**UNIVERZA V LJUBLJANI**

**EKONOMSKA FAKULTETA**

# MAGISTRSKO DELO

## NAČRTOVANJE PRENOVE INFORMACIJSKE PODPORE IN UVAJANJE PREDMETNE INFORMACIJSKE TEHNOLOGIJE V NEPROFITNO DELOVNO OKOLJE

**Ljubljana, maj 2008** **BRANKO GLAVAN**

**IZJAVA**

Študent Branko Glavan izjavljam, da sem avtor tega magistrskega dela, ki sem ga napisal pod mentorstvom prof. dr. Andreja Kovačiča in skladno s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne 5. 5. 2008 Podpis ……

# Vsebina

# Tabele



# Slike

# Seznam uporabljenih okrajšav

| | |
|---|---|
| ANSI | American National Standards Institute (zasebna neprofitna organizacija za razvoj standardov v ZDA) |
| ASD | Adaptive Software development (agilna metodologija razvoja RR) |
| AUP | Agile Unified Process (agilna metodologija razvoja RR) |
| BPMI | Business Process Management Initiative (neprofitna organizacija za standardizacijo splošnih poslovnih procesov pod okriljem OMG) |
| BPMN | Business process modeling notation (grafična notacija poslovnih procesov) |
| BPR | Business Process Renovation (metoda prenove poslovnih procesov) |
| BRA | Rule-based Business Renovation approach (metoda prenove poslovnih procesov) |
| BSP | Business System Planning (metoda ugotavljanja informacijskih potreb) |
| BUP | Basic Unified Process ( metodologija razvoja RR) |
| COM | Component object model (Microsoftova tehnologija za komunikacijo med objekti) |
| CORBA | Common object request broker architecture (OMG standard za vzajemno delovanje programskih komponent) |
| CORE | Controlled Requirements Expression (metodologija razvoja RR) |
| CRC | Class-Responsibility-Collaboration (pripomoček v objektnem načrtovanju) |
| CSF | Critical Success Factors (metoda vzpostavljanja odločilnih dejavnikov) |
| DBMS | Data base management system (sistem za upravljanje podatkovnih baz) |
| DCOM | Distributed component object model (Microsoftova tehnologija za komunikacijo med komponentami) |
| DSDM | Dynamic Systems Development Method (agilna metodologija razvoja RR) |
| DSDM | Dynamic Systems Development Method (metodologija razvoja RR) |
| ERP | Enterprise Resource Planning (celovita uporaba RR v podjetju) |
| EssUP | Essential Unified Process (metodologija razvoja RR) |
| EUP | Enterprise Unified Process (metodologija razvoja RR) |
| FAQ | Frequently Asked Questions (način zbiranja in širjenja znanja s pomočjo interneta) |
| FDD | Feature Driven Development (agilna metodologija razvoja RR) |
| FMC | Fundamental modeling concepts (koncept opisovanja sistemov RR) |
| GGN | Gozdnogospodarski načrti |

| | |
|---|---|
| GGOJN | Gozdnogojitveni načrti |
| GIS | Geographic information system (geografski informacijski sistem) |
| GoF | Gang-of- four (avtorji načrtovalskih vzorcev v inženiringu RR) |
| HKOM | Privatno omrežje za prenos podatkov med organizacijami naše javne uprave |
| HTTP/HTTPS | Hypertext Transfer Protocol /Hypertext Transfer Protocol over Secure Socket Layer (standardna komunikacijska protokola med odjemalci in strežniki) |
| IEC | International Electrotechnical Commission (mednarodna nevladna neprofitna organizacija za objavo standardov s področja elektrotehnike) |
| IEEE | Institute of Electrical and Electronics Engineers (mednarodna neprofitna organizacija za promocijo električne in elektronske tehnologije) |
| IIS | Internet information services (skupina Microsoftovih spletnih storitev za strežnike z Windows operacijskim sistemom) |
| IKT | Informacijsko komunikacijska tehnologija |
| iODBC | Independent open database connectivity (standardne funkcije za dostopanje in uporabo podatkovnih baz namenjene odprtim sistemom) |
| ISAC | Information System Work and Analysis of change (metoda ugotavljanja informacijskih potreb) |
| ISIC | International Standard Industrial Classification (of All Economic Activities) (klasifikacijski sistem Združenih narodov) |
| ISO | International Organization for Standardization (mednarodno združenje za standardizacijo) |
| IT | Informacijska tehnologija |
| J2EE | Java Platform Enterprise Edition (Java platforma za razvoj RR na osnovi vzajemno delujočih komponent na aplikacijskih strežnikih) |
| JAD | Joint Application Development (metodologija razvoja RR) |
| JDBC | Java database connectivity (funkcije za dostopanje in uporabo podatkovnih baz za Java programsko okolje) |
| MDA | Model Driven Architecture (načrtovanje RR na osnovi standardiziranih modelov OMG) |
| MKGP | Ministrstvo za kmetijstvo gozdarstvo in prehrano |
| MySQL | Brezplačni sistem za upravljanje podatkovne baze |
| .NET | Microsoftova platforma za razvoj RR na osnovi vzajemno delujočih komponent |
| ODBC | Open database connectivity (standardne funkcije za dostopanje in uporabo podatkovnih baz neodvisno od računalniškega okolja) |
| OECD | Organisation for Economic Co-operation and Development (mednarodno združenje organizacij za gospodarsko sodelovanje in razvoj) |
| OGC | OpenGeospatialConsortium (konzorcij organizacij na področju standardizacije v GIS) |

| | |
|---|---|
| OLE | Object linking and embedding (Microsoftova tehnologija za vstavljanje in povezovanje objektov) |
| OLE-DB | Object linking and embedding for data base (Microsoftove funkcije za dostopanje in uporabo podatkovnih baz) |
| OLTP | Online Transaction Processing (sprotna obdelava transakcij) |
| OMG | Object Management Group (konzorcij za vzpostavljanje standardov porazdeljenih objektno zasnovanih sistemov) |
| OML | OPEN modeling language (jezik za modeliranje) |
| OMT | Object Modeling Technique (metodologija razvoja RR) |
| OOAD | Object-oriented analysis and design (oblika razvojnega inženiringa RR) |
| OOT | Objektno orientirana tehnologija |
| OPEN | Object-oriented Processs Environment Notation (metodologija razvoja RR) |
| OpenUP | Open Unified Process (metodologija razvoja RR) |
| OPF | OPEN process framework (metodološko ogrodje OPEN) |
| ORACLE | Sistem za upravljanje relacijske podatkovne baze firme Oracle Corp. |
| ORM | Object role modeling (modeliranje RR s pomočjo vlog) |
| OSI | Open systems interconnection (model za standizirano povezovanje računalniških omrežij) |
| PRINCE2 | Projects in Controlled Environments (metodologija vodenja projektov) |
| RFC | Request for comment (dogovorna oblika ustvarjanja novih idej za uporabo internetne tehnologije) |
| RPC | Remote Procedure Call (tehnologija za izvedbo procedur v oddaljenem naslovnem prostoru) |
| RR | Računalniške rešitve |
| RUP | Rational Unified process (metodologija razvoja RR) |
| RUP-SE | Rational Unified Process-System Engineering (metodologija razvoja RR) |
| SADT | Structured Analysis and design Technique (metodologija razvoja RR) |
| SASS | Structured Analysis and System Specification (metodologija razvoja RR) |
| SCR | Software Cost Reduction requirements method (metodologija razvoja RR) |
| SOA | Service Oriented Architeture (modelna arhitektura za uporabo poslovnih procesov) |
| SOAP | Simple Object Access Protocol (protokol za izmenjavo XML sporočil) |
| SSADM | Structured System Analysis and Design Technology (metodologija razvoja RR) |
| SSM | Soft System Methodology (metodologija razvoja RR) |
| SWOT | Strengths, Weaknesses, Opportunities and Threats (metoda strateškega planiranja) |

| | |
|---|---|
| SysML | Systems modeling language (modelirni jezik) |
| TAD | Tabular Application Development (metodologija razvoja RR) |
| TCP/IP | Transmission Control Protocol/Internet protocol (skupina internetnih komunikacijskih protokolov) |
| UDDI | Universal Description Discovery and Integration (register za objavljanje poslovnih storitev) |
| UP | Unified Process (agilna metodologija razvoja RR) |
| UML | Unified modeling language (poenoteni jezik modeliranja) |
| URL | Uniform Resource Locator (globalni identifikator dokumenta v svetovnem spletu) |
| VDM | Vienna Development model (metodologija razvoja RR) |
| VORD | Viewpoint-Oriented Requirements Definition (metodologija razvoja RR) |
| W3C/IETF | World Wide Web Consortium/Internet Engineering Task Force (mednarodna organizacija za standardizacijo za področje svetovnega spleta) |
| WCF | Windows Communication Foundation (Microsoftova razvojna platforma za vzajemno delovanje aplikacij del .NET platforme) |
| WMS | Web map service (OGC standard za dinamično predstavitev georefernciranih podatkov) |
| WSDL | Web Services Description Language (opisni jezik za opisovanje spletnih storitev na osnovi XML) |
| XML | Extensible Markup Language (specifikacija označevalnega jezika) |
| XP | Extreme programming (agilna metodologija razvoja RR) |
| ZGS | Zavod za gozdove Slovenije |
| ZOG | Zakon o gozdovih |

# 1. UVOD

## 1.1 Predstavitev problematike

Informacijsko komunikacijska tehnologija z integracijo informacijskih in telekomunikacijskih sistemov spreminja naš način potrošnje, razmišljanja in načina dela. Uvajanje sodobne komunikacijske in informacijske tehnologije za podporo poslovnim procesom spreminja način poslovanja podjetij, spodbuja rast novih storitev, obstoječim pa spreminja njihovo dosedanjo podobo. Podjetja iščejo konkurenčne prednosti, prihaja do sprememb poslovnih modelov in načinov poslovanja, znanih iz industrijskega obdobja.

Pomemben del sodobne družbe so neprofitne organizacije. Naloge storitev v javnem interesu so poverjene različnim organizacijam in izvajalcem. Izvajanje storitev na področju javnega interesa, tako kot pri ostalih dejavnostih, sloni na izvedbi poslovnih procesov in njihovi informatizaciji. Kulture profitnih in neprofitnih organizacij si postajajo čedalje bolj podobne. Poslovni sistemi na področju javnih storitev čedalje bolj stremijo k odličnosti. Prehod administrativnih oblik vodenja na podjetniške čedalje bolj sili neprofitne organizacije v prevzemanje uspešnih vzorcev upravljanja podjetij. Mednje gotovo sodi tudi uporaba strateškega načrtovanja, ki omogoča usklajevanje in prilagajanje organizacije na potrebe in izzive njenega okolja. Neprofitne organizacije ne prejemajo toliko signalov o uspešnosti in neuspešnosti svojega poslovanja iz okolja kot običajna podjetja v tržnem okolju (Možina et al. 2002, 706). Morda je tudi to eden od razlogov za počasnejše uvajanje novih menedžerskih pristopov, temelječih na strateškem načrtovanju, v poslovanje neprofitnih organizacij.

Poleg izzivov vodenja na spreminjanje delovanja neprofitnih organizacij, vplivajo tudi drugi dejavniki. Dinamičen razvoj informacijske tehnologije predstavlja enega takšnih dejavnikov. Za primer lahko vzamemo razmah uporabe interneta ter posledično povezovanje organizacij in javnosti ter uvajanje novih E-storitev. Povezani informacijski sistemi omogočajo večjo porazdeljenost informacij in znanja, vplivajo na spreminjanje informacijske podpore procesom v organizacijah, spreminjajo arhitekturo informacijskih sistemov. S podobnimi motivi, kot obravnavamo zbliževanje pristopov načinov upravljanja profitnih in neprofitnih organizacij, lahko preučujemo področje informatizacije profitnih organizacij in prenašamo uspešne razvojne vzorce iz enega v drugo okolje. Načrtovanje razvoja informatike podjetja je sestavni del poslovnega načrtovanja (Kovačič, Bosilj-Vukšić, 2005, 233). Razlikovanje med strateško in podporno funkcijo informatike je eno od osnovnih izhodišč za njeno nadaljnje načrtovanje. Na ravni nosilcev in izvajalcev uvajanja informacijske podpore se danes v podjetjih zahteva fleksibilnost, prilagodljivost, kreativnost, strateško razmišljanje, poznavanje tehnologije in poznavanje poslovanja. Od njih se pričakuje, da bodo sposobni upravljati spremembe, da bodo poznali in uvajali nove standarde. Vloga informatika v informacijski družbi zahteva od informatika interdisciplinarno znanje s področij upravljanja, poslovanja in uporabe informacijske tehnologije (Kovačič, Bosilj-Vukšić, 2005, 250). Tudi v neprofitnih organizacijah postaja čedalje bolj pomemben dejavnik uspešne informacijske podpore organiziranost službe za informatiko in znanje njenih kadrov.

V snovanju in načrtovanju razvoja informatike je treba načrtovati informacijske potrebe podjetja tudi z vidika potencialov, ki jih ponuja tehnologija. Na področju izdelave računalniških rešitev je mit o samostojnih aplikacijah in nedotakljivih podatkovnih modelih zdavnaj zbledel. Informacijska arhitektura čedalje bolj temelji na porazdeljenih računalniških sistemih.

V porazdeljenih bazah podatkov so podatki ene same baze ter programska oprema za upravljanje baz podatkov porazdeljeni po več računalnikih (Grad, Jaklič, 1996). Globalna povezanost strojne opreme, vzajemno delovanje aplikacij in integracija razpoložljivih podatkov postaja del svetovne gospodarske infrastrukture. Strategija razvoja programske opreme, temelječe na konceptu ločevanja podatkov in funkcionalnosti, je presežena. Razvojno paradigmo usmerja predmetna orientiranost. Računalniški programi postajajo skupki komponent, predmetov, ki jih lahko vedno znova uporabimo (Ambler, 1998b).

Razvoj, izdelava, integracija in vzdrževanje sestavin, ki jih omogoča predmetno orientiran koncept izdelave programske opreme, zahteva skrbno načrtovanje. Za potrebe načrtovanja imamo na voljo vrsto predmetno usmerjenih metodologij razvoja računalniških rešitev. Postopki razvoja in izdelave programske opreme so v domeni programskega inženirstva opisani v obliki procesov. Ti procesi so sprva vsebovali strogo formalizirane, kaskadne razvojne metode, danes pa so v uporabi bolj prožne, agilne metode. Eden od standardnih, lahko bi rekli referenčnih procesov razvoja programske opreme, je proces poenotenega razvoja programske opreme (Jacobson, Booch, Rumbaugh, 1999a). V okviru tega razvojnega procesa je uporabljena metoda za predmetno zasnovano zbiranje in analizo zahtev, ki bi jo moral vsaj okvirno poznati vsak udeleženec v razvoju programske opreme. Temelji na primerih uporabe, ki so dokaj razširjena oblika sistematičnega zbiranja in analize funkcionalnih zahtev sistema. Definiranje zahtev sistema je eden temeljnih procesov v njegovem razvoju. Z njim opredelimo splošni kontekst sistema, vsako posamično funkcijo in nalogo sistema ter želene rezultate sistema (Damij, 2000, 160-161). Predstavitev sistema sloni na različnih pogledih, ki jih lahko predstavimo z modeli. Danes večina analitikov in načrtovalcev na področju razvoja programske opreme uporablja splošni večnamenski grafični jezik kot sredstvo medsebojne komunikacije. Poenoteni jezik za modeliranje je privzet tudi kot standard. Modeliranje postaja čedalje bolj razširjeno kot tehnika za vzpostavljanje komunikacijskega kanala in boljše razumevanje obravnavane problematike med udeleženci v razvoju informacijskih sistemov. Na področju uvajanja informacijske podpore v podjetju nam modeli lahko služijo tako na ravni predstavitve poslovnih procesov kot na ravni predstavitve informacijske arhitekture. Poslovni vidik modeliranja in informacijski vidik modeliranja se morata na neki točki soočiti in vzpostaviti medsebojno povezavo. Na ta način lahko pričakujemo ustreznost informacijske podpore poslovnim potrebam (Kovačič, Bosilj-Vukšić, 2005).

Predmetno zasnovana izdelava računalniških rešitev se vzajemno navezuje tudi na arhitekturo informacijskih sistemov. Poseben konzorcij, Object Management Group, skrbi za razvoj standardov na področju predmetne zasnove programov. Za konceptualno razumevanje med udeleženci konzorcija so bili vzpostavljeni različni referenčni modeli kot abstraktni okvir za razumevanje relacij med entitetami v obravnavani domeni. Poznavanje referenčnih modelov je lahko koristen pripomoček tudi pri razvoju aplikacij, saj lahko usmerja celoten koncept naše načrtovane informacijske podpore. Eden od standardov, ki že dlje časa usmerja zasnovo programske opreme v izdelavo porazdeljenih objektov oziroma programskih komponent, se imenuje posrednik objektnih zahtev in poizvedb (CORBA). Referenčni model, ki ga ne bi smeli spregledati ob modeliranju, povezovanju in pretvarjanju modelov v večje ali manjše komponente, je storitvena arhitektura (Service Oriented Architecture, SOA). Na poti razvoja računalniških rešitev, ki temeljijo na komponentah, se srečamo tudi z načrtovalskimi vzorci. Ti omogočajo zbiranje znanja in vzpostavljanje besednjaka, kar je še zlasti pomembno za dialog med uporabniki, programerji in vodji projektov. Načrtovalski vzorec je splošna ponovljiva rešitev za izvedbo informacijske podpore pri ponavljajočih zahtevah (Gamma et al. 1995, str. 6-7). Vsi navedeni dejavniki že vsak zase predstavljajo razvojni potencial s široko paleto možnosti konkretne uporabe na primerih v vsakdanjem življenju. V snovanju in izvedbi podpore informacijskim procesom v organizaciji je potrebno ta razvojni potencial usmerjati med

udeležence razvoja. Od stopnje njihovega znanja in razumevanja informacijske tehnologije ter od uspešnosti vzajemne medsebojne komunikacije je v veliki meri odvisna uspešnost končne informacijske podpore uporabnikom. Za obvladovanje sprememb pri uvajanju novih tehnologij je nujno potrebno zagotavljati dotok novega znanja na različne ravni upravljanja in izvedbe dela v podjetjih. Raziskava na osnovi študijskih primerov slovenskih podjetij o uspešnosti in uporabi informacijsko komunikacijskih tehnologij je pokazala, da so najpomembnejši prepoznani dejavniki uspešnosti projektov uporabe informacijsko komunikacijske tehnologije vezani na načrtovanje, pogled menedžmenta na uvajanje te tehnologije, vlogo informacijske tehnologije v strateških razvojnih programih, oddelke za informatiko, hkratno uvajanje organizacijskih sprememb, usposobljenost zaposlenih in dodatno izobraževanje, skrb za motiviranost in komunikacijo ter dobro izbiro partnerjev in odnosi z njimi (Stare, Bučar, 2005).

Lahko zaključimo, da dinamika sprememb v razvoju informacijsko komunikacijske tehnologije vpliva na načine našega dosedanjega razmišljanja, delovanja in organiziranosti. To velja tako za področje profitnih kot neprofitnih organizacij. Tam, kjer spremljanje tehnološkega razvoja zaradi različnih vzrokov zaostaja, se lahko poslužujemo posnemanja uspešnejših vzorcev delovanja. Med večje ovire uspešnejšim projektom razvoja informacijske podpore lahko štejemo pomanjkanje znanja in komunikacijske ovire, ki se pojavljajo na trikotniku vodenja organizacije, zagotavljanje informacijske podpore in uporabe informacijskih rešitev. Širjenje znanja, celovita obravnava procesov, ki se dogajajo znotraj tega trikotnika in poznavanje pripomočkov uspešnejše komunikacije med udeleženci procesov lahko znatno pospešijo uvajanje novih tehnologij, tudi v neprofitnem delovnem okolju.

## 1.2. Cilj in namen magistrskega dela

Vsaka služba oziroma organizacijski oddelek, ki izvaja informacijsko podporo v izbranem delovnem okolju, se danes na nek način prilagaja spremembam na področju informacijske tehnologije. Ključni dejavnik uspešnega uvajanja in obvladovanja sprememb postaja znanje, organizacijsko in tehnološko. To velja tudi za organizacije, ki imajo neprofitni značaj svojega delovanja. V neprofitnih organizacijah je vpliv tekmovalnosti trga manj izrazit, zato se spremembe dogajajo bolj poredko in s počasnejšim tempom sledijo tehnološkim spremembam. Odsotnost tekmovalnosti je verjeten razlog, da spremembe informacijske podpore niso strateško načrtovane. Pa vendar tudi neprofitne organizacije niso imune na vplive novih tehnoloških sprememb.

Sprožilci, ki spodbujajo preobrazbo informacijske podpore, so v neprofitnih okoljih lahko zelo različni. Tam, kjer se zavedajo strateškega pomena priložnosti, ki jih omogoča sodobna informacijska tehnologija, običajno poteka pobuda in načrtovanje preobrazbe od zgoraj navzdol. So pa tudi okolja, iz takšnega tudi izhajam, kjer nova tehnologija vstopa pri stranskih vratih, z uvajanjem uporabe predmetnih programskih jezikov v izdelavo lastnih računalniških rešitev, s povezovanjem uporabnikov v enotno računalniško omrežje, z novimi računalniškimi rešitvami, ki dopolnjujejo dosedanje načine informacijske podpore. Osnovno vprašanje, na katerega želim podati odgovor, se glasi: Ali je zgolj znanje predmetnega programiranja zadosten pogoj za izgradnjo informacijskega sistema, ki sloni na uvajanju predmetno zasnovanih računalniških rešitev v nekem podjetju oziroma, na katera znanja se je koristno navezovati, da to dosežemo?

V nalogi skušam predstaviti čim bolj celovit spekter dejavnikov; procesov, metodologij, metod, vzorcev, referenčnih modelov, komunikacijskih in drugih sredstev, ki izhajajo iz znanstvenih spoznanj in so predlagani udeležencem v razvoju informacijskih sistemov kot praktični pripomočki za razumevanje njihovega poslanstva. Namen poglobljenega preučevanja omenjenih

dejavnikov je osredotočen na njihovo opisovanje, razlago in razumevanje njihove medsebojne povezanosti.

Izhajam iz teze, da lahko na osnovi celovite obravnave zahtevanih znanj v vsakem delovnem okolju prepoznamo specifične manjkajoče dejavnike, ki lahko kritično zavirajo ali pa pospešujejo večjo uspešnost prenove informacijske podpore in uvedbo predmetno zasnovanih računalniških rešitev. Obrnjeno drugače lahko vedno, glede na specifične razmere v izbranem okolju, izdelamo ustrezen vzorec aktivnosti, potrebnega znanja in ukrepanja za uspešno izvedbo projektov posodabljanja informacijske tehnologije. Predstavljeni nabor v dosegljivi strokovni literaturi najbolj pogosto navedenih dejavnikov, povezanih s predmetno zasnovo računalniških rešitev, obravnavam kot indikatorje, katerih prisotnost v izbranem okolju kaže na večjo verjetnost uspešne celovite informacijske podpore delovnim procesom.

Cilj, ki ga v splošnem zasledujem, lahko ponazorim s tremi vprašanji: Kako naj v podjetje vpeljemo predmetno zasnovane računalniške rešitve in pri tem povežemo tako tehnologe, razvijalce, upravo kot uporabnike? Kako naj načrtujem posodobitev informacijske podpore, da bo tveganje čim manjše in učinek čim večji? Kakšno znanje pri tem potrebujem in ali mi lahko pri tem kdo pomaga?

Eden od vzporednih ciljev naloge je usmerjen v iskanje ustrezne strategije pri uvajanju novih predmetno zasnovanih računalniških rešitev, ki bi lahko sobivale z nekaterimi obstoječimi rešitvami oziroma zasnovo informacijskega sistema. Vprašanje, na katerega želim odgovoriti, se glasi: Kakšne možnosti mi ponuja predmetna tehnologija na področju novih računalniških rešitev v odnosu na ohranjanje obstoječih rešitev? Na osnovi preučevanja različnih arhitektur sistema in pregleda nekaterih standardov, ki na področju predmetno zasnovanega programiranja usmerjajo razvoj programskih komponent, želim oceniti možnosti vključevanja novih in obstoječih računalniških rešitev. Podati želim smernice, ki naj bi omogočile načrtno uvajanje predmetne tehnologije v konkretno delovno okolje čim hitreje in s čim večjim učinkom.

## 1.3. Metode dela

Področje, ki ga preučujem, predstavlja kompleksen fenomen, kjer vložki in izidi niso vedno v jasni medsebojni povezavi. Zagotavljanje informacijske podpore in lasten razvoj računalniških rešitev predstavlja sistem prepletenih dejavnikov; udeležencev, metodologij, orodij in pripomočkov. Prizadeval sem si za čim bolj celovito in na nekaterih področjih tudi poglobljeno razumevanje obravnavanega področja. Uporabljena metoda dela je študija primerov različnih metodoloških pristopov za izvedbo informacijske podpore oziroma izdelavo računalniških rešitev, ki se najpogosteje navajajo v dosegljivi literaturi. Študija primerov se navezuje na preučevanje nekaterih teoretičnih izhodišč gradnje informacijskih sistemov ter možnosti, ki jih ponuja sodobna informacijsko komunikacijska tehnologija v povezavi s predmetno informacijsko tehnologijo. Skozi preučevanje primerov sem želel poiskati in podrobneje izpostaviti nekatere dejavnike, s pomočjo katerih iščem stične točke z načinom delovanja organizacije, organiziranostjo službe za informacijsko podporo, kadrovskimi viri ter potrebnimi znanji udeležencev v razvoju lastnih računalniških rešitev.

V nalogi je obravnavana problematika celovitega pristopa k informacijski podpori. Izhaja iz izkušenj in spoznanj profitnih organizacij. V iskanju širjenja in odličnosti izvajanja nekaterih storitev se delovanje neprofitnih organizacij čedalje bolj zgleduje po modelih načrtovanja, pristopih vodenja in organiziranosti profitnih vzornic. Na osnovi zaznanega zbliževanja

delovanja je vzpostavljena analogija med profitnimi in neprofitnimi organizacijami tudi na področju zagotavljanja informacijske podpore.

Podrobneje je obravnavana vloga informatike v organizaciji. Obravnavane so teoretične in praktične usmeritve in možnosti povezovanja poslovnega modeliranja, načrtovanja informatike in organiziranosti službe. S stališča tehnološkega razvoja je predstavljen koncept predmetno zasnovanega razvoja računalniških rešitev. Predstavljene so nekatere razvojne smernice na področju razpršene zasnove informacijskih sistemov in možnosti uporabe računalniških komponent. Na osnovi primerjav različnih metodologij razvoja računalniških rešitev in izkustvenih spoznanj, predstavljenih v različnih virih, sta podrobneje predstavljena dva pristopa k zbiranju zahtev uporabnikov. Referenčno je predstavljen primer standardnega procesa zbiranja zahtev uporabnikov in model, primeren za agilne metode programiranja.

V nalogi skušam celovito predstaviti informacijsko arhitekturo. Podanih je nekaj smernic razvoja, ki se navezujejo na komponentno zasnovo računalniških rešitev. Tako področje načrtovanja računalniških rešitev kot področje uporabe računalniških komponent in povezane informacijske arhitekture je obravnavano s stališča preučevanja nekaterih standardov s tega področja. Na osnovi celovitega pogleda na razvoj predmetnih računalniških rešitev v lastni režiji je izbranih nekaj pomembnejših dejavnikov uspeha za njihov razvoj s stališča notranje službe za informatiko. Ti dejavniki so kvalitativno ocenjeni z vidika nekaterih akterjev okolja, v katerem delujem. Spomočjo SWOT analize so nakazane smernice oziroma strateške aktivnosti, ki lahko vplivajo na izboljšanje stanja. Preučevanje različnih dejavnikov, udeleženih v razvoju informacijskih sistemov, je izvedeno na osnovi dosegljivih gradiv domače ter tuje literature in virov. Uporabljeni so tudi prispevki in članki z najnovejšimi spoznanji s področja informacijske tehnologije, informacijske infrastrukture, menedžmenta ter izgradnje informacijskih sistemov za podporo upravljanju in odločanju.

V nalogi sem uporabil znanje, pridobljeno na podiplomskem študiju na Ekonomski fakulteti v Ljubljani, ter lastne izkušnje. Za primer neprofitne organizacije, ki želi posodobiti svoj informacijski sistem, sem uporabil Zavod za gozdove Slovenije, kjer sem zaposlen. Pri iskanju odgovorov na zastavljena vprašanja nastopam kot notranji opazovalec v organizaciji, ki vidi potencial v uporabi predmetne tehnologije za nove računalniške rešitve, primanjkuje pa ji konceptualnih znanj, projektnega načina dela v razvoju računalniških rešitev, sodobnejših oblik vodenja in prožnejših oblik delovanja.

## 1.4. Struktura poglavij

Prvo poglavje predstavlja predstavitev problematike, obravnavane v nalogi, namene in cilje dela. Drugo poglavje je namenjeno predstavitvi storitev oziroma dejavnosti, ki so v javnem interesu. Izvedena je kratka predstavitev značilnosti neprofitnih organizacij, ki izvajajo dejavnosti v javnem interesu. Predstavljenih je nekaj priložnosti, ki jih prinaša informacijsko komunikacijska tehnologija. Obravnavani so motivi, ki lahko spodbujajo neprofitne organizacije k spremenjenemu načinu delovanja ter sodobni pristopi vodenja neprofitnih organizacij. Predstavljen je pomen strateškega načrtovanja in prevzemanje vzorcev ravnanja profitnih organizacij v neprofitni sektor. Vzpostavljena je analogija organizacije dela in delovanja med profitnimi in neprofitnimi organizacijami. Predstavljen je model učeče organizacije kot ena od oblik preoblikovanja delovanja organizacije v iskanju odzivov na zaznane spremembe notranjega in zunanjega okolja.

V tretjem poglavju je predstavljen pomen informatike v organizaciji. Obravnavan je pomen arhitekture informacijskega sistema in načrtovanja računalniških rešitev. Predstavljeno je modeliranje poslovnih procesov in možnosti povezovanja poslovnega modeliranja z modeliranjem računalniških rešitev. Obravnavana je vloga informatike v organizaciji in pomen strateškega načrtovanja informatike. Predstavljena je vloga informatike v podjetju in oblike organiziranosti ter delovanja službe za informatiko.

Prvi del četrtega poglavja je namenjen podrobnejši predstavitvi sodobne paradigme predmetno zasnovanega razvoja računalniških rešitev. Predstavljeni so koncepti, besednjak in pojmi, povezani z načrtovanjem in izvedbo računalniških rešitev. Obravnavani so dejavniki, ki omogočajo čim hitrejšo vpeljavo predmetne tehnologije. Predstavljen je pomen načrtovanja informacijskih rešitev, predmetno modeliranje ter jezik za modeliranje. V tem poglavju so obravnavani nekateri formalni pristopi v načrtni izdelavi programskih rešitev. Predstavljene so tradicionalne metodologije razvoja računalniških rešitev in nekaj novejših pristopov. Predstavljeni so dodatni načini načrtnega delovanja v razvoju računalniških rešitev. Obravnavani so procesni vzorci, ki omogočajo povezovanje različnih procesov v razvoju računalniških rešitev in procesna ogrodja, s katerimi lahko snujemo lastne razvojne metodologije. Za potrebe zbiranja zahtev in analize je podrobneje predstavljena metoda, ki sloni na primerih uporabe. Drugi del četrtega poglavja se posveča informacijsko komunikacijski tehnologiji in vidikom vzajemne povezanosti s predmetno tehnologijo razvoja računalniških rešitev. Obravnavan je pomen arhitekture informacijskih sistemov. Predstavljene so nekatere tehnološke specifikacije za izvedbo arhitekture sistema. Obravnavan je pomen načrtovalskih vzorcev v snovanju programskih rešitev ter vloga vmesnikov v zasnovi komponent.

Peto poglavje je namenjeno predstavitvi konkretnega delovnega okolja z neprofitnim značajem, v katerem delujem. Predstavljen je namen javne gozdarske službe, poslanstvo in izvajalci. Obravnavanih je nekaj motivov in dejavnikov, ki lahko pomembno vplivajo na snovanje oziroma prenovo informacijske posodobitve. Predstavljena je stopnja dosežene informacijske podpore v organizaciji. V luči nekaterih dejavnikov, ki so pogosto omenjani v preučevanih primerih, so podani predlogi in delni zaključki skladni z zaznanimi možnostmi in pogojih delovanja v organizaciji. Predlagana je uvedba prilagojene razvojne metodologije. V iskanju smernic delovanja službe za informatiko je opredeljenih nekaj dejavnikov, povezanih s predmetnim razvojem računalniških rešitev, ki so uporabljeni kot podlaga za SWOT analizo. Na osnovi analize so podane strateške aktivnosti za usmerjanje delovanja službe za informatiko. Nakazane so možne organizacijske spremembe in novi načini organizacijskega delovanja. Predstavljene so grožnje in priložnosti, ki izhajajo iz koncepta uvajanja predmetne informacijske tehnologije. V šestem poglavju so podane sklepne ugotovitve.

# 2. NEPROFITNE ORGANIZACIJE IN POMEN INFORMACIJSKE TEHNOLOGIJE ZA NJIHOVO DELOVANJE

Neprofitne organizacije so pomemben del sodobne družbe. Razumevanje razlogov za njihov obstoj je ključno za njihovo razlikovanje od profitno naravnanih organizacij. Neprofitne organizacije so najpogosteje ustanovljene z namenom, da bi zagotavljale storitve, ki jih širša družba spozna za zelo dragocene, profitne organizacije pa jih nočejo ali ne morejo zagotavljati širši družbi (Možina et al. 2002, str. 695). Neprofitno organizacijo v najširšem smislu je treba razumeti kot koncept, model izvajanja javnih služb in ne kot posebno organizacijsko obliko. Če neprofitno organizacijo razumemo v tem kontekstu, lahko rečemo, da nedobičkonosno deluje na področju zadovoljevanja javnih dobrin in v splošno družbeno korist.

Pravno-organizacijske oblike neprofitnih organizacij so podjetja, zavodi, ustanove in društva. Njihovo delovanje v Sloveniji je predpisano z zakoni za delovanje posamezne oblike organizacije (Zakon o gospodarskih družbah, 2006, Zakon o zavodih, 1991, Zakon o ustanovah, 1995 in Zakon o društvih, 2006).

V današnjem času se priložnosti informacijsko komunikacijske tehnologije (IKT) dokazujejo na številnih področjih delovanja gospodarstva in širše družbe. Z uvajanjem in posodabljanjem IKT prihaja do novih proizvodov, storitev in načinov poslovanja. Profitne organizacije se v iskanju konkurenčnih prednosti tradicionalno posvečajo izzivom uvajanja novih tehnologij. Temu prilagajajo svoje strategije razvoja, organiziranost in načine delovanja. Tudi neprofitne organizacije se čedalje pogosteje zgledujejo po načinu vodenja, konceptih, postopkih in orodjih, ki jih na področju IKT uporabljajo v profitnem sektorju. V tem pogledu IKT prinaša v neprofitni sektor številne izzive, ki jih lahko povežemo s strategijo njihovega delovanja v izrabi IKT, od organiziranosti službe za informacijsko podporo do izdelave računalniških rešitev.

Državna uprava predstavlja enega večjih izvajalcev storitev, povezanih z javnim interesom, kjer se jasno zavedajo priložnosti, ki jih ponuja informacijsko komunikacijska tehnologija. Na nivoju Evropske unije (Resolucija evropskih ministrov o javni upravi in javnih storitvah) ministri priznavajo, da odpirajo informacijsko komunikacijske tehnologije neslutene priložnosti za preprostejši dostop do podatkov javnega značaja in da bodo bolj jasno opredeljeni pogoji za njihovo uporabo prispevali k višji gospodarski rasti in zaposlovanju po vsej Evropi. Informacijsko komunikacijska tehnologija omogoča v tem pogledu učinkovito ogrodje pri zagotavljanju kakovostnih storitev uprave na osnovi načel odprtosti, možnosti sodelovanja vseh, zanesljivosti, učinkovitosti in poenotenosti.

Tudi poskusi, da bi razvili učeče se skupnosti, so del strategije, ki so jo sprejele evropske vlade in izobraževalne inštitucije kot odgovor na zaznane potrebe informacijske družbe, kjer naraščata potreba in interes posameznikov, gibanj in inštitucij za uveljavljanje kakovosti življenja, humane cilje in človekoljubne dejavnosti. V tem pogledu je mogoče pričakovati porast graditve različnih skupnosti s skupnimi interesi. V tej rabi se skupnost ne zanaša na konceptualizacijo fizičnega prostora, temveč na pojme družbenega prostora. Socialna kohezija zahteva, da se udeležba razširi čez meje lokalnih skupnosti in jih poveže v širšo celoto. V oskrbi s podatki in informacijami ter tudi v izvajanju novih storitev ima informacijsko komunikacijska tehnologija za področje delovanja neprofitnega sektorja velik potencial. Poraja se mnogo vprašanj, kako ga čim hitreje in učinkoviteje izrabiti.
Med profitnimi in neprofitnimi organizacijami obstajajo značilne razlike, ki se odražajo v iskanju ravnovesja med nedobičkonosnim poslanstvom in ekonomsko logiko poslovanja. Delovanje neprofitnih organizacij je izvzeto iz tržnih mehanizmov regulacije in financiranja. S tem je

povezano financiranje, kjer v ozadju finančne podpore poslanstvu neprofitnim organizacijam naletimo na državni proračun, prispevke donatorjev in prispevke potencialnih uporabnikov storitev. Neprofitne organizacije tako kot profitne potrebujejo navezanost na porabnike svojih storitev, saj z njimi v okviru izvajanja svojega poslanstva utemeljujejo (dokazujejo) koncept organiziranosti in obseg sredstev financiranja. Ena od značilnosti neprofitnih organizacij je pretežna odvisnost od financiranja iz javnih virov. Med posledicami netržnega delovanja neprofitnih organizacij je včasih omenjena šibkost vodenja in organizacije, ki ovira mnoge majhne in velike neprofitne organizacije, da bi bolj učinkovito opravljale svoje poslanstvo (Mesec, 2006). Omenjeni dejavniki lahko omejevalno vplivajo tudi na posodabljanje IKT, namenjeno delovanju neprofitnih organizacij.

Informacijsko komunikacijska tehnologija se kot generična tehnologija lahko uporablja v vseh dejavnostih. Analize navajajo, da so vplivi informacijsko komunikacijske tehnologije na učinkovitost potencialno največji prav v storitvenih dejavnostih, čeprav ob tem poudarjajo, da je treba upoštevati raznolikost storitvenih dejavnosti (Stare, Bučar, 2005, str. 99).

Rezultati preučevanja potencialov uvajanja informacijsko komunikacijske tehnologije v Sloveniji kažejo na neizkoriščene možnosti na rast produktivnosti v storitvenih podjetjih in v zagotavljanju bolj celovitih storitev na področju javne uprave. Ugotovitve kažejo, da je glavna ovira v neizkoriščenosti tega potenciala pomanjkanje znanja. V celovitem uvajanju informacijsko komunikacijske tehnologije je izpostavljena vloga strokovnjakov z interdisciplinarnimi znanji, ki kombinirajo tehnična, organizacijska in menedžerska znanja (Stare, Bučar, 2005, str. 179).

Na osnovi gornjih navedb lahko rečemo, da pri uvajanju in posodabljanju IKT, ki bo služila boljšemu delovanju neprofitnih organizacij, obstaja neizkoriščen manevrski prostor. Ta manevrski prostor obsega tako iskanje novih priložnosti za distribucijo storitev s pomočjo IKT kot tudi iskanje drugačne organiziranosti, spremenjenega vodenja ter prilagojenega delovanja službe, ki v neprofitni organizaciji skrbi za informacijsko podporo. V tej povezavi neprofitne organizacije potrebujejo sodobna znanja tako s področja menedžmenta kot s področja načrtovanja informacijskih sistemov, ki delujejo v okolju informacijsko komunikacijske infrastrukture. Veliko teh znanj lahko pridobimo tudi s preučevanjem delovanja in uporabe informacijske tehnologije v profitnih organizacijah.

## 2.1 Storitve v javnem interesu, neprofitne organizacije

Neprofitna organizacija je ciljno orientiran, socialen, odprt, dinamičen in sestavljen sistem. Njen cilj je zadovoljevati potrebe različnih interesnih skupin s proizvodi in storitvami. Ustvarjenega dobička organizacija ne sme razdeliti med člane, temveč ga namenja za razširitev, rast in izboljšanje kakovosti storitev in dobrin. Neprofitne oziroma nedobičkonosne organizacije so skupni pojem za javno upravo, za družbene dejavnosti in za prostovoljne organizacije, ki poslujejo brez dobička, ali pa z njim, vendar cilj njihovega poslovanja ni dobiček; če do njega pride, se z njim ne razpolaga po svobodni presoji, ampak se le-ta vlaga nazaj v dejavnost organizacije in služi kot sredstvo za razširitev te dejavnosti ali pa za dvig kakovosti storitev (Možina et al. 2002).
Skupino neprofitnih organizacij lahko razdelimo na javni sektor, ki pokriva prostor z odločilnim vplivom države, mešane javno-zasebne in zasebne organizacije ter na nevladne organizacije in skupnosti (Mesec, 2006). Javni sektor je zbir vseh javnih organizacij, ki opravljajo družbene in gospodarske javne dejavnosti po netržnih načelih, kar se v prvi vrsti kaže s proračunskim financiranjem. Javne organizacije so vse pravno urejene in se pojavljajo v različnih statusno

pravnih oblikah. To so predvsem javni zavodi, zavodi s pravico javnosti, javni gospodarski zavodi, javna podjetja in organizacije v sistemu državne uprave. Javne organizacije niso nujno organizirane kot neprofitne. Pojma javni in neprofitni ne gre enačiti, čeprav so največkrat negospodarske javne službe organizirane kot neprofitne (Mesec, 2006).

## 2.2 Strateško načrtovanje v neprofitnih organizacijah

Značilne razlike med profitnimi in neprofitnimi organizacijami po mnenju nekaterih zaznamujejo naslednja področja (Možina et al. 2002, str. 697):

- poslanstvo in organiziranje dejavnosti,
- strateško odločanje in načrtovanje,
- poslovni rezultati in merjenje učinkov.

Podrobnejše preučevanje pokaže na razlike med profitnimi organizacijami tudi na drugih področjih. Nekatere značilne razlike so podane v tabeli 1.

Tabela 1: Razlike med profitnimi in neprofitnimi organizacijami

| **Značilnosti** | **Profitne organizacije** | **Neprofitne organizacije** |
|---|---|---|
| **Cilji in strategije** | • Svobodno odločajo o ciljih, panogah odjemalcev in o strategijah do njih.<br>• Spremembe menedžmenta so lahko nagle in številčne. | • Izvajanje storitev je obvezno, nedopustno je opuščanje ali spreminjanje programov, financerji strogo omejujejo izbiro strategij. |
| **Viri financiranja** | • Sredstva si priskrbijo s tržno dejavnostjo. Večinoma skušajo povečevati obseg in vrednost storitev.<br>• Pridobivajo nove odjemalce, skušajo povečevati delež, ki ga imajo na tržišču. | • Organizacije, ki prejemajo sredstva od uporabnikov:<br>o so odvisne od tržišča,<br>o skušajo povečevati število odjemalcev.<br>• Organizacije, ki prejemajo sredstva iz javnih virov:<br>o pridobivanje sredstev je le deloma povezano z obsegom in kakovostjo storitev,<br>o pomembno je zadovoljstvo virov sredstev,<br>o skušajo omejevati število odjemalcev, če ne vpliva na višino in kakovost sredstev. |
| **Strokovnjaki in menedžerji** | • Upoštevana so strokovna in menedžerska znanja ter osebnostne lastnosti, strokovnjaki počasneje napredujejo kot menedžerji. | • Strokovnjaki imajo večjo veljavo kot menedžerji.<br>• Strokovnjaki pogosto nimajo menedžerskih znanj, lahko imajo strokovne interese, ki niso nujno enaki smotrom organizacije. |

| Značilnosti | Profitne organizacije | Neprofitne organizacije |
|---|---|---|
| **Upravljanje in usmerjanje** | • Lastniki močno vplivajo na obvladovanje organizacije.<br>• Menedžment ima na skrbi predvsem učinkovitost, upravljavci pa usmerjanje in nadzor nad uspešnostjo organizacije. | • Lastništvo je zamegljeno, odloča upravni odbor, ki ni plačan. |
| **Vrhovni menedžment** | • Velika in nedeljena oblast vrhovnega menedžmenta. | • Pogosto je dvojno poslovodenje:<br>o v državnih organizacijah politika in državni aparat,<br>o v profesionalnih organizacijah stroka in poslovodenje,<br>o v prostovoljskih organizacijah prostovoljci in plačani funkcionarji. |
| **Vpliv politike** | • Vpliv je omejen, organizacije upoštevajo interese okolja na analitičen način in dolgoročno. | • Vplivi so različni:<br>o kratkoročno obnašanje voljenih managerjev, upravljavcev, dodeljevalcev sredstev,<br>o zahteve javnosti po transparentnosti delovanja,<br>o skrivanje informacij zaradi konkurenčnosti. |
| **Merjenje uspešnosti** | • Dobiček je sinteza sestavin uspešnosti. | • Količino in kakovost storitev je težko meriti, ker niso finančno ovrednotene. |
| **Lastništvo** | • Delničarji vložijo svoja sredstva v kapital družbe. | • Sredstva so last organizacije in ne posameznikov. |
| **Pridobitna dejavnosti** | • Glavna dejavnost družbe. | • Lahko jo opravljajo pod posebnimi pogoji. |

Vir: povzetek, Mesec, 2006

Med osnovne prvine načrtovanja v neprofitnih organizacijah se uvršča opredelitev poslanstva. Prva dolžnost vodilnega menedžmenta je, da vsi v organizaciji razumejo in sprejemajo poslanstvo. Pomen poslanstva je izpostavljen zaradi dejstva, da menedžerji z njim lahko delno kompenzirajo ekonomske težave neprofitnih organizacij in olajšajo dileme kratkoročnih poslovnih odločitev (Mesec, 2006). Določitev poslanstva naj bi bila začetna faza procesa strateškega ravnanja organizacije. Med druge elemente neprofitnega načrtovanja poleg opredelitve poslanstva uvrščamo še:

- Strateško prednost: organizacija mora imeti stvarne, ustrezne in učinkovite strateške načrte, ki zagotavljajo najboljšo izrabo njenih potencialov in razpoložljivih virov. Enako pomembno je, da ima postavljene sisteme dobro pripravljenega in enotnega načrtovanja,

spremljanja in ocenjevanja njenega delovanja. Prav tako mora biti sposobna in dovolj močna prilagoditi načrte nepričakovanim spremembam.

- Podporo in omrežje: podpora mora biti aktivna in mora izražati privrženost članov, kot da organizacija v resnici pripada njim. Uprava mora imeti stalen dostop do vseh potrebnih informacij. Organizacija mora imeti jasen koncept odgovornosti (do svojih članov, ciljnih skupin in donatorjev) in zavezanost k sodelovanju na vseh ravneh. Z vlado in vsemi ustanovami, ki delujejo na istem področju, morata biti vzpostavljena dobra povezava in sodelovanje.
- Vire: organizacija mora imeti ustrezna sredstva, ki ji omogočajo izvajanje njenih dejavnosti. Hkrati mora imeti dolgoročno strategijo svojega lastnega financiranja in svojih programov. Strategija mora vsebovati določeno stopnjo finančne neodvisnosti in raznovrstnost virov prihodka.
- Programe: programi organizacije morajo biti uravnoteženi in uresničljivi. Biti morajo stabilni in dolgoročni.
- Skupine: organizacija mora imeti zadostno število privrženih in sposobnih ljudi, ki delujejo timsko pri reševanju problemov. Vodenje mora biti prilagodljivo, usmerjeno v akcije in sposobno razreševati nasprotja.

V teoriji je poznanih več modelov strateškega menedžmenta, ki pa so v glavnem razviti za profitno naravnane organizacije. Eden od modelov strateškega načrtovanja, razvitih posebej za neprofitne organizacije, je Brysonov model. Vsebuje deset korakov strateškega načrtovanja, od začetnega sporazuma o organizaciji strateškega menedžmenta do kontrole uresničevanja strategij.

Eden ključnih korakov v Brysonovem modelu je snovanje strategij. Splošna definicija strategije ne obstaja, zato različni avtorji ponujajo različne pristope. Po Coulterju je strategija serija ciljno usmerjenih odločitev in akcij, ki uravnotežijo organizacijske sposobnosti in vire s priložnostmi in nevarnostmi iz okolja (Možina et al. 2002, str. 717). Thompson in Strickland navajata, da je strategija vzorec aktivnosti in poslovnih pristopov v uporabi strateškega poslovodstva, z namenom ugoditve zahtev kupca ali naročnika, izgradnje dobre tržne pozicije ter uresničevanja organizacijskih ciljev (Možina et al. 2002). Po Lynchu je strategija vzorec osnovnih ciljev, smotrov, namenov, ključnih politik in načrtov, potrebnih za uresničevanje organizacijskih ciljev (Možina et al. 2002).

Načini snovanja strateškega načrtovanja v neprofitnih organizacijah se praviloma zgledujejo po konceptih podjetniškega strateškega načrtovanja. Temeljijo na konceptih, postopkih in orodjih, ki so uporabni tudi pri odzivanju na spremembe v neprofitnem okolju (McNamara, 2008).

Med glavne pristope k snovanju strategij za neprofitno okolje lahko uvrščamo (Možina et al. 2002, str. 718-723):

- Harvardski model politike organizacije: sloni na analizi prednosti, slabosti, nevarnosti in priložnosti (SWOT), usmerjen je v razvoj ujemanja med organizacijo in njenim okoljem.
- Sistem strateškega načrtovanja: temelji na formiranju odgovorov na vprašanja: kam gremo (poslanstvo), kako tja priti (strategije), kakšen je načrt akcije (proračun), kako vemo, da smo na pravi poti (nadzor).
- Pristop menedžmenta udeležencev: sloni na preučevanju križanja in usklajevanja skupin udeležencev, ki kakor koli nastopajo znotraj ali zunaj organizacije in na katere organizacija vpliva.
- Portfeljski model: sloni na matriki BCG (bostonska svetovalna grupa), ki poslovna področja kategorizira na štiri tipe glede na stopnjo rasti njihove panoge in stopnjo rasti

tržnega deleža. Ker uporablja samo ekonomske kriterije, je morda manj primerna za neprofitne organizacije.
- Analiza konkurenčnih prednosti: model predpostavlja analizo konkurenčne prednosti na ravni panoge na osnovi konkurenčnih sil pogajalske moči dobaviteljev, pogajalske moči kupcev, nevarnosti vstopa konkurence, nevarnosti substitucije, konkurence med podjetji v panogi.
- Menedžment strateških vprašanj: vrhnji menedžment imenuje začasne time za hitro oblikovanje odzivov na spremembe.
- Logični inkrementalizem: strategija predstavlja skupino odločitev, ki se postopno obravnavajo. Odločitve se sprejemajo na nižjih ravneh organizacije, kjer je tudi več informacij za sprejem dobrih odločitev.
- Model strateškega načrtovanja kot ogrodje za inovacije: model dopušča inovacije in podjetništvo, hkrati pa vzdržuje centralni nadzor. Uporablja analizo SWOT in portfeljske metode.

V prizadevanjih za izboljšanje izkoriščanja potencialov in virov delovanja neprofitnih organizacij na splošno lahko obravnavamo strateško načrtovanje kot temeljni gradnik načrtovanja. Z njim usmerjamo delovanje tako na ravni organizacije kot na ravni posameznih organizacijskih ali programskih enot. Na ravni zaznanih sprememb okolja se s pomočjo strateškega načrtovanja lahko ustrezno odzivamo in usmerjamo delovanje notranjih virov.

## 2.3 Učeča organizacija kot odgovor na zahteve po spremembah

Spremembe v okolju delovanja organizacij spodbujajo k iskanju novih oblik organiziranosti in delovanja. Ena od oblik delovanja, ki spreminja procesno organiziranost in pomeni nov pristop v razumevanju organizacije, je učeča organizacija (Dimovski et al. 2005).

Principi delovanja učeče organizacije omogočajo njihovo izboljšano poslovanje in konkurenčno prednost. V njih ljudje povečujejo svoje sposobnosti za doseganje rezultatov na osnovi skupinskega dela, svobodnega usmerjanja skupnih prizadevanj, sistemskega učenja ter distribucije zbranega znanja v sistemu svoje organiziranosti (Peterlin, 2007, str. 21-23).

Analiza slovenske javne uprave in organizacijskih vplivov uvajanja informacijsko telekomunikacijske tehnologije (elektronski zajem, procesiranje, shranjevanje in komuniciranje) je pripeljala do ugotovitve, da je učeča se organizacija ključni dejavnik uspeha. Pripravljenost na spremembe, decentralizacija, vzpostavljanje odgovornosti in delegiranje, stalno izobraževanje, mreženje znanj in nagrajevanje zaposlenih pa ključni procesi tega učenja. Rezultati analize so nakazali, da bi boljša in učinkovitejša uporaba z informacijsko tehnologijo povezanih orodij in storitev pripomogla k izboljševanju poslovnih procesov, sodelovanju, informiranju, prenosu znanja, s tem pa bi omogočili večjo dodano vrednost človeških virov (Stare, Bučar, 2005, str. 164).

Pojem učeča organizacija označuje organizacijo, sposobno pridobivati in prenašati znanje med zaposlenimi ter hkrati oblikovati vedenje, ki odraža pridobljeno znanje (Možina et al. 2002, str. 17). Pri tem učenje ni omejeno samo na menedžment, ampak naj bi doseglo vse ravni organizacije, vse njene poslovne funkcije. Ena od skritih vrednosti vsake organizacije je intelektualni kapital, ki je opredeljen kot eden virov primerjalne prednosti. Večinoma je deljen na človeški kapital (osebno znanje, sposobnosti za opravljanje nalog), kapital organizacije (tehnološke procese, tok informacij) in kapital poslovnih odnosov (informacije o odnosih s poslovnimi partnerji). Na ravni organizacije znanje intelektualnega kapitala omogoča sinergijske

učinke, ki segajo od individualnega preko skupinskega in organizacijskega do interorganizacijskega znanja (Možina et al. 2002, str. 19).

V svojem bistvu je učeča organizacija svojevrsten mehanizem organizacije za spodbujanje, kreiranje, shranjevanje in distribucijo znanja zaposlenih, ki želi preseči odvisnost od individualnega znanja. V ciklusih ponavljajočega akumuliranja in distribucije zbranega znanja si učeča organizacija prizadeva povečevati svoj potencial delovanja, hkrati pa omogoča posameznikom, ki v takšni organizaciji delujejo, tudi njihov razvoj ter možnost lažjega odzivanja na spremembe. Takšen pristop povečuje ustvarjalne možnosti delovanja zaposlenih, saj slednji svoje formalno pridobljeno znanje lahko lažje prilagodijo na kreativno reševanje problemov s področja konkretnega delovanja organizacije.

Med ključne prednosti, ki jih po raziskavah dosegajo učeče organizacije, se uvrščajo (Možina et al. 2002, str. 23):

- sistematično reševanje problemov,
- preizkušanje novih pristopov,
- učenje na podlagi preteklih izkušenj,
- učenje iz primerov drugih,
- hitro prenašanje znanja na vse ravni organizacije.

Temeljni vzorec delovanja učeče organizacije sloni na timskem delu in zmanjševanju števila individualnih nalog. Značilnost takšne organizacije je decentralizacija odločanja in spodbujanje avtonomnega dela skupin. Učeča organizacija izpostavlja pomen pretoka znanja in izmenjave izkušenj. Pretok znanja ne poteka od vrha navzdol, ampak postaja vzajemen učni proces, ki ga spodbuja in omogoča vodstvo. V tej vlogi prenašanja odgovornosti na zaposlene nanje prenaša tudi podatke (informacije), ki jih potrebujejo za delovanje in odločanje. V procesu pretoka podatkov učeče organizacije čedalje bolj izkoriščajo možnosti informacijsko komunikacijske tehnologije. Sem sodi uporaba programske opreme za skupinsko delo, uporaba sistemov za podporo odločanju (angleško: decision support systems), uporaba izvršilnih informacijskih sistemov (angleško: executive information systems) in uporaba ekspertnih sistemov. Poleg tega je izpostavljen pomen uporabe interneta in možnosti uporabe medorganizacijskih informacijskih sistemov (Dimovski et al. 2005, str. 188).

V obdobju sprememb, ki prihajajo iz notranjega in zunanjega okolja, iz razvoja informacijske tehnologije ter zahtev po izboljšanju kakovosti izdelkov in storitev, si organizacije na različne načine prizadevajo odzvati na spremembe. Učeče organizacije uvajajo in gradijo spremembe na področjih spreminjanja funkcije vodij, decentralizacije odločanja in participacije zaposlenih, pooblaščanja zaposlenih in delitve odgovornosti, samostojnem delovanju timov, dostopnosti informacij in prilagodljivi organizacijski kulturi, ki ustvarja občutek pripadnosti organizaciji (Dimovski et al. 2005, str. 106). Med ključnimi nosilci sprememb v organizaciji so vodje, ki zaposlenim omogočajo priložnosti za njihove dosežke in razvoj.
S svojim aktivnim delovanjem lahko vodja ustvari ugodno delovno okolje, kjer bodo zaposleni sodelovali pri oblikovanju delovnih nalog (participativni slog vodenja) (Dimovski et al. 2005, str. 275). Za uspešno delovanje skupin je pomembno poznavanje tudi nalog drugih udeležencev in poznavanje pričakovanj njihovega vloženega dela. Pri tem je ključnega pomena informiranje udeležencev in zagotavljanje povratnih informacij, ki lahko pokaže na vrzeli v znanju posameznika. Analiza povratnih informacij tudi pogosto pokaže na nezadostno znanje ali podcenjujoč odnos do znanja zunaj stroke posameznika.

## 2.4 Posodabljanje uporabe informacijsko komunikacijske tehnologije, analogija med profitnimi in neprofitnimi organizacijami

V sferi storitvenega sektorja je bilo ugotovljeno, da je za obvladovanje sprememb pri uvajanju novih tehnologij treba zagotavljati dotok spremljajočega znanja na različne ravni upravljanja in organizacije dela v podjetjih. Študije na ravni slovenskih podjetij navajajo, da je izkoriščanje prednosti IKT zahteven proces, ki terja vrsto spremljajočih, organizacijskih, menedžerskih in inštitucionalnih sprememb.

V zadnjih desetih letih se prevladujoče uveljavlja mnenje, da neprofitni sektor potrebuje posebna menedžerska znanja (Možina et al. 2002, str. 705). Temeljna vloga neprofitnih organizacij je izpolnjevanje najvišjih načel sodobne družbe. V neprofitnih organizacijah se vloga menedžmenta čedalje bolj spreminja od vloge motivacijskih vodij k novim menedžerskim vlogam, od tradicionalne administrativne kulture k racionalnemu menedžerskemu slogu vodenja. Znanja o menedžmentu neprofitnih organizacij so specializirana sestavina splošnih znanj o menedžmentu, zato je racionalno privzemati splošna menedžerska znanja ter jim dodajati znanja, ki zadevajo specifičnost neprofitnih organizacij (Mesec, 2006). Podobno analogijo med profitnimi in neprofitnimi organizacijami lahko vzpostavimo tudi za področje uvajanja in posodabljanja informacijske tehnologije. Na osnovi preučevanja študijskih primerov slovenskih podjetij sodijo med najpomembnejše dejavnike uspeha načrtovanje, vloga informacijske tehnologije (IT) v strateških razvojnih programih, pogled menedžmenta na tehnologijo, IT oddelki v podjetju, hkratno uvajanje organizacijskih sprememb, znanje, usposobljenost, izobraževanje zaposlenih, izbira zunanjih partnerjev in odnosov z njimi ter skrb za celovito motiviranost in komunikacijo (Stare, Bučar, 2005, str. 138).

Uvajanje informacijsko komunikacijske tehnologije v organizacijo poteka skozi izvajanje dejavnosti informatike. Slednjo izvajamo skozi različne aktivnosti, med katerimi se najpogosteje prepletajo zagotavljanje infrastrukture v obliki naprav in povezav, skrb za vzdrževanje podatkovnih zbirk, razvoj in uvajanje programske opreme za pomoč pri izvajanju procesov organizacije ter skrb za pomoč uporabnikom pri uporabi informacijsko komunikacijske opreme. Delovanje službe za informatiko v podjetju sloni na izvedbi procesov, v katerem se znanje informatikov prepleta z znanjem in izkušnjami drugih udeležencev, ki zasledujejo iste cilje organizacije. Današnje zahteve po znanju informatikov presegajo zgolj poznavanje programiranja (Kovačič, Bosilj-Vukšić, 2005, str. 250). Razvoj IKT spremljajo zahteve tako po razširjanju znanja na druga področja kot tudi zahteve po poglabljanju specialnih znanj. Oglejmo si podrobneje vlogo dejavnosti informatike v organizaciji ter nekatere ključne dejavnike uspešnega razvoja lastnih računalniških rešitev s pomočjo predmetne informacijske tehnologije.

# 3. INFORMATIKA V ORGANIZACIJI IN INFORMACIJSKI SISTEM

Uporaba tehnologije predstavlja sistematično uporabo znanstvenega in drugega organiziranega znanja pri izvedbi praktičnega dela. Posledica uvedbe tehnoloških sprememb so lahko tudi družbene spremembe, če se nanjo prilagodijo vse gospodarske in politične inštitucije, skupno z ideologijo, obnašanjem in stališči. Prenos in uvajanje tehnologije zahteva vedenje, znanje, organizacijo in izkušnje (Čuš, 1997 str. 119-129).

Delujemo v obdobju, ki ga je zaznamovala informacijska revolucija. Sprožil jo je razmah razvoja in uporabe informacijske in komunikacijske tehnologije. Informacijska revolucija povzroči prehod v informacijsko družbo takrat, ko se večina ljudi ukvarja z informatiko. V informacijski družbi težišče ekonomskih aktivnosti in tehnoloških sprememb postane obdelava podatkov in informacij. Večina narodnega gospodarstva postane odvisna od informacijske tehnologije (Gradišar, Resinovič, 2001, str. 3). Današnje hitro spreminjajoče globalno gospodarstvo prevzema informacijsko tehnologijo kot gonilno silo razvoja, ki vpliva na posodabljanje načinov poslovanja in iskanje konkurenčnih prednosti. Govorimo o digitalni ekonomiji, E-upravi in globalni spletni povezanosti, kjer ima tudi vsak posameznik možnost postati najmanj viden, če že ne svetovno konkurenčen s svojim znanjem. Družbena sprememba, ki smo jo poimenovali prehod v informacijsko družbo, je sprožila zahteve po pridobivanju novih znanj ter spreminjanju načina našega razmišljanja in dosedanjega organiziranega delovanja.

Zavestno dejavnost ljudi, ki se izvaja in vzdržuje z njihovo voljo, da bi se ustvarila celota iz množičnosti, predstavlja **organizacija**. Izvajanje delovanja organizacije vključuje organizacijo dela in usklajevanje proizvodnih tvorcev v vseh pripravljalnih, izvajalnih in zaključnih fazah celotnega poslovnega procesa (Možina et al. 2002, str. 375). Z razvojem tehnologije se spreminja tudi odnos med neposrednimi proizvodnimi aktivnostmi in tistimi, ki omogočajo ter zagotavljajo ta delovni proces. Na spremembo teh odnosov ima pomemben vpliv informacijska tehnologija, ki prinaša v organizacijo številne nove priložnosti.

**Informacijska tehnologija** predstavlja sredstva in vedenje o obravnavanju in distribuciji podatkov ter o oblikovanju informacij. Sestavljajo jo računalniška oprema, računalniški programi, telekomunikacije ter ustrezne tehnike in postopki (Možina et al. 2002, str. 621). Na področju uporabe informacijske tehnologije se pogosto uporablja izraz **informacijsko komunikacijska tehnologija**. Ta definicija je bila vzpostavljena za namene proučevanja različnih vplivov informacijske tehnologije na spremembe v gospodarskem razvoju in za vzpostavitev enotnih mednarodnih primerjav (OECD, 2000). Na teh osnovah sloni tudi razvrščanje informacijske opreme in storitev v razrede mednarodne standardne industrijske klasifikacije (ISIC, rev. 3) ter postavke naše standardne klasifikacije dejavnosti (Stare, Bučar, 2005, str. 15-21).

**Informacijska veda**, kot interdisciplinarna znanstvena veja, zagotavlja znanstveno podlago zbiranju, klasifikaciji, hranjenju, prikazovanju in objavljanju informacij. V tem pogledu preučuje vzajemno delovanje med ljudmi, organizacijami in informacijskimi sistemi, in presega okvir delovanja računalniške znanosti, ki se prvenstveno ukvarja s teoretskimi osnovami procesiranja informacij v računalniških sistemih.

Dejavnost informatike pomeni uvajanje vedenja in izsledkov informacijske vede v operativno delo. **Informatiko** lahko obravnavamo kot dejavnost oblikovanja, uvajanja in izvajanja informacijskih sistemov v organizaciji.

Obsega ugotavljanje potreb po podatkih in informacijah, njihovo organiziranost, uporabo informacijske tehnologije in izdelavo računalniških rešitev (Možina et al. 2002, str. 624). Druga definicija pravi, da je informatika znanstvena disciplina, ki se ukvarja s strukturo, programskimi jeziki in programiranjem naprav za obdelavo podatkov, pa tudi z metodologijo njihove uporabe, vključno z vzajemnim vplivom med človekom in strojem (Gradišar, Resinovič, 2001, str. 2).

V izvedbi dejavnosti informatike pogosto uporabljamo pojme, kot so **informatizacija**, informatizacija poslovanja in informatizacija poslovnih procesov. Gre za aktivnosti, ki jih izvajamo kot proces, usmerjen v celovito uvedbo informacijske tehnologije v podjetje. Vključuje uvajanje uporabe orodij, programskih rešitev in iskanje informacijskih rešitev za podporo poslovnim procesom podjetja (Kovačič, Bosilj-Vukšić, 2005, str. 14, 352, 417). S stališča organizacije je informatizacija namenjena upravljanju njenega delovanja, procesiranju podatkov in informacij, upravljanju znanja s področja poslovnih procesov podjetja ter vključevanju konceptov informacijske vede za izboljševanje poslovnih procesov.

Informatizacija organizacije temelji na vzpostavljanju in vzdrževanju informacijskega sistema. **Informacijski sistem** organizacije je skupek ljudi, procesov in informacijske tehnologije, ki zagotavlja podatke in informacije, potrebne za njeno delovanje in usklajevanje delovanja za doseganje zastavljenih ciljev (Možina et al. 2002, str. 625).

Informacijski sistem je namenjen obdelavi podatkov in s svojimi storitvami (aplikacijami) posredno ali neposredno omogoča končnim uporabnikom njihovo interpretacijo. Končni uporabniki informacijskega sistema se v svojem delovanju tako znotraj organizacije kot zunaj nje medsebojno povezujejo. Povezovanje prometa podatkov med posameznimi točkami informacijskega sistema sloni na storitvah, ki jih zagotavljajo **telekomunikacijski sistemi**. Sodobno delovanje informacijskih sistemov je v pogojih medsebojnega povezovanja končnih uporabnikov čedalje bolj povezano s telekomunikacijskimi sistemi. Integracijo informacijskega okolja končnih uporabnikov na osnovi komunikacijskih sistemov opredeljujemo kot **informacijsko komunikacijski sistem** (Vidmar, 2002, str. 19-57).

Informacijska tehnologija predstavlja del informacijsko komunikacijskega sistema, ki se ukvarja s preučevanjem, načrtovanjem, razvojem in uvajanjem računalnikov ter programske opreme za njegovo delovanje. Komunikacijska tehnologija je v tem sistemu namenjena povezovanju delovnih mest, uporabnikov ter računalniških sistemov (strojne in programske opreme) (Vidmar, 2002, str. 65-69).

Razvoj informacijskega sistema lahko opredelimo kot proces, v katerem je olajšana ali izboljšana skupina človeških aktivnosti ob pomoči informacijske tehnologije, ki smo jo uporabili na osnovi analize, načrta, izdelave, uvedbe in za katero zagotavljamo podporo (Mursu et al. 2007).

Informatika zagotavlja organizaciji konceptualno znanje s področja upravljanja podatkov in informacij ter tehnološko podporo za avtomatizacijo poslovanja, distribucijo znanja in informatizacijo poslovnih procesov.

Posodabljanje načinov poslovanja podjetij je zgodovinsko gledano povezano tudi z razvojem informacijske in telekomunikacijske tehnologije. Informatizacija predstavlja odziv podjetij na priložnosti, ki jih vidijo v možnosti uporabe informacijske tehnologije. Najbolj značilni trendi, povezani z razvojem informacijske tehnologije, so (Gradišar, Resinovič, 2001, str. 11):

- večja dostopnost računalniške tehnologije,
- povezovanje računalniške in komunikacijske tehnologije,

- višja stopnja avtomatizacije podjetja,
- večja vrednost v obliki informacij,
- nove oblike organizacije,
- krajši poslovni cikli,
- večja svetovna konkurenca,
- postopno uvajanje svetovnih standardov.

Razvoj telekomunikacijskih sistemov je v veliki meri povezan z razvojem informacijskih sistemov, saj je z njimi neločljivo povezan. Osnovna značilnost telekomunikacijskih sistemov je medsebojno povezovanje različnih informacijskih okolij in omogočanje komunikacije oziroma izmenjave podatkov med njimi. Tehnološki razvoj telekomunikacijskih sistemov je zaznamovan predvsem z nadomeščanjem začetnih analognih prenosnih naprav, kot so parica in koaksialni kabli z optičnimi kabli, ter s povezanim nadomeščanjem analognih sistemov z digitalnimi. Prav te tehnološke spremembe so v veliki meri spodbudile razvoj novih storitev telekomunikacijskih sistemov, ki segajo od začetne izmenjave telefonskih in telegrafskih sporočil do današnje integrirane izmenjave digitalnih podatkov v obliki govora, zvoka, besedila, slik ali videovsebin (Vidmar, 2002, str. 26-45).

Dinamičen razvoj informacijsko komunikacijske tehnologije v navezavi s temeljnimi spremembami poslovnih modelov in načinov poslovanja spreminja tudi vlogo in pomen informatizacije v podjetjih. To vpliva tako na preoblikovanje procesov, ki jih izvajajo podjetja, kot tudi na organiziranost, skozi katero podjetja izvajajo osnovno dejavnost in povezane podporne dejavnosti.

V tržno reguliranem poslovnem okolju si podjetja neprestano prizadevajo vzpostavljati konkurenčno prednost pred drugimi. Razvijanje in prilagajanje poslovne strategije je eden od najpomembnejših vzvodov vodstva podjetja pri iskanju konkurenčnih prednosti (Kovačič, Bosilj-Vukšić, 2005, str. 17-21). Uresničevanje poslovne strategije poteka skozi poslovanje in skozi izvedbo poslovnih procesov podjetja. Ob prilagajanju poslovnih strategij pogosto prihaja do spreminjanja poslovnih procesov. Ta zajema njihovo racionalizacijo in standardizacijo, običajno pa jih spremljajo tudi organizacijske spremembe. Proces prilagajanja poslovne strategije spreminja procese poslovanja podjetja, kar se običajno odraža tudi na njihovi informatizaciji.

Informatizacija poslovanja je sprva predstavljala predvsem zbiranje, klasifikacijo in hranjenje podatkov. Pomen informatizacije za podporo izvedbe poslovnih procesov se je povečeval z razvojem informacijske tehnologije. Avtomatizacija poslovnih procesov je izboljševala njihovo izvedbeno učinkovitost. Sčasoma se je pokazalo, da razvoj informacijske tehnologije poleg avtomatizacije procesov omogoča tudi drugačne načine izvedbe dela od do tedaj uveljavljenih. Danes nam to dokazuje svetovni splet kot infrastruktura, ki uvaja obdobje E-poslovanja, to nam dokazujejo tudi celovite programske rešitve, poimenovane s kratico ERP (angleško: Enterprise Resource Planning). Predstavljajo lupino, znotraj katere se izvajajo vsi poslovni procesi organizacije.

Lahko rečemo, da je razvoj informacijsko komunikacijske tehnologije močno vplival na spreminjanje poslovanja podjetij in je prerasel v gibalo sprememb, ki sili vodstva podjetij v ponovno preučevanje poslovnih strategij. Če v strateških usmeritvah niso prisiljeni spreminjati današnjega poslanstva svojih podjetij, so prav gotovo pod velikim pritiskom ob strateškem opredeljevanju prihodnje vizije delovanja. Možnosti uporabe informacijske tehnologije so za posamezne dejavnosti različne, vendar se kaže, da so v iskanju konkurenčne prednosti podjetja

čedalje bolj prisiljena vključevati v svojo poslovno strategijo tudi strateško načrtovanje informatike (Kovačič, Bosilj-Vukšić, 2005, str. 236).

## 3.1 Strateško načrtovanje informatike

Poslovna uspešnost podjetij postaja čedalje bolj odvisna od informatizacije v podjetju, zato postaja strateško načrtovanje informatike čedalje bolj nujen sestavni del poslovnega načrtovanja. Informatizacija poslovnih procesov se izkazuje kot pomemben element konkurenčne prednosti, zato mora biti njihovo načrtovanje tesno povezano in usklajeno s cilji organizacije. Ob strateškem načrtovanju informatike se je koristno zavedati pomena vzajemne usklajenosti poslovnega in tehnološkega vidika informatizacije poslovanja. To pomeni, da izvedbo poslovnih procesov obravnavamo tudi s stališča priložnosti, ki jih ponuja informacijsko komunikacijska tehnologija. Strateško načrtovanje informatike, ki izhaja iz takšnega poslovnega načrtovanja, kjer se informatika zgolj prilagaja poslovni strategiji, ne more imeti vpliva na uspešnost in konkurenčnost poslovanja (Kovačič, Bosilj-Vukšić, 2005, str. 233).

S strateškim načrtom informatike, ki bo skladen s poslovno strategijo podjetja, usmerjamo razvoj informatizacije na osnovi opredeljenih ciljev, poslovnih pravil, dejavnikov uspeha, informacijskih potreb poslovnih procesov in možnosti IKT. V strateškem načrtu poslovne procese predstavimo kot abstraktne modele programskih rešitev, njihove informacijske potrebe pa kot modele podatkov. Upoštevati moramo, da je informacijska tehnologija zaradi svoje omejene razpoložljivosti, prilagodljivosti in stroškov uvajanja lahko tudi omejevalni dejavnik načrtovanja. Izvedljivost strateškega načrta informatike je večja ob upoštevanju tehnoloških možnosti ter ob sočasnem načrtovanju procesov in podatkov. Takšen način strateškega načrtovanja informatike zahteva proste roke pri spreminjanju in prilagajanju poslovnih procesov. Podjetja ga lahko najlažje uresničijo skozi prenovo poslovnih procesov.

S strateškim načrtovanjem informatike, ki upošteva skladnost med poslovnim in tehnološkim vidikom informatizacije, vzpostavljamo stično točko med menedžmentom in uporabo informacijsko komunikacijske tehnologije. V takšnem primeru lahko sega strateško načrtovanje informatike do nivoja oblikovanja poslovne strategije.

Pri oblikovanju strategij razvoja informacijskih sistemov in izvedbi informacijske podpore je dobrodošlo poznavanje trendov njihovega spreminjanja in pogled na spremembe, ki se dogajajo v širšem okolju. Med spremembe, ki jih poganja tehnološki napredek in posledično vplivajo na zgradbo ter izvedbo informacijske podpore v organizacijah, štejemo trende, kot so (Gradišar, Resinovič, 2001, str. 415):

- reinženiring,
- sestopanje (angleško: downsizing),
- organizacijsko izločanje (angleško: outsorcing),
- decentralizacijo.

Omenjeni trendi lahko vplivajo tudi na oblikovanje razvojne strategije v organiziranosti in delovanju službe za informatiko ter načrtovanje obsega in strokovnega profila kadrov za izvedbo tega funkcijskega področja v organizaciji.

Če strategijo razvoja informatike vzpostavljamo zgolj na nivoju njene trenutne podporne funkcije v podjetju, imamo manj manevrskega prostora.

V takšnih pogojih bo strateški načrti najverjetneje bolj prilagojen že vzpostavljeni poslovni ali korporacijski strategiji delovanja podjetja. Bolj bo izhajal iz analize dosežene stopnje informatizacije, iskanja neizkoriščenih priložnosti, razpoložljivih kadrovskih virov in analize učinkovitosti organiziranosti in delovanja. V ospredju strateškega načrta bodo verjetno bolj izstopale smeri razvoja na organizacijskem, kadrovskem, izobraževalnem in komunikacijskem nivoju delovanja službe za informatiko.

## 3.2 Poslovni procesi, poslovno in informacijsko modeliranje

Ena od pogosto omenjenih možnosti strateškega načrtovanja razvoja informatike se navezuje na izvedbo poslovnih procesov oziroma danes čedalje pogosteje na njihovo prenovo, zato si zasluži malo podrobnejšo obravnavo. Poslovni proces je niz koordiniranih opravil in aktivnosti, ki jih izvajajo tako ljudje kot tehnična oprema z namenom, da dosežejo zastavljen cilj, je definicija, ki jo za poslovni proces navaja neprofitna organizacija za standardizacijo splošnih poslovnih procesov BPMI (angleško: Business Process Management Initiative). Poslovni proces je aktivnost, ki pretvori fizično substanco ali podatek iz enega stanja v drugega (Hay, 2006, str.133).

Obstaja mnogo opredelitev poslovnega procesa, najpogosteje pa v zvezi s poslovnim procesom srečujemo naslednjo trditev: poslovni proces je niz povezanih aktivnosti, katerih izvedba povzroči preoblikovanje vhodnih sestavin in uporabno vrednost za izbranega akterja, običajno je rezultat izvedbe procesa izdelek ali storitev. Aktivnost je najmanjši del procesa, ki ne vključuje zahteve po odločanju in je zato ni potrebno nadalje razgrajevati. Aktivnosti znotraj procesa izrabljajo vire in proizvajajo rezultate.

Eden prvih, ki je tlakoval obravnavanje dela kot proces, je bil Adam Smith, ki se je v 18. stoletju ukvarjal z opisovanjem izvedbe dela. Iz njegovih preučevanj izhaja znamenita delitev dela, ki je vplivala na vrsto teorij in konceptov o izvedbi, ravnanju in kontroli izvedbe dela. Koncept poslovnih procesov, kot se odraža v njihovem današnjem pojmovanju, izhaja iz spoznanj Fredericka W. Taylorja, ki je v začetku 20. stoletja podal osnovne principe znanstvenega menedžmenta. Iz korenin njegovih spoznanj izhajajo današnji pogledi na poslovne procese in izvajanje njihovega preoblikovanja.

Poslovni procesi so eden od mehanizmov, s pomočjo katerih izvajamo delitev dela za zagotavljanje proizvodov ali storitev z dodano vrednostjo. Preučevanje poslovnih procesov sloni na njihovih značilnostih, kamor sodijo vhodno-izhodne količine, zaporedje izvedenih aktivnosti, potrošnik, ki mu je poslovni proces namenjen, in dodana vrednost za potrošnika. Značilnost poslovnega procesa je vpetost v organizacijsko strukturo ter zmožnost medsebojnega povezovanja različnih funkcionalnosti, ki jih izvajamo v organizaciji. Med značilnosti, s katerimi opredeljujemo poslovni proces, so najpogosteje uvrščene naslednje lastnosti (Kovačič, Bosilj-Vukšić, 2005, str. 180):

- ima cilj,
- ima specifičen vhod,
- ima specifičen izhod,
- uporablja vire,
- ima lastnika,
- ima omejitve,
- potrebuje čas za izvedbo,
- vsebuje vrsto aktivnosti, ki se izvajajo v nekem zaporedju,

- vsebuje pravila za izvedbo aktivnosti,
- lahko vpliva na eno ali več organizacijskih enot,
- ima uporabno vrednost za akterja (stranko), ki je lahko notranji ali zunanji,
- ima vgrajeno ravnanje v primeru neskladnosti,
- ima merljive značilnosti, ki omogočajo oceno učinkovitosti.

Z namenom njihove boljše preglednosti poslovne procese običajno predstavljamo na abstraktnem nivoju v obliki procesnih modelov.

### 3.2.1 Modeli, poslovni modeli, modeliranje poslovnih procesov

Modele uporabljamo kot pripomočke pri izvedbi različnih nalog. So komunikacijsko sredstvo in način za predstavitev obravnavanega problema, ki ga lahko preslikamo v drugo obliko brez izgube podrobnosti. Omogočajo predstavitev notranjih mehanizmov zgradbe ali delovanja opazovanih sistemov. Modeli so lahko besedni, analitični (matematični), slikovni (shematični) ali simulacijski (Avison, Fitzgerald, 1996, str. 452.). Modeli so abstraktne preslikave realnega sveta. Predstavljajo sredstvo za ponazoritev naše predstave o stvarnosti. Običajno jih namenoma oblikujemo tako, da predstavljajo realnost iz določenega zornega kota, ki je za nas ali druge opazovalce posebej zanimiv. Modeliranje običajno izvajamo na zanemarjanju nebistvenih podrobnosti.

Poslovni model je opredeljen kot model delovanja podjetja v okolju. Omogoča pregled poslovnih procesov, s katerimi izvajamo izbrano poslovno strategijo. Na njegovi osnovi lahko spremljamo potreben obseg podatkov in navodil, ki jih uporabljajo izvajalci procesov na različnih nivojih podjetja. S poslovnim modelom abstraktno predstavimo poslovanje organizacije. Poslovni model služi različnim vidikom uporabe:

- poslovnemu,
- procesnemu,
- izvedbenemu.

V tem pogledu ga lahko uporabljamo za različne namene upravljanja in usmerjanja razvoja podjetja (Kovačič, Bosilj-Vukšić, 2005, str. 21-22).

Modeliranje poslovnih procesov (angleško: business process modelling, tudi business proces discovery) uporabljamo za predstavitev razumevanja aktivnosti in podatkov, ki jih podjetje uporablja za doseganje poslovnih ciljev. Modeliranje uporabljamo tudi pri načrtovanju preoblikovanja poslovnih procesov. Model, ki prikazuje stanje, imenujemo izhodiščni model, prenovljeni model predstavlja izvajanje aktivnosti po preoblikovanju poslovnih procesov. Modeli poslovnih procesov so ključnega pomena pri analizah procesov, pri odločanju o razvoju ali prenovi procesov, pri razvoju programske opreme, pri uvajanju celovitih programskih rešitev in pri načrtovanju integracije procesov različnih podjetij. Omogočajo (Kovačič, Bosilj-Vukšić, 2005, str. 177-178):

- pregled nad poslovanjem,
- razumevanje povezanosti poslovnih procesov,
- odkrivanje slabosti v izvajanju procesov,
- preizkušanje pred uveljavljanjem v praksi,
- odkrivanje informacijskih potreb izvajalcev procesa.

Na nivoju izgradnje ali obnove informacijskih sistemov je uporaba poslovnega modela eden od osnovnih pripomočkov za razumevanje vsebine sistema. Omogoča razumevanje poslovnih procesov organizacije, ki jih želimo informacijsko podpreti. Nekateri avtorji zagovarjajo uporabnost koncepta in tehnike poslovnega modeliranja tudi v primerih, ko obravnavani sistem nima nobenega opravka z opravili, ki jih poimenujemo poslovna dejavnost. Modeliranje izvajamo do stopnje razumevanja vsebine, ki jo izpeljujemo iz razumevanja delovanja (funkcioniranja) poljubnega obravnavanega „poslovnega modela“ (Jacobson, Booch, Rumbaugh, 1999c, str. 122).

V dokumentiranje lastnosti poslovnega procesa lahko vključimo več ali manj značilnih parametrov, odvisno od namena modeliranja. V odvisnosti od namena lahko izbiramo stopnjo abstrakcije ali dodatni obseg lastnosti poslovnega procesa, ki ga spremljamo oziroma dokumentiramo. V tem pogledu izhajamo iz zahtev naše ključne aktivnosti, ki je spodbudila zahtevo po modeliranju poslovnih procesov. Zahteve po optimizaciji poslovne odličnosti izvedbe procesa verjetno terjajo dokumentiranje drugačnega nabora parametrov poslovnega procesa kot načrtovanje njegove informacijske podpore.

Koncept „poslovnega modeliranja“ ter pazljiva izbira in dokumentiranje ustreznih (ključnih) parametrov v izvedbi poslovnega procesa omogoča vzpostavitev stičnih točk z drugimi modeli, s katerimi ponazarjamo drugačne poglede na izvedbo poslovnih procesov.

### 3.2.2 Podatkovno modeliranje

Podatkovno modeliranje je del načrtovanja informacijskih sistemov, ki se ukvarja s predstavitvijo entitet in njihovih lastnosti v obravnavanem (poslovnem) sistemu. Ena od definicij pravi, da je podatkovno modeliranje vzpostavljanje razumevanja podatkov realnega sveta in njihovih medsebojnih povezav (Grad, Jaklič, 1996, str. 14). Objavljeni standard za sintakso in semantiko konceptualnega modelirnega jezika opredeljuje podatkovni model kot grafično in tekstno predstavitev analize, s katero predstavimo podatke, potrebne organizaciji za doseganje njenega poslanstva, ciljev in funkcij. Podatkovni model identificira entitete, domeno (atribute) in medsebojne relacije med podatki (Std. IEEE 1320.2-1998, str. 25). Podatkovno modeliranje je proces izdelave podatkovnega modela.

Med podatkovnim modeliranjem izvajamo organizacijo in strukturiranje podatkov. Podatkovni model namreč lahko predstavimo na enega od treh načinov (po ANSI, 1975):

- konceptualni (abstraktni) pogled (opredelitev entitetnih razredov, poslovnih entitet se odrazi v modelu entitet-povezav ali semantičnem modelu podatkov),
- logični pogled (izvajamo opis podatkov s tabelami, skupinami atributov, objektnimi razredi; izdelujemo hierarhične, mrežne, relacijske podatkovne modele),
- fizični pogled (oblika hranjenja podatkov).

S podatkovnim modelom zagotavljamo tako abstraktno predstavitev entitet obravnavanega sistema kot njihovih lastnosti, medsebojnih relacij in operacij, ki jih z njimi izvajamo. Za začetno obdobje razvoja računalniške tehnologije so bili značilni modeli, ki so ločeno modelirali podatke in ločeno procese.

V današnjem času na ravni predstavitve entitet v obravnavanem sistemu prevladujejo predmetni modeli. Z njihovo pomočjo združujemo modeliranje podatkov in procesov.

Pri predmetnem modeliranju vsak objekt realnega sveta predstavimo s predmetom v modelu. Značilnost predmetnih modelov je predstavitev vzajemnega delovanja predmetov (objektov), ki združujejo tako podatke kot možne operacije nad temi podatki. Na ta način predmetni model vključuje tako znanje predmetov kot aktivnosti na nivoju abstrakcije njihove odgovornosti (Std. IEEE 1320.2-1998, str. 17).

V obravnavi definicij in upravljanju predmetov v porazdeljenih informacijskih sistemih lahko naletimo na prepletanje pojmov informacijski model in podatkovni model (Pras, Schoenwaeldner, 2003). Zaradi te nejasnosti je bil podan predlog, da se pojem informacijski model uporablja za modeliranje na bolj abstraktnem (konceptualnem) nivoju kot podatkovni model. Osnovni namen informacijskih modelov naj bi bil formalno predstavljanje problemske domene z navedbo relacij med objekti brez navajanja podrobnosti konkretne realizacije. Po drugi strani pa bi podatkovni modeli vsebovali več podrobnosti (nižja stopnja abstrakcije). Skladno s predlogom naj bi bil namen podatkovnega modela opredelitev podrobnosti v izvedbi računalniških rešitev (Pras, Schoenwaeldner, 2003).

V splošnem lahko naletimo na mnogo predstavitev informacijskega (podatkovnega) modela. Vse te predstavitve imenujemo podatkovni modeli, ne glede na to, ali gre za objektne modele, entitetno relacijske modele ali XML sheme.

S stališča podporne vloge, ki jo ima informacijska tehnologija pri izvajanju poslovnih procesov, želimo s pomočjo podatkovnega modeliranja vzpostaviti povezavo s poslovnim modeliranjem. Med poslovnim in podatkovnim modelom torej obstaja velika potreba njune skladnosti in povezljivosti. Nekateri avtorji opozarjajo na problematiko neskladja med obema oblikama modeliranja in pozivajo na zapolnjevanje manjkajoče povezave med modeloma (Kovačič, Bosilj-Vukšić, 2005, str. 255).

Ob upoštevanju obravnavanih definicij in značilnosti predmetnega modeliranja lahko rečemo, da ocenjeno nesoglasje v povezovanju obeh vrst modeliranja lahko zapolnimo s predmetnim (konceptualnim) modeliranjem, ki temelji na abstraktni predstavitvi medsebojnega delovanja entitet in vgrajenih poslovnih pravil, na osnovi katerih poteka izvajanje poslovnega procesa.

## 3.3 Načrtovanje razvoja računalniških rešitev

Sodobno delovanje informacijskih sistemov temelji na povezanem in vzajemnem delovanju ljudi in naprav, ki jih pojmujemo kot informacijsko komunikacijsko infrastrukturo. Delovanje sistema sloni na računalniških rešitvah za podporo izvedbe procesov, ki uporabljajo razpoložljive vire sistema. S podrobnejšim načrtovanjem novih računalniških rešitev uresničujemo cilje, ki običajno izhajajo iz smernic strateškega načrtovanja.

Izkušnje so pokazale, da na uspešnost projektov razvoja informacijskega sistema ter konkretnih računalniških rešitev vplivata ustreznost in kakovost njihovega načrtovanja. V projektih razvoja računalniških rešitev se kot dejavniki tveganja najpogosteje navajajo kompleksnost, spremenljivost in odvisnost od uporabljene tehnologije. Zavedanje o pomenu načrtovanja za uspešno izvedbo procesa informatizacije med strokovnjaki že dolgo spodbuja nastanek in razvoj različnih metod in tehnik načrtovanja ter izvajanja projektov izdelave računalniških rešitev (Bygstad, Munkvold, 2003, str. 3). Želja po nadzorovanju kompleksnih projektov izdelave računalniških rešitev je pripeljala do formalizacije razvojnega procesa, ki se neprestano prilagaja spreminjajočim tehnološkim možnostim.

Metode načrtovanja in razvoja informacijskih sistemov ter podpornih računalniških rešitev so se začele razvijati v času razvoja informacijskega inženirstva. Informacijsko inženirstvo, kot ga je zasnoval James Martin, se je uveljavilo v osemdesetih letih prejšnjega stoletja. Predpisovalo je strog pristop k analizi, načrtovanju in izvedbi računalniških rešitev. Slonelo je na izvajanju funkcij organizacije, definiciji procesov in potrebnih podatkih. Poslovno znanje, potrebno za izvedbo računalniških rešitev, se je vgrajevalo na osnovi intervjujev uporabnikov. Osnove načrtovanja so slonele na analizi pretoka podatkov ter tehnikah načrtovanja podatkovnih zbirk, ki so jih vzdrževali podatkovni administratorji in sistemski analitiki.

Informacijsko inženirstvo je od začetnih oblik delovanja doživljalo preobrazbo v bolj celovite pristope načrtovanja informacijskih sistemov. To je vključilo preučevanje prepletenosti organizacijskih procesov, delovanja kadrov in organizacijskih enot, usmerjenih v uresničevanje ciljev organizacije, skladno z njenimi strateškimi usmeritvami. Možnosti informacijsko komunikacijske tehnologije in trend iskanja konkurenčnih prednosti so pripeljali do zahtev po prenovi poslovnih procesov. Vzpostavljeni so bili novi izvirni načini usklajevanja dela, podatkov in znanja.

Razvoj informacijskih sistemov je zadnjih nekaj let usmerjen k celovitim programskim rešitvam ERP, k sprotni obdelavi transakcij OLTP (angleško: online transaction processing), k razvoju aplikacij za integracijo obstoječe programske opreme, razvoju aplikacij za podporo E-poslovanja, razvoju informacijskih sistemov za upravljanje (angleško: management information systems) ter razvoju informacijskih sistemov za direktorje (angleško: executive information systems).

Razvoj celovitega informacijskega sistema v podjetju postaja čedalje bolj kompleksen proces. Vključuje razreševanje tehnoloških problemov, povezanih z uporabljeno informacijsko arhitekturo in njenimi komponentami ter tudi razreševanje organizacijskih in socialnih problemov, povezanih z vsebino informacijskega sistema podjetja, ki ga razvijamo.

Dobro razumevanje vsebine in zahtev informacijskega sistema predstavlja enega od kritičnih faktorjev uspeha pri razvoju sistema. Prepoznavanje zahtev sistema predstavlja začetno stopnjo v procesu njegovega razvoja. Med vire za prepoznavanje zahtev sistema prištevamo strokovni kader, ki izvaja proces informatizacije, in izvajalce poslovnih procesov podjetja oziroma uporabnike sistema. V začetni stopnji procesa informatizacije skušajo oboji, na osnovi vzajemnega sodelovanja, izdelati seznam zahtev sistema oziroma model delovanja, ki bo prikazoval tako njegovo funkcionalnost kot tehnične specifikacije oziroma omejitve, ki vplivajo na njegovo izvedbo. Na tej osnovi lahko ugotovimo, da poslovno modeliranje postaja čedalje bolj nujni sestavni del načrtovanja informacijskega sistema.

Tudi nekateri avtorji (Avison in Fitzgerald, 2003, Kruchten, 1999) opozarjajo na velik pomen izdelave poslovnega modela pred opredelitvijo zahtev informacijskega sistema. Avtorji utemeljujejo, naj bi prav poslovni model omogočal bolj celovito razumevanje vsebine informacijskega sistema in funkcijskih zahtev, ki jih mora podpreti.

Po mnenju nekaterih je poslovni model ključno konceptualno orodje, ki omogoča zajem, izmenjavo in oblikovanje skupne vizije poslovnega delovanja med nosilci poslovnih procesov in nosilci informacijske podpore v podjetju. S poslovnim modelom ustvarimo skupni jezik in vzajemno razumevanje udeležencev v razvoju sistema (Osterwalder, Pigneur, Tucci, 2005, str. 28). Pomen zgodnjega poslovnega modeliranja in povezovanja z informacijskim (podatkovnim) modeliranjem je izpostavljen tudi v nekaterih metodologijah razvoja informacijskih sistemov.

Med tiste, ki izpostavljajo poslovno modeliranje kot sestavni del procesa razvoja informacijskega sistema in programske opreme, sodijo Business Engineeering with Object Technology (Taylor, 1995), Watch (Montilva et al. 2000), MERISE, Mainstream Objects (Yourdon et al. 1995), OOAD (Martin, Odell, 1992), Business Modeling with UML (Eriksson, Penker, 2000), OPEN Process Framework (Firesmith, Henderson-Sellers, 2000), Rational Unified Process (Kruchten, 1999), Enterprise Modeling with UML (Marshall, 2000), TAD - Tabular Application Development („Tabelarni razvoj aplikacij“, Damij, 2001).

Kot rešitev in odgovor na iskanje manjkajoče povezave med nekdaj tradicionalnim entitetno-relacijskim informacijskim modeliranjem in sledenju razvijajočim kompleksnim poslovnim procesom je bila načrtovalcem ponujena metoda objektnega modeliranja vlog ORM (angleško: object role modeling). ORM predstavlja metodo za načrtovanje podatkovnih modelov na konceptualnem nivoju, kjer aplikacijo namesto z uporabo preoblikovanih oziroma izvedbenih podatkovnih struktur opisujemo na konceptualnem nivoju z izrazi, ki jih razume uporabnik (Halpin, 1998). Metoda omogoča preprostejše komuniciranje med strokovnjaki poslovne domene in strokovnjaki informacijskega modeliranja. Objektno modeliranje vlog se uporablja za konceptualno modeliranje, kjer izkoriščamo prednosti preprostosti uporabe naravnega jezika za postopek analize preprostih enot informacij ter primerov. Model, ki ga s tem dobimo, omogoča nadaljnjo izdelavo pogledov, bodisi entiteno-relacijskih bodisi objektno orientiranih [URL: http://www.orm.net/pdf/dppd.pdf].

Današnje načrtovanje in izvedba projektov razvoja računalniških rešitev so formalno podprti s številnimi metodologijami. Večina sodobnih metodologij temelji na abstraktni obravnavi sistema, za katerega izdelujemo rešitev. Med abstraktnimi koncepti razgradnje zelo pogosto naletimo na zahteve po predstavitvi izvedbe poslovnih procesov. Tipični koncepti, na osnovi katerih poteka načrtovanje končnih računalniških rešitev, so poslovni procesi, naloge (aktivnosti), akterji, poslovna pravila, poslovni objekti in dogodki. Modeli predstavitve izvedbe poslovnih procesov so v procesu razvoja prevedeni na elemente (sestavine) ciljnih računalniških rešitev. Modeliranje na različnih konceptualnih nivojih je povezano z objektnim modeliranjem, s katerim predstavljamo entitete realnega sveta in njihove medsebojne odnose. Razvoj računalniške rešitve sloni na vzajemnem sodelovanju številnih udeležencev (Bruegge, Dutoit, 2004, str. 10). Načrtovanje razvoja računalniških rešitev zato sloni na modeliranju različnih pogledov na delovanje sistema. Razumevanje med udeleženci na osnovi modelov in izdelanih pripomočkov, načrtovanje in izvedba delnih rešitev ter spremljanje izdelave računalniških rešitev skozi različne razvojne faze pa se uvršča med kritične faktorje uspešnosti v projektih razvoja računalniških rešitev.

## 3.4 Arhitektura informacijskega sistema

Arhitektura informacijskega sistema predstavlja v snovanju informacijskega sistema celovit pogled na zasnovo delovanja informacijskega sistema. Njen namen je predstavitev komponent, ki nastopajo v informacijskem sistemu, njihovo povezavo in delovanje. Tehnološki del te arhitekture zajema načrtovanje opreme (računalniške, komunikacijske in programske), potrebne za delovanje sistema. Opredeljuje tudi lokacijo opreme, njenih uporabnikov, dostopanje do podatkov in vzajemno povezanost teh komponent. Arhitektura sistema predstavlja abstraktno predstavitev delovanja sistema na ravni strateške zasnove, ki omogoča vpogled v podsisteme in sočasno delovanje razvijalcev podsistemov ter njihovega sodelovanja (Bruegge, Dutoit, 2004, str. 734).

Za osnovo predstavitve arhitekture informacijskega sistema lahko izberemo enega od arhitekturnih vzorcev. Med danes najpogostejše arhitekturne vzorce štejemo (Wang, Fung, 2004) objektno orientirano arhitekturo (angleško: Object-oriented architecture), na komponentah zasnovano arhitekturo (angleško: Component-based architecture) in storitveno zasnovano arhitekturo (angleško: Service-based architecture).

S pomočjo arhitekture informacijskega sistema predstavimo globalno sliko komponent, ki tvorijo informacijski sistem. Arhitektura informacijskega sistema predstavlja sestavni del korporacijske arhitekture (angleško: Enterprise Architecture). Ta je v svojem bistvu celovit poslovni model delovanja podjetja v okolju (procesi, informacijski sistem, kadrovski resursi, organizacijske enote, poslovanje organizacije).

V kontekstu razvoja informacijskih sistemov predstavlja arhitektura informacijskega sistema referenčni okvir, znotraj katerega načrtujemo razvoj komponent (sestavnih delov), ki bodo v tehnološkem smislu združljive ter ga bo mogoče na osnovi izbrane arhitekture dopolnjevati, vzdrževati ali nadomeščati.

## 3.5 Vloga informatike, organiziranost službe, kadri

Temeljna vloga informatike v organizaciji je podrejena zagotavljanju delovanja njenega informacijskega sistema. Nosilci izvajanja informatike v organizaciji so usposobljeni kadri, ki lahko organizirano delujejo bodisi znotraj bodisi zunaj nje.

Vloga informatike in organiziranost službe informatike je praviloma odvisna od razumevanja njenega strateškega pomena za podjetje. Zgodovinsko gledano je služba za informatiko prehajala skozi različne organizacijske oblike. Sodobni pogledi vlogo informatike v podjetju razširjajo na zagotavljanje pomoči menedžmentu pri poslovnem odločanju, na razvoj informacijske arhitekture ter na povezovanje razvoja informatizacije s poslovno strategijo podjetja.

Z napredkom informacijske tehnologije so povezani trendi, kot so reinženiring, sestopanje, organizacijsko izločanje in decentralizacija. Vplivajo na preoblikovanje organiziranosti službe za informatiko ali celo njeno izločanje iz organizacije. Sestopanje predstavlja prenos računalniških rešitev z velikih računalnikov na cenejše in prilagodljive mreže manjših. Organizacijsko izločanje je prenos računalniških obdelav na zunanje specializirane pogodbene organizacije. Decentralizacija predstavlja sistem dislociranih in med seboj povezanih informacijskih podsistemov v organizaciji (Gradišar, Resinovič, 2001, str. 415-416).
Službo za informatiko lahko obravnavamo bodisi s stroškovne bodisi prihodkovne strani. V nekaterih okoljih je delovanje službe za informatiko organizirano pod okriljem profitnega centra, ki ponuja storitve tudi na trgu. Za okolja, ki stremijo k zniževanju stroškov, je bolj značilno zunanje izvajanje informatizacije ali posameznih aplikacij ter najemanje strojne in programske opreme.

Vloga kadrov informatike je odvisna od obsega njihovega znanja. Od sodobnega informatika so danes pričakovana interdisciplinarna znanja s področja poznavanja informacijske tehnologije, podjetniških ter komunikacijskih znanj. Čas zmotnega prepričanja, da za uspešen razvoj informacijskega sistema podjetja zadostuje znanje poznavanja računalniške tehnologije in programiranja, je minil (Kovačič, Bosilj-Vukšić, 2005, str. 250).Hitro spreminjanje informacijske tehnologije zahteva od kadrov, zaposlenih na področju informatike, neprestano prilagajanje.

Podjetja skušajo z ustreznim načrtovanjem in razvojem poslovne informatike trajno zagotavljati predvsem pridobivanje novega znanja, njegovo kroženje in medsebojno izmenjavo. Rast produktivnosti snujejo na prilagajanju organiziranosti in na osnovi obstoječih kadrovskih potencialov podjetja.

## 3.6 Zagotavljanje informacijskih storitev, dileme notranje ali zunanje izvedbe

Področje informatizacije je bilo v preteklosti pogosto obravnavano kot servisna dejavnost znotraj podjetja. Ta oblika organiziranosti je bila sčasoma prepoznana kot okorna. Spremenjeni pogledi vodilnega menedžmenta na informatiko in njeno strateško vlogo v poslovanju podjetja spodbujajo reorganizacijo centrov za informatiko.

Ena od predlaganih organizacijskih oblik, primernih tudi za podjetja, ki delujejo geografsko razpršeno, je služba za informatiko, znotraj katere je vzpostavljenih več funkcionalnih delovnih področij (Kovačič, Bosilj-Vukšić, 2005, str. 253):

- načrtovanje razvoja,
- razvoj programskih rešitev,
- pomoč in storitve uporabnikom,
- računalniška operativa za informatizacijo poslovnih enot.

V nekaterih okoljih je pogosta dilema o notranjem ali zunanjem izvajanju informatizacije, ki se običajno pojavlja pri presoji stroškov oziroma učinkovitosti lastne službe za informatiko. V odvisnosti od obsega storitev informatizacije, ki jih kupujemo oziroma si jih zagotavljamo sami, je treba celovito oceniti stroške na naslednjih področjih:

- vzdrževanje infrastrukture (računalniška oprema, omrežja),
- vlaganje v znanje kadrov za informatiko,
- vlaganje v razvoj novih programskih rešitev,
- vzdrževanje obstoječih programskih rešitev,
- zagotavljanje pomoči uporabnikom,
- zagotavljanje varnosti, zanesljivosti in kakovosti delovanja informatike v podjetju.

Po raziskavah se številna podjetja pri nas spopadajo s pomanjkanjem usposobljenih informatikov. Veliko majhnih in srednje velikih podjetij sploh nima posebnih oddelkov, ampak le odgovorno osebo. Takšna podjetja običajno najemajo zunanje izvajalce, pogosto samo posamične strokovnjake, vendar se ob tem srečujejo s precejšnjimi težavami pri vrednotenju in nadzorom nad projekti (Stare, Bučar, 2005, str. 141).

Ankete in raziskave so pokazale, da je za uspešno izvedbo projektov uvajanja in prenove informatizacije v podjetju, za odločanje o zunanjih izvajalcih, ter za odločanje o izboru dobaviteljev opreme in storitev informacijske tehnologije velikega pomena dobra komunikacija med menedžerji in strokovnjaki s področja informacijske tehnologije (Stare, Bučar, 2005). Splošno znanje informatikov o poslovnih procesih in organizaciji ter poznavanje poslovnih procesov podjetja pa naj bi bilo komunikacijsko ogrodje ključnega pomena za prenos znanja in reševanje problemov skupaj z uporabniki informacijske tehnologije.

Ugotovitve, da je za pomoč zunanjih izvajalcev ključna uspešna komunikacija med zunanjimi izvajalci in nosilci projekta znotraj podjetja, dopuščajo sklepanje o velikem pomenu organizacije projekta, na katerem sloni medsebojno sodelovanje. To velja tudi za projekt izdelave računalniške rešitve, ki bi moral, že v osnovi, celovito in hkrati podrobno opredeliti želen obseg funkcionalnosti in kakovosti končne rešitve. Takšen pristop običajno terja veliko časa in vloženega dela. V pogostem iskanju zmanjševanja stroškov razvoja je zaznan trend opuščanja takšnih pristopov (agilni razvoj računalniških rešitev). Preučevanje primerov klasičnih metodologij razvoja kaže na velik pomen ključnih oseb, ki znotraj organizacije sodelujejo na projektu razvoja novih računalniških rešitev. Njihova odločilna znanja so poznavanje poslovnih procesov, poznavanje metod zbiranja zahtev uporabnikov, analize in modeliranja delovanja sistema, zmožnost odkrivanja napak, beleženje ter prenašanje in povezovanje znanja med sodelavci in posredovanje le-tega na zunanje izvajalce. Odsotnost mehanizmov za spremljanje poteka dela zunanjih izvajalcev (projekt) in posrednikov komunikacije (pripomočki in lastno osebje) lahko uvrstimo med dejavnike tveganja pri najemanju storitev zunanjih izvajalcev.

V nekaterih okoliščinah lahko opiranje na pomoč zunanjih izvajalcev obravnavamo kot strateško aktivnost za hitrejše pridobivanje manjkajočega tehnološkega in drugega potrebnega znanja pri razvoju novih računalniških rešitev. S primerno kombinacijo naročanja novih (praviloma nestandardnih) računalniških rešitev in šolanjem za njihovo vzdrževanje (nadgradnjo), v navezi z izvajalcem, je možno načrtno pridobiti nekatera znanja. Če izvajalec pristane na takšen način trženja svojega znanja, lahko pričakujemo znaten pospešek v procesu pridobivanja novih znanj, ki jih potrebujemo za razvoj lastnih računalniških rešitev, pa smo jim v preteklosti namenili premalo pozornosti.

Med prednosti, ki jih prinaša pomoč zunanjih izvajalcev, bi načeloma lahko uvrščali večji vnos sodobnega tehnološkega znanja, prenos preverjenih sistemov in tudi dobro prakso izvedbe razvojnih projektov. Temu seveda ne bo tako, če izvajalcu služimo kot učni poligon. Med tveganja zunanjega izvajanja informacijskih storitev lahko uvrstimo tudi odvisnost od enega izvajalca in izgubljanje neodvisnosti strokovne zasnove izvedbe procesov iz domene delovanja. Po drugi strani opiranje na več zunanjih izvajalcev ali njihova pogosta menjava prinaša tveganje, povezano s spreminjanjem arhitekture informacijskega sistema, vnašanje nezdružljivih ali slabo združljivih delnih rešitev in odvisnost od različnih tehnologij, za katere moramo zagotavljati podporo. Med tveganja lahko uvrstimo tudi vnašanje komponent, izvorno izdelanih za poslovni model drugega uporabnika, in njihovo prilagajanje našim potrebam. Takšen pristop je lahko grožnja poznejšim nadgradnjam ali zahtevam po prilagoditvi našim specifičnim potrebam.

Zunanja izvedba storitev informatizacije je lahko dejavnik vzajemne koristi tako za naročnika kot izvajalca. Vendar bi to oceno lažje podal za profitno delovno okolje. Za nekatere neprofitne organizacije se je zaradi načina financiranja koristno izogibati odvisnosti, ki se lahko izrodi iz začetne simbioze. V takšni spregi lahko pričakujemo, da bomo storitve izvajalca plačevali dražje kot izvedbo storitev v lastni režiji. Na dolgi rok pa je vprašljivo tudi lastno snovanje strategije informatike in nadaljnja usoda sprva ohranjenih lastnih informacijskih storitev.

Razreševanje dileme o notranjem ali zunanjem izvajanju informatizacije oziroma o prenašanju funkcij službe za informatiko na zunanje izvajalce zahteva premišljeno tehtanje koristi in tveganj. Odločitev lahko izhaja iz pogleda, ki ga ima vodilni menedžment na strateško vlogo informatike v podjetju, kar zahteva upoštevanje različnih dejavnikov.

Zunanja izvedba informacijskih storitev zahteva dober povezovalni člen med naročnikom in izvajalcem ter nekatere nadzorne mehanizme sodelovanja.

Vsaka izvedba informacijskih storitev sloni na povezovanju tehničnega znanja računalniških strokovnjakov in poznavanju izvedbe poslovnih procesov podjetja. Sami lahko na kakovost izvedbe naročenih storitev vplivamo s poznavanjem standardov, vgrajenih v uporabljeno informacijsko tehnologijo, s poznavanjem izvedbe procesov organizacije, s sposobnostjo komuniciranja, z zmožnostjo skupinskega dela ter s poznavanjem in dograjevanjem začrtane arhitekture informacijskega sistema.

## 3.7 Načrtovanje vzpostavitve oziroma prenove informacijske podpore poslovanja

Načrtovanje informacijskega sistema lahko obravnavamo kot formaliziran, podatkovno usmerjen način načrtovanja, opredeljen z modelom delovanja podjetja, njegovimi funkcijami, procesi in uporabljenimi podatki (Kovačič, Bosilj-Vukšić, 2005, str. 255-56).

Pri ugotavljanju informacijskih potreb na ravni delovanja podjetja se lahko poslužujemo različnih metodoloških možnosti (Gradišar, Resinovič, 2001, str. 417; Kovačič, Bosilj-Vukšić, 2005, str. 257):

- metoda ISAC (angleško: Information System Work and Analysis of change) ima korenine na Švedskem in povezuje analizo problemov in podatkov,
- metoda CSF (angleško: Critical Success Factors) skuša informacijske potrebe izpeljati iz ključnih dejavnikov uspeha,
- metoda BSP (angleško: Business System Planning) skuša informacijske potrebe izpeljati iz problemov, njihovih rešitev in nabora potrebnih odločitev,
- metoda BPR (angleško: Business Process Renovation) uvaja ravnanje s spremembami procesov in jih povezuje z uvajanjem ustreznih informacijskih tehnologij,
- metoda BRA (angleško: Rule-based Business Renovation approach) sloni na zasnovi meta-modela poslovnih pravil in vzdrževanju poslovnih pravil v posebnem repozitoriju.

Med mnogimi uspešnimi primeri uvajanja informatizacije v naši praksi je bil uporabljen model prenove poslovanja, temelječ na metodi BRA. Proces načrtovanja, razvoja, informatizacije in prenove poslovanja poteka v iterativnem izvajanju naslednjih faz (Kovačič, Bosilj-Vukšić, 2005, str. 257-58):

- opredeljevanje strateških informacijskih potreb ter strateško modeliranje procesov organizacije in uporabljenih podatkov,
- snovanje arhitekture informacijskega sistema v podjetju,
- izdelavo programa projektov izgradnje informacijskih podsistemov.

Na področju načrtovanja informacijskih sistemov in izvedbe informacijske podpore poslovanju je na voljo še več drugih metod, tehnik in pristopov, in sicer (Gradišar, Resinovič, 2001, str. 438):

- stopnje rasti,
- analiza strategije naložb,
- načrtovanje s pomočjo scenarijev,
- načrtovanje s pomočjo analize povezav,
- modeliranje podjetij.

Strateške informacijske potrebe je najbolje izpeljati iz opredeljene strategije podjetja, saj to predstavlja najbolj učinkovito povezovanje aktivnosti vodilnega menedžmenta v podjetju in aktivnosti, ki jih izvajajo informatiki v podjetju. Na osnovi strateških informacijskih potreb opredelimo arhitekturo informacijskega sistema, ki predstavlja ciljno globalno zgradbo informacijskega sistema podjetja (Kovačič, Bosilj-Vukšić, 2005, str. 261-63). Njen sestavni del je tudi podatkovna in tehnološka arhitektura. Podatki, ki jih uporabljamo v izvajanju procesov podjetja oziroma so izdelek v informacijskem procesu podjetja, predstavljajo enega ključnih podjetniških dejavnikov (vir), ki ga je treba načrtovati in upravljati tako kot ostale vire podjetja. Kot informacijsko tehnologijo obravnavamo opremo (računalniško, programsko, komunikacijsko), ki jo v informacijskem procesu uporabljamo. Načrtovan poslovni model in arhitektura informacijskega sistema predstavljata osnovo za nadaljnji razvoj oziroma uvajanje računalniških rešitev. Ta poteka preko procesa izbiranja primernih obstoječih rešitev oziroma odločanja o izdelavi lastnih (Kovačič, Bosilj-Vukšić, 2005, str. 264).

Za načrtovanje informacijske podpore bi lahko rekli, da strateške informacijske potrebe najprej predstavimo z arhitekturo informacijskega sistema. Z njo predstavimo temeljno zasnovo povezanosti in vzajemnega delovanja informacijske tehnologije, podatkovnih virov in izvajalcev aktivnosti modeliranih poslovnih procesov. Na tej osnovi izvajamo nadaljnje aktivnosti, kot so nakup, dopolnitev, razvoj in uvajanje informacijske podpore v obliki konkretnih računalniških rešitev.

# 4. PREDMETNA PARADIGMA (OO TEHNOLOGIJA)

Računalniški program predstavlja združitev opisa različnih vrst podatkov in zaporedje računskih, logičnih in drugih operacij, ki jih izvaja računalnik (Grad, Dacar, 1996).

## 4.1 Strukturiran pristop proti objektno orientiranemu

Delovanje računalniškega programa lahko sloni na proceduralnem ali objektnem vzorcu izvajanja. Paradigma (vzorec) proceduralne izvedbe sloni na izvedbi zaporednih aktivnosti nad podatki, ki jih izvaja klicana procedura. Proceduralno delovanje računalniških rešitev sloni na strukturni delitvi na podatke (podatkovni model) ter na funkcionalnost (procesni model). S takšnim strukturiranim pristopom modeliramo sistem tako, da so podatki ločeni od obnašanja sistema tako v načrtovanem modelu kot v implementaciji računalniške rešitve. Pri objektno-orientiranem pristopu opredelimo sistem kot zbirko interaktivnih objektov. Objekti delajo stvari (vsebujejo funkcionalnost). Izraza predmetni pristop in objektni pristop sta sinonim za isti pojem, zato se v nalogi izmenoma pojavljata oba. Z uporabo objektno orientiranih konceptov skušamo razviti informacijski sistem, ki čim bolj verno odraža dogajanje v okolju, ki ga zaznavamo. Sloni na zaznavanju dogajanj v okolju, izvedba sistema predstavlja obnašanje in delovanje, ki je odraz realnosti (Damij, 2001, str. 5).

V objektnem pristopu razvoja računalniških rešitev imamo namesto aplikacije, ki pristopa k podatkom, aplikacijo, ki obstaja v nečem, čemur lahko rečemo objektni prostor (Ambler S. W., 1998b, str. 3). Objektni prostor si lahko predstavljamo kot začasni računalniški pomnilnik, kjer hkrati obstajata tako aplikacija kot podatki zanjo. Objektni način izvajanja računalniških rešitev predpostavlja izvedbo naloge ob pomoči objektov, ki predstavljajo entitete, v katerih so združeni tako podatki kot tudi opravila. Med objekti poteka izmenjava sporočil. Računalniška aplikacija se izvaja na osnovi klicev in izvedbe opravil, vgrajenih v objekte.

## 4.2 Značilnosti objektno orientirane tehnologije

V zadnjem obdobju razvoj informacijskih sistemov in računalniških rešitev čedalje bolj usmerja objektno orientirana paradigma. Vpliva na način modeliranja podatkov, načrtovanja računalniških rešitev in način snovanja informacijskih sistemov. Način objektnega razmišljanja se je na ravni uporabe informacijske tehnologije razrasel v široko področje uporabe, pogosto imenovano tudi objektno orientirana tehnologija (OOT).

Objektno orientiran razvoj informacijskih sistemov temelji na predstavljanju entitet realnega sveta v obliki objektov informacijskega sistema. Med temeljne koncepte objektne razvojne paradigme štejemo:

- objekte: objekt je vsebnik podatkov in procedur. Predstavlja preslikavo entitete iz realnega sveta v svet objektnega modeliranja;
- razrede objektov: razred objektov predstavlja skupino objektov s podobnimi lastnostmi, z enakim obnašanjem, enakimi odnosi z drugimi objekti ter z enakim pomenom v sistemu;

- atribute: atribut je podatek, ki ga poseduje objekt kot predstavnik razreda. Različni primerki objektov istega razreda lahko posedujejo enake ali različne vrednosti atributa;
- relacije: objekti so lahko v medsebojni relaciji. Asociacija je tip relacije, ki omogoča izvodu objekta pridobitev storitve na osnovi aktiviranja delovanja drugega objekta. Asociacijo med razredi objektov opredeljujemo v obliki tipov ena-ena, ena-mnogo, mnogo-mnogo;
- operacije: operacije predstavljajo delovanje objekta. Operacija predstavlja funkcijo, ki jo lahko izvede objekt ali razred objektov. Algoritem ali zaporedje ukazov, ki omogočajo izvedbo posamezne operacije, imenujemo metoda. Metode lahko zahtevajo parametre, ki prilagodijo delovanje operacije. Rezultat izvedbe metode je neka povratna vrednost;
- dedovanje: s pomočjo dedovanja lahko vzpostavljamo hierarhijo med razredi. Dedovanje omogoča prevzemanje atributov in operacij z višje ležečih (nadrazredi) hierarhičnih razredov na nižjem hierarhičnem nivoju (podrazredi). Proces dedovanja lahko izvajamo dvosmerno. Dopolnjevanje atributov in operacij na nižjem nivoju imenujemo specializacija. Posploševanje atributov in operacij na višjem hierarhičnem nivoju imenujemo generalizacija.

### 4.2.1 Modeliranje podatkov

Omenjene so že bile različne možnosti predstavitve organiziranosti in strukturiranja podatkov. S fizično predstavitvijo podatkovnega modela prikažemo konkretno (fizično) razporeditev podatkov na spominskih medijih računalnikov. Logična predstavitev podatkovnega modela predstavi podatke, ki se pojavljajo v obravnavanem sistemu, ter njihovo logično organiziranost z vidika uporabljene informacijske tehnologije. V praksi se je logična predstavitev podatkovnega modela spreminjala z razvojem informacijske tehnologije. Kronološko gledano ločimo hierarhični, mrežni, relacijski in objektni model logične predstavitve podatkov (Grad, Jaklič, 1996, str. 115).

Pred pojavom objektne razvojne paradigme je bil (in je še) zelo razširjen relacijski podatkovni model. Ta je zasnovan na osnovi matematične teorije relacij. Logična predstavitev podatkov je izvedena v obliki relacijskih tabel. Objektno modeliranje podatkov se od predhodnikov razlikuje po tem, da združuje modeliranje podatkov in modeliranje procesov. Logična predstavitev podatkov v tem primeru temelji na predstavljanju objektnih razredov.

Tretji način predstavitve podatkov je konceptualna predstavitev, kjer opišemo semantiko pojavov in stvari v domeni obravnavanega sistema ter njihove medsebojne povezave. Glede na različne načine predstavitev bi lahko rekli, da objektno modeliranje podatkov omogoča tudi njihovo konceptualno predstavitev.

Pri prehajanju na objektno zasnovane računalniške rešitve bi, z ozirom na dokaj pogosto predstavitev že vzpostavljenih podatkov v obliki relacijskih podatkovnih shem, lahko pričakovali, da jih bomo uporabili kot podlago za objektno modeliranje. Res jih lahko, vendar s pridržkom. Nekateri avtorji namreč opozarjajo na previdnost (Ambler, [URL: http://www.agiledata.org/essays/drivingForces.html]). Ambler opozarja na grožnjo, ki izhaja iz odklonov med objektno in relacijsko paradigmo predstavitve podatkov. Predlaga iskanje skladnosti med zasnovo računalniške rešitve in samo zasnovo podatkovne baze, zato sugerira preverjanje skladnosti s fizičnim podatkovnim modelom. Po drugi strani avtor opozarja tudi na razlike med kartiranjem objektov v relacijski bazi in uporabljenimi koncepti objektov.

### 4.2.2 Modeliranje sistemov, objektno modeliranje

Informacijski sistem in računalniške rešitve, zasnovane na objektno orientiranih konceptih in objektnih programskih jezikih, omogočajo ponovno uporabo sistemskih komponent. To je ena pomembnejših odlik objektnega modeliranja (Damij, 2001). Vsak sistem lahko obravnavamo kot nabor povezanih delov. Ti lahko med seboj tudi komunicirajo. Pri preučevanju delovanja sistemov in njihovih elementov si pomagamo z abstrakcijo podrobnosti, s katero izvajamo poenostavitev zaznavanja realnega stanja (sveta) v obliko modela vzajemno delujočih objektov. V splošnem želimo z modeliranjem oblikovati dovolj preproste modele (in sestavine), da jih lahko opazovalec v celoti razume (Bruegge, Dutoit, 2004, str. 35-36).

Področje delovanja (domena) računalniške rešitve pokriva vse vidike zahtev oziroma problemov uporabnika v sistemu, ki ga modeliramo. Za analitike in razvijalce je pomembno razumevanje domene aplikacije, da bi lahko modeliran sistem dosegel svoj namen. V procesu razvoja programske rešitve se z objektno zasnovano analizo posvečamo modeliranju domene aplikacije, z objektno zasnovanim načrtovanjem pa modeliranju v domeni rešitve (Bruegge, Dutoit, 2004, str. 41).

Objektno zasnovano modeliranje je aktivnost, ki jo predpisujejo in vključujejo številne metode razvoja računalniških rešitev. Osnovni namen objektno zasnovanega modeliranja je (Milićev, 2001, str. 5):

- razvoj modela programske opreme na višjem nivoju abstrakcije kot to omogoča objektno orientiran programski jezik,
- specifikacija modela na vizualen način, s pomočjo grafične notacije,
- transformacija abstraktnih, vizualnih modelov v izvedbeno obliko (implementacijsko formo). Ta predstavlja tipično programsko kodo v enem od objektnih programskih jezikov, ki izhaja iz objektnega modela.

S predstavitvijo sistema s pomočjo objektnega modela lahko predstavljamo objekte sistema z različnih gledišč. Vzpostavljeni objektni model uporabimo v različnih fazah gradnje informacijskega sistema. Objekti v sistemu izvajajo opravila (procese) in hkrati vsebujejo podatke. S formiranjem objektnega modela združujemo procesni model (funkcionalnost sistema) in podatkovni model (podatke sistema). Z uporabo istega modela (v različnih pogledih) v fazi analize, načrtovanja, kodiranja ter zasnovi podatkovne baze zmanjšamo možnost napak in komunikacijskega šuma med različnimi fazami izgradnje sistema.

## 4.3 Pripomočki v abstrakciji opazovanih sistemov

Preučevanju izbranega sistema, ki ga želimo predstaviti v obliki objektnega modela, praviloma izvajamo na različnih ravneh abstrakcije. Na višji ravni abstrakcije želimo razumeti koncept delovanja sistema. Na področju predstavitve znanja so poznane različne metode vizualizacije konceptov. Ena od njih so tematski zemljevidi objektov (angleško: topic maps). Omogočajo predstavitev informacij v obliki tematskih gesel (angleško: topic), s katerimi predstavimo poljuben koncept, asociacije in dogodke. Tematski zemljevidi objektov predstavljajo ISO standard (ISO/IEC 13250: 2003) za predstavljanje in izmenjavo znanja.

Koncepte kartiramo s pomočjo konceptnih diagramov. Z njimi predstavimo konceptualni model in relacije med koncepti. Ti diagrami so dokaj popularni v tehniki, imenovani viharjenje možganov (brainstorming) in kartiranje idej (angleško: mind mapping). Kartiranje konceptov je osnovno komunikacijsko sredstvo med različnimi udeleženci pri razvoju ali posnetku kompleksnega znanja.

Pri kartiranju konceptov se bolj izrazito osredotočamo na dejansko znanje oziroma poznano stanje, manj na ideje. S kartiranjem konceptov želimo v čim večji meri sistematično predstaviti poznavanje/razumevanje sistema, ki ga preučujemo.

### 4.3.1 Ontologija domene, metapodatki

Pri opredeljevanju konceptov oziroma abstraktnih pojmov se srečujemo tudi z zahtevo po predstavitvi znanja oziroma vedenja o obravnavanih pojmih in pojavih. Pri predstavljanju znanja pogosto uporabljamo izraz predstavljanje znanja o domeni. Znanje o domeni predstavlja statični nabor temeljnih podatkov in objektov iz obravnavanega področja (Corcho, Perez-Gomez, 2000).

Na področju predstavljanja znanja in ugotavljanja obstoja entitet ima velik pomen ontologija. Ontologija izvorno predstavlja filozofsko disciplino, ki se ukvarja z vprašanji: „Kaj obstaja?, Kaj je?, Kaj sem?, Kaj mi to pove?“ in podobnimi vprašanji o obstoju. Temeljno področje ontologije je iskanje subjektov, objektov in relacij. Ontologija kot filozofska disciplina izhaja iz razprav starogrških filozofov o konceptih realnosti in naravi bivanja. Z ontologijo običajno definiramo skupen slovar za raziskovalce, ki želijo izmenjevati znanje s področja izbrane domene. Ob tem ima lahko vsaka domena več pravilnih ontologij (Noy, McGuiness, 2001, str. 23). Kot splošna metodologija preučevanja obstoja in splošnih značilnosti entitet, ki nas obdajajo, se je ontologija v sodobnem času razširila tudi na področja informacijskih znanosti in umetne inteligence, kjer se ukvarja s predstavitvijo podatkov in znanja. Na področju računalniške tehnologije, ki se ukvarja z umetno inteligenco pod pojmom ontologija, razumemo eksplicitno formalno specifikacijo izrazov v domeni in relacij med njimi (Gruber, 1993).

Izdelava standardiziranih ontologij oziroma splošnih slovarjev domene je razširjena na področjih sistemov umetne inteligence, ter na področjih, kjer omogoča podporo, izmenjavo in ponovno uporabo formalno predstavljenega znanja. Standardizirana ontologija predstavlja slovar znanja, ki omogoča strokovnjakom znotraj različnih strokovnih domen njegovo beleženje in izmenjavo. Vanj so vključene definicije temeljnih konceptov v domeni in medsebojne relacije.

Lahko rečemo, da z ontologijo izdelamo specifikacijo univerzalnih konceptov in na ta način osnujemo bazo znanja. Formalna ontološka analiza se poslužuje klasifikacije in taksonomije, ki ureja pojme v hierarhično zgradbo, običajno na nadrejene in podrejene tipe. Na področju računalniške tehnologije z izdelavo ontologije opredelimo niz tipov, lastnosti in tipov relacij. V praksi se večina ontologij osredotoča na opredeljevanje razredov. Slednji predstavljajo koncepte v domeni. Izbor ontoloških kategorij predstavlja prvi korak v oblikovanju podatkovne baze, baze znanja ali objektno zasnovanega sistema (Sowa, 2000, str. 51).

Ob izdelavi ontologije običajno nastaja seznam izrazov iz domene. Ti izrazi predstavljajo temeljne semantične enote za prevajanje konceptov. Na ta način izdelamo tezaver oziroma žargon (besednjak) iz domene. Na njegovi osnovi lahko zgradimo slovar z natančnimi pomenskimi definicijami. Z izdelavo slovarja vzpostavimo metapodatke o entitetah v domeni. Pojem metapodatek je opredeljen v svoji splošni in najširši definiciji kot podatek o podatku. V

obravnavanem kontekstu je obravnavan metapodatek kot prinašalec (izmenjevalec) informacije o objektih. V razumevanju pojma metapodatek se lahko opremo tudi na definicijo, ki v opredeljevanju podatkov loči tri stopnje abstrakcije (Grad, Jaklič, 1996, str. 16):

- realni svet, ki ga predstavljajo objekti obravnavanega dejanskega problema z vsemi svojimi svojstvi,
- metapodatki, ki vsebujejo opredelitve podatkov (objektov),
- dejanski podatki, ki predstavljajo vrednosti predhodno opredeljenih zapisov oziroma elementarnih postavk.

V postopku modeliranja sistema razvoj ontologije v domeni obravnavajmo kot aktivnost, s katero izpostavimo formalne izraze, ki so zanjo značilni. Hkrati lahko vzpostavimo hierarhijo tipov in metapodatkovno zasnovo domene. V procesu razvoja računalniških rešitev je z vidika vzpostavljanja ontologije sistema izpostavljen predvsem naročnik (uporabnik) sistema. Razvoj ontologije in njeno uporabo lahko obravnavamo kot dejavnik za zmanjševanje komunikacijskega šuma pri izmenjavi znanja med strokovnjaki različnih profilov, ki sodelujejo v razvoju računalniške rešitve.

### 4.3.2 Razvoj objektnega modela

Ob pomoči vzpostavljene ontologije lahko naredimo seznam objektov domene in dogodkov, ki se dogajajo v obravnavanem sistemu. V objektnem modelu se osredotočimo na najvažnejše tipe objektov iz vsebine sistema. Če se nismo posebej posvečali razvoju ontologije, potem koncept delovanja sistema, objekte, dogodke in povezave odkrivamo s pomočjo specifikacije zahtev, ki se običajno izvaja z intervjuji med strokovnjaki v domeni. Objekte sorodnega tipa uvrščamo v razrede. Razredi objektov domene sistema se nam kažejo v treh pojavnih oblikah (Jacobson, Booch, Rumbaugh, 1999c, str. 119):

- poslovni objekti, ki predstavljajo stvari, ki jih uporabljamo v poslovni dejavnosti (na primer naročila, pogodbe, računi),
- objekti iz realnega sveta in koncepti, ki jim sledimo (na primer sovražno letalo, izstrelek, pot),
- dogodki, ki so se ali se bodo zgodili (na primer prihod letala, odhod vlaka, odmor za kosilo).

Znanje iz domene lahko pridobivamo na različne načine. Običajni so razgovori in intervjuji z izvajalci. Nekateri avtorji predlagajo uporabo prilagojenih vprašalnikov, s pomočjo katerih lahko pridobimo veliko znanja iz domene na osnovi formalne ontologije, vključene v semantiko zastavljenih vprašanj (Thai et al. 2006).

Namen modeliranja domene je opis in vzpostavitev razumevanja vzajemnega delovanja najpomembnejših razredov iz konteksta domene. Modeliranje domene mora prispevati k razumevanju problema, ki ga sistem rešuje v povezavi z njegovo vsebino (Jacobson, Booch, Rumbaugh, 1999c, str. 120).

Kot še bolj sistematičen način za prepoznavanje primerov uporabe in notranjih razredov sistema je predlagan razvoj poslovnega modela (Jacobson, Booch, Rumbaugh, 1999c, str. 122-128).

Skladno z navedbami avtorjev model domene predstavlja le poseben primer celotnega poslovnega modela.

### 4.3.3 Predstavljanje modelov, modelirni jeziki, poenoteni jezik modeliranja

Snovanje in izdelava (inženiring) programske opreme sloni na nekaj temeljnih aktivnostih (Bruegge, Dutoit, 2004, str. 5), in sicer na:

- modeliranju,
- reševanju problemov,
- zbiranju znanja in
- aktivnostih, osredotočenih v razumevanje sistema.

Modeliranje smo že opredelili kot abstraktno preslikavo nekega pogleda na stvarnost. V procesu modeliranja skušamo uresničiti osnovni namen modela, in sicer vzpostavitev sredstva za pridobivanje spoznanj, prenos znanja in preizkušanje brez tveganja za izvirnik (Kovačič, Bosilj-Vukšić, 2005, str. 177).

Za predstavitev zgradbe obravnavanega sistema, informacij ali znanja o domeni lahko poleg naravnih uporabljamo tudi umetne jezike. Ti vsebujejo niz pravil, s katerimi interpretiramo pomen predstavljenih komponent v zgradbi. Umetni jeziki so lahko grafični ali opisni. Opisni jezik za modeliranje običajno vsebuje standardizirane ključne besede, ob katerih navajamo parametre, s katerimi tvorimo izraze, ki jih lahko pozneje interpretiramo ob pomoči računalnika. Grafični modelirni jeziki običajno vsebujejo niz diagramov, ki vsebujejo različno poimenovane simbole, s katerimi predstavljamo obravnavane koncepte. Simbole predstavljenih konceptov povezujemo s črtami, s katerimi so ponazorjene relacije med njimi. Običajno grafični modelirni jeziki vsebujejo tudi druge grafične oznake za predstavitev raznih pravil ali omejitev, značilnih za obravnavane koncepte.

Dejavnik temeljnega pomena za natančno in uspešno komunikacijo v projektih razvoja programske opreme je uporaba notacije, ki je razumljiva udeležencem in je primerna za predstavitev njihovega pogleda na sistem (Bruegge, Dutoit, 2004, str. 29). Ob takšni trditvi se samo poraja sklepanje, da si širšo uporabno vrednost in razumljivost med različnimi udeleženci lahko zagotovimo z uporabo notacij, ki so uveljavljene kot industrijski standard.

Razvoj posameznih modelirnih jezikov se je, zgodovinsko razvojno gledano, prilagajal specifiki zahtev za posamezna področja uporabe, kot, na primer, za ekonomsko, poslovno, podatkovno modeliranje ali modeliranje za potrebe razvoja računalniške programske opreme. Za splošno in namensko rabo imamo danes na voljo različne jezike, med katerimi so bolj znani SysML (angleško: systems modeling language), Petrijeve mreže, BPMN (angleško: Business process modeling notation), FMC (angleško: Fundamental modeling concepts) in vrsto drugih, med katerimi so nekateri, kot, na primer, Express in Express-G (ISO 10303-11) ter tudi mednarodno standardiziran, poenoteni jezik modeliranja (angleško: UML-Unified modeling language (ISO/IEC 19501).

Med pogosto citirane in uporabljane sodi poenoteni jezik modeliranja (UML). Uporablja se na različnih področjih, od modeliranja programske opreme do uporabe na področju sistemskega inženiringa, modeliranja poslovnih procesov in predstavljanja različnih organizacijskih struktur. Izhaja iz poenotenja različnih notacