški sistem		Ni ocene
Manjša možnost napak pri prenosu podatkov		Ni ocene
Pridobljena nova znanja tehničnih delavcev (100 delovnih ur po 20 EUR namesto po 50 EUR, z verjetnostjo 50%)		1.500
SKUPAJ	9.100	8.300

Pridobljena nova znanja lastnih tehničnih delavcev bodo pomenila prihranek oziroma prirastni prihodek za našo organizacijo le v primeru, ko bodo naši strokovnjaki na

prihodnjih nalogah dejansko opravljali specifična tehnična opravila sami in za to delo ne bo več treba najemati zunanjih strokovnjakov. Prihranek bo v razliki cene ure zunanjega in lastnega strokovnjaka, ob predpostavki enake učinkovitosti obeh. Ta prihranek bo nastal le v primeru, ko bomo v prihodnje res sami delali na projektih s podobnimi tehničnimi izzivi, sicer prihranka ne bo. Vsaj pri oceni bodočih prihrankov zaradi pridobljenih novih znanj lastnih tehničnih delavcev moramo zato upoštevati tudi negotovost izvedbe morebitnih prihodnjih projektov, v katerih bi to znanje organizaciji lahko prineslo dejanske prihranke. Za namen te analize predpostavimo, da je verjetnost koristne uporabe teh znanj 50%.

Iz navedenega lahko ocenimo, da bodo celotni stroški izgradnje in uvedbe sistema znašali 9.100 EUR, pričakovana vrednost koristi novega sistema pa bo 9.800 EUR letno. Opisana kalkulacija je prikazana Tabeli 1. Za koristi, ki jih ne moremo izraziti v številu delovnih ur, ni podana ocena. To so boljša obveščenost, avtomatiziran prenos podatkov in manjša možnost napak.

8.3 Smiselnost izvedbe projekta

V primeru, ko bi znali realno izraziti v številkah stroške in bodoče prihodke zaradi izvedbe projekta, bi lahko izračunali njegovo **stopnjo donosnosti** (angl. *return on investment,* v nadaljevanju ROI), to je kvocient med razliko med koristmi in stroški in stroški:

$$ROI = (Celotne\ prihodki - Celotni\ stroški) / Celotni\ stroški. \quad (1)$$

Če upoštevamo ocenjene prirastne prihodke novega sistema v obdobju treh let, bi tako lahko po enačbi (1) izračunali ROI v višini (3 x 8.300 EUR – 9.100 EUR)/ 9.100 EUR = 174 %.

S tako zbranimi ocenami lahko izračunamo tudi **dobo povračila investicije**, ki nam pove, v kolikšnem času se nam povrnejo stroški projekta:

$$Doba\ povračila = Znesek\ investicije / Znesek\ koristi\ na\ letni\ ravni \quad (2)$$

Kot je razvidno iz enačbe (2), je v našem primeru torej doba povračila enaka 9.100 / 8.300 = 1,096. Po tem izračunu bi se nam investicija povrnila po dobrem letu dni uporabe novega sistema.

Ali se razvoj izplača ali ne, je v strogo finančnem smislu težko odgovoriti, saj so ocene zneskov zelo približne in je morda zato izračunana donosnost nerealno visoka. Verjamemo pa, da ga je smiselno izvesti, ker:

1. glavni uporabnik v njem prepozna veliko koristi in je tudi pripravljen tvorno sodelovati,
2. imamo na voljo svoj tehnični kader za sodelovanje na projektu,

3. je projekt majhen in ima zaradi majhnosti veliko možnosti, da uspe in
4. nudi veliko možnosti za učenje o uvedbi novih tehnologij v naše okolje.

SKLEP

V tem delu smo na primeru prenove in razširitve manjšega informacijskega sistema pokazali, kako lahko z uporabo sodobnih in hkrati zrelih orodij hitreje in lažje razvijemo zahtevano informacijsko rešitev. V uvodu smo si zastavili sedem ciljev, ki smo jih tudi realizirali:

Pregledali smo različne metodologije razvoja informacijskih sistemov in si po preučitvi najpogostejših agilnih pristopov sestavili svoj lastni agilni pristop, prilagojen na konkretno situacijo v organizaciji, v okviru katere bo potekalo delo na prenovi informacijskega sistema. S tem smo realizirali prvi zastavljeni cilj. Pregledali smo značilnosti orodja Oracle APEX, s katerim bomo izvedli prenovo in ugotovili, da je primerno za uporabo agilnega razvojnega pristopa. S tem smo dosegli drugi cilj.

Preučili smo proces načrtovanja in podpore gostovanju vojaških misij, kot ga določa Načrt zagotavljanja podpore države gostiteljice in tako realizirali tretji cilj. Analizirali smo trenutno stanje informacijske rešitve in pregledali smo zahteve za njeno razširitev ter tako realizirali četrti in peti cilj.

Izdelali smo načrt za realizacijo prenove z upoštevanjem izbranega agilnega pristopa in z uporabo orodij Oracle APEX in XML DB. Z uporabo pristopa prispevčne analize smo izvedli poskus ovrednotenja nastalih prirastnih stroškov in prihodkov zaradi razvoja in uvedbe novega sistema. Izračunali smo možno donosnost projekta in ugotovili, da je projekt smiselno izvesti. Tako smo realizirali šesti zastavljeni cilj.

Po preučitvi literature in na osnovi lastnih izkušenj smo pokazali, da uporaba sodobnih orodij, kot sta Oracle APEX in XML DB, pa tudi internetnih tehnologij, ki so do zdaj že dosegle stopnjo zrelih tehnologij, bistveno olajša delo razvijalcem informacijskih rešitev v primerjavi z načinom dela v preteklosti. Tako smo realizirali sedmi zastavljeni cilj.

Dosegli smo v uvodu zastavljeni namen naloge, saj smo izvedli analizo trenutnega stanja informacijske podpore obravnavanega področja in pripravili načrt za njeno prenovo, v skladu s pričakovanji naročnika in tehnološkimi, varnostnimi in finančnimi omejitvami. Prav tako smo prikazali, da zrelost tehnologije, razvojnih orodij in metodologij, ki so danes na voljo razvijalcem, omogoča hitrejši in lažji razvoj informacijskih rešitev.

V delu smo prikazali veljavnost obeh v uvodu postavljenih hipotez. Ugotovili smo, da je prenovo informacijskega sistema za podporo gostovanju vojaških misij možno

ekonomično izvesti z razvojnim orodjem Oracle APEX in z lastno agilno metodologijo. Prav tako smo pokazali, da trenutna zrelost tehnologije, razvojnih orodij in metodologij omogoča hitrejši in lažji razvoj zanesljivih spletnih programskih rešitev v primerjavi z načinom dela v preteklosti.

V nalogi je opisan en od možnih pristopov k razvoju majhne spletne aplikativne rešitve s podatki v relacijski bazi. Opisani primer je lahko v pomoč razvijalcem, ki se bodo pri svojem delu srečevali s podobnimi izzivi.

Pričakujemo, da se bo obravnavani informacijski sistem tudi po izvedbi trenutno znanih nalog še razvijal. Predvidevamo lahko, da bo kmalu treba uporabnikom omogočiti izpolnjevanje obrazcev in njihovo oddajo v elektronski obliki. Kako bomo to realizirali, ali bomo naš sistem povezali z dokumentnim sistemom ali kako drugače, bomo razmišljali takrat, ko bo do te zahteve prišlo, če bo prišlo.

Pri pripravi tega dela sem med študijem prišla do spoznanj, za katera menim, da so koristna tudi za organizacijo, v kateri delujem. Menim, da bi se v prihodnje morali v naši organizaciji v večji meri posvetiti sistematičnemu uvajanju agilnih razvojnih pristopov v delo službe, pristojne za informatiko. Prav tako bi morali večjo pozornost nameniti skrbi za vsakoletno posodabljanje strateškega načrta informatike v povezavi s strateškim načrtom organizacije. Menim tudi, da bi bilo koristno preučiti ponudbo orodij za avtomatsko testiranje, med njimi izbrati primerno orodje za naše delovno okolje in izbrano orodje sistematično uvesti v uporabo.

Na v nalogi obravnavanem projektu sem sodelovala kot razvijalka in sistemska analitičarka in tudi kot tehnični vodja projekta. Zato je moj pogled na prenovo informacijskega sistema morda bolj tehnološki kot pa vsebinski. Trenutne potrebe uporabnikov poznam, njihove prihodnje potrebe lahko predvidim, a moja predvidevanja so lahko tudi popolnoma napačna. Zato je prav, da dajejo pobudo za izbiro smeri razvoja v prihodnosti uporabniki oziroma lastnik procesa. Informatiki pa moramo spremljati razvoj tehnologij in jih čim bolje poznati, da jih lahko pametno izberemo in uporabimo pri iskanju konkretnih rešitev za izkazane potrebe.

LITERATURA IN VIRI

1. *Agile Alliance*. Najdeno 11. aprila 2016 na spletnem naslovu https://www.agilealliance.org.
2. *Agile Manifesto*. Najdeno 11. aprila 2016 na spletnem naslovu agilemanifesto.org.
3. *Agile process*. Najdeno 20. aprila 2016 na spletnem naslovu http://www.agile-process.org/ byfeature.html.
4. Bajec, M., & Krisper, M. (2003). Agilne metodologije razvoja informacijskih sistemov. *Uporabna informatika, 11*(2), str. 68-76. Ljubljana: Slovensko društvo Informatika.
5. Beck, K., & Andres, C. (2004). *Extreme programming explained: Embrace Change* (2nd ed.). Boston: Addison-Wesley Professional.
6. Berdugo, M. (2014). *Software development - in house vs outsourcing*. Najdeno 11. aprila 2016 na spletnem naslovu https://www.linkedin.com/pulse/20140724152738-2996742-software-development-in-house-vs-outsourcing.
7. Cimolini, P., & Cannell K. (2012). *Agile Oracle Application Express (Expert's Voice in Oracle)*. New York: Apress. E-knjiga.
8. Dogša Tomaž, (1993). *Verifikacija in validacija programske opreme*. Maribor: Tehniška fakulteta.
9. Grad, J., & Jaklič, J. (1996). *Baze podatkov*. Ljubljana: Ekonomska fakulteta.
10. Gradišar, M., Jaklič, J., Talib, D., & Baloh, P. (2005). *Osnove poslovne informatike*. Ljubljana: Ekonomska fakulteta.
11. Gradišar, M., Jaklič, J., & Turk, T. (2012). *Osnove poslovne informatike*. Ljubljana: Ekonomska fakulteta.
12. *Internetworldstats*. Najdeno 20. julija 2016 na spletnem naslovu www. internetworldstats .com.
13. *Introduction to XML*. Najdeno 11. aprila 2016 na spletnem naslovu http://www. w3schools.com/xml/ xml_whatis.asp.
14. ISO. (2008). *Standard ISO/IEC 12207*. Systems and software engineering – Software life cycle processes. Najdeno 15. avgusta 2016 na spletnem naslovu http://www.iso.org/iso/ catalogue_detail?Csnumber=43447.
15. Jaklič, J. (2002). *Upravljanje in uporaba podatkov*. Ljubljana: Ekonomska fakulteta.
16. Jerman-Blažič, B., Klobučar, T., Perše, Z., & Nedeljković, D. (2001). *Elektronsko poslovanje na internetu*. Ljubljana: Ekonomska fakulteta.
17. Koratamaddi, C. (2015). *Application Express 5.0 Overview*. Redwood Shores: Oracle. Najdeno 23. maja 2016 na spletnem naslovu http://www.oracle.com/ technetwork/developer-tools/apex/learnmore/apex-50-overview-2526922.pdf.
18. Korelič, I. (2011). Agilno pri poslovnem obveščanju. *Monitor Pro: nove tehnologije za poslovni svet, 11*(3), Ljubljana: Mladina, 26-27.
19. Kovačič, A., Groznik, A., & Ribič, M. (2009). *Temelji elektronskega poslovanja*. Ljubljana: Ekonomska fakulteta.
20. Kovačič, A., Jaklič, J. , Indihar Štemberger, M., & Groznik, A. (2004). *Prenova in informatizacija poslovanja*. Ljubljana: Ekonomska fakulteta.

21. Laudon, K.C., & Traver, C. G. (2012). *E-commerce 2012*. New Jersey: Prentice Hall.
22. Mandya, A. & Mani, R. (2013). *Oracle Application Express Workshop II*. Student Guide. Redwood Shores: Oracle.
23. Metoda. (b.l.). V *Slovarju slovenskega knjižnega jezika*. Najdeno 30. avgusta 2016 na spletnem naslovu http://bos.zrc-sazu.si/cgi/a03.exe?name=sskj_testa&expression= metoda&hs=1.
24. Metodologija. (b.l.). V *Slovarju slovenskega knjižnega jezika*. Najdeno 30. avgusta 2016 na spletnem naslovu http://bos.zrc-sazu.si/cgi/a03.exe?name=sskj_testa&expression= metodologija&hs=1.
25. Ministrstvo za obrambo Republike Slovenije. (2014, 5. junija). *Pravilnik o varovanju komunikacijskega in informacijskega sistema Ministrstva za obrambo*. Ljubljana: Ministrstvo za obrambo, 2014.
26. Model. (b.l.). V *iSlovarju*. Najdeno 20. julija 2016 na spletnem naslovu www.islovar. org/islovar.
27. Nonaka, I. (1991). The Knowledge-Creating Company. *Harvard Business Review, 69*(6), str. 96-104.
28. Oracle. (b.l. a). *Apex Builder User's Guide: Managing Application Security*. Najdeno 20. avgusta 2016 na spletnem naslovu https://docs.oracle.com/cd/E14373_01/appdev.32 /e11838/sec.htm#HTMDB12000.
29. Oracle. (b.l. b). *Oracle Application Express Learn More.* Najdeno 1. avgusta 2016 na spletnem naslovu http://www.ilo.org/dyn/jobcrisis/f?p=4600:28:1665614926112843 ::NO:::.
30. Oracle. (b.l. c). *Oracle Database XE Overview*. Najdeno 20. junija 2016 na spletnem naslovu http://www.oracle.com/technetwork/database/database-technologies/express edition/overview/index.html.
31. Oracle. (b.l. d). *Oracle SQL Developer*. Najdeno 20. junija 2016 na spletnem naslovu http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/ index.html.
32. Oracle. (b.l. e) *SQL Developer Downloads*. Najdeno 26. septembra 2016 na spletnem naslovu http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/ index.html.
33. Oracle. (b.l. f). *Oracle SQL Developer Licence Terms*. Najdeno 20. septembra 2016 na spletnem naslovu http://www.oracle.com/technetwork/licenses/sqldev-license-152021.html.
34. Oracle. (2010). *Oracle Exadata Overview*. Najdeno 20. avgusta 2016 na spletnem naslovu http://www.slideshare.net/amekawy/exadata-overview-audio.
35. Pipenbaher, B. (2006). *Podpora države gostiteljice (Host nation support - HNS)*. Ljubljana: Ministrstvo za obrambo Republike Slovenije, Direktorat za obrambne zadeve, Sektor za civilno obrambo.
36. Rational. (1998). *Rational unified process: Best Practices for Software Development Teams*. Cupertino: Rational Software.

37. Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2012). *Introduction to systems analysis and design: an agile, iterative approach* (6th ed.). Australia (etc.): Course Technology, Cengage Learning.
38. Shelly, G. B., Cashman, T. J. , & Rosenblatt, H. J. (2008). *Systems analysis and design*. Australia: Thomson Course Technology.
39. Serhal, L. K., Srivastava, T., Koratamaddi, C., & Clement, S. (2011a). *Oracle Database 11g: Use XML DB*. *Volume I. Student Guide.* (Gradivo za tečaj.) Oracle.
40. Serhal, L. K., Srivastava, T., Koratamaddi, C., & Clement, S. (2011b). *Oracle Database 11g: Use XML DB*. *Volume II. Student Guide.* (Gradivo za tečaj.) Oracle.
41. Software testing times. (2010). *V-model is a basis of structured testing*. Najdeno 30. avgusta 2016 na spletnem naslovu http://www.softwaretestingtimes.com/2010/04/v-model-is-basis-of-structured-testing.html.
42. SQL. (b.l.) V *iSlovarju*. Najdeno 20. junija 2016 na spletnem naslovu www.islovar.org/islovar.
43. Sutherland, J. (2016). *V gruču do uspeha – Umetnost vodenja projektov z metodo SCRUM*. Ljubljana: Založba Pasadena.
44. Tajnikar, M., Brščič, B., Bukvič, V., & Ponikvar, N. (2004). *Upravljavska ekonomika z vajami.* Ljubljana: Ekonomska fakulteta.
45. TCP/IP. (b.l.) V *MaFiRa-Wiki*. Najdeno 20. septembra 2016 na spletnem naslovu http://wiki.fmf.uni-lj.si/wiki/TCP/IP.
46. Turban, E. (2012). *Electronic commerce 2012: a managerial and social networks perspective*. Boston: Pearson.
47. Turban, E., & King, D. (2003). *Introduction to e-commerce*. Upper Saddle River (NJ): Prentice Hall.
48. Tutorialspoint. (b.l.) SDLC: V-model. Najdeno 20. julija 2016 na spletnem naslovu http://www. tutorialspoint.com/sdlc/sdlc_v_model.htm.
49. Vlada Republike Slovenije. (2015, 17. junij). *Načrt zagotavljanja podpore države gostiteljice v Republiki Sloveniji*. Ljubljana: Vlada Republike Slovenije, 2015.
50. Vlada Republike Slovenije, Center za informatiko. (2000). *EMRIS – Enotna metodologija razvoja informacijskih sistemov*. Najdeno 16. septembra 2016 na spletnem naslovu http://www2.gov.si/mju/emris.nsf.
51. Zupančič, M. (2010). *Programski proizvod EHONAS: Evidenca civilnih nastanitvenih zmogljivosti za nudenje podpore države gostiteljice (R3). Navodilo za uporabo programa.* Interno gradivo. Ljubljana: Ministrstvo za obrambo.
52. Žurga, G. (2015). *Kakovost državne uprave*. Ljubljana: Fakulteta za družbene vede.

PRILOGA

PRILOGA 1: SEZNAM KRATIC IN OKRAJŠAV

Kratica	Pomen
APEX	Oracle Application Express
AD	angl. *after delete*
ASD	prilagodljiv razvoj programske opreme (angl. *adaptive software development*)
AUP	agilno združen proces (angl. *agile unified process*)
BI	angl. *before insert*
BU	angl. *before update*
B2B	podjetje –podjetje (angl. *business to business*)
B2B2C	poslovanje podjetij z drugimi podjetji, ki poslujejo s končnimi potrošniki (angl. *business to business to consumer)*
B2C	podjetje – potrošnik (angl. *business to consumer*)
B2E	podjetje – zaposleni (angl. *business to employee*)
B2G	podjetje – država (angl. *business to government*)
CASE	računalniško podprto načrtovanje programske opreme (angl. *computer-aided software engineering)*
C2B	potrošnik – podjetje (angl. *consumer to business*)
C2C	med potrošniki (angl. *consumer to consumer*)
C2G	potrošnik – država (angl. *consumer to government*)
DDL	jezik za definiranje podatkov (angl. *data definition language*)
DFD	diagram toka podatkov (angl. *data flow diagram)*
DML	jezik za manipulacijo podatkov (angl. *data manipulation language)*
DSDM	metoda dinamičnega razvoja sistemov (angl. *dynamic system development method*)
EC	elektronsko poslovanje (angl. *electronic commerce)*
EDI	računalniška izmenjava podatkov (angl. *electronic data interchange*)
EMRIS	enotna metodologija razvoja informacijskih sistemov
E-R diagram	diagram entitet povezav (angl. *entity relationship diagram)*
E2E	povezovanje javnih elektronskih trgov (angl. *exchange to exchange*)
FDD	razvoj narekovan z lastnostmi (angl. *feature-driven development*)
FTP	Protokol za prenos datotek (angl. *file transfer protocol*)
G2G	znotraj javne oziroma državne uprave (angl. *government to government*)
HNS	podpora države gostiteljice (angl. *host nation support*)
HTML	jezik za označevanje nadbesedila (angl. *hyper text markup language*)
HTTP	protokol za prenos nadbesedila (angl. *hyper-text transfer protocol)*
IP	internetni protokol (angl. *Internet Protocol*)

se nadaljuje

nadaljevanje

Kratica	**Pomen**
ISO	mednarodna organizacija za standardizacijo (angl. *International Organization for Standardization)*
JAD	skupni razvoj aplikacij (angl. *joined application development)*
KIS	komunikacijski in informacijski sistem
LDAP	internetni protokol za dostop do imenikov na osnovi arhitekture odjemalec strežnik (angl. *lightweight directory access protocol*)
LOGFAS	Informacijski sistem logistike (angl. *Logistics Functional Area Services*)
MO	Ministrstvo za obrambo
NATO	Organizacija severnoatlantskega sporazuma (angl. *North Atlantic Treaty Organisation*)
ODBC	odprta povezljivost z zbirkami podatkov (angl. *open database conectivity*)
OTN	Oraclovo tehnološko omrežje (angl. *Oracle Technology Network*)
PHP	razširjen odprtokodni programski jezik za razvoj dinamičnih spletnih vsebin (prvotno v angl. *personal homepage tools*)
PL	proceduralni jezik (angl. *procedure language*)
RAD	hitri razvoj aplikacij (angl. *rapid application development)*
ROI	donosnost naložbe (angl. *return on investment*)
RS	Republika Slovenija
RUP	angl. *Rational Unified Process*
SA&D	sistemska analiza in načrtovanje (angl. *systems analysis & design)*
SDLC	življenjski cikel razvoja sistema (angl. *system development life cycle)*
SQL	Strukturirani poizvedovalni jezik (angl. *structured query language*)
SUPB	sistem za upravljanje s podatkovno bazo (angl. *database management system)*
TCO	skupni stroški lastništva (angl. *total cost of ownership)*
TCP	protokol nadzora nad prenosom (angl. *Transmission Control Protocol*)
TCP/IP	standardiziran protokolni sklad za komunikacijo v heterogenem okolju
UML	poenoteni jezik modeliranja (angl. *unified modelling language)*
URL	enolični krajevnik vira (angl. *uniform resource locator*)
USB	univerzalno serijsko vodilo (angl. *Universal Serial Bus*)
XML	razširljivi označevalni jezik (angl. *extensible markup language)*
XP	ekstremno programiranje (angl. *extreme programming)*
WebDAV	angl. *Web Distributing Authoring and Versioning*
WWW	svetovni splet (angl. *world wide web*)
W3C	WWW konzorcij (angl. *WWW consortium*)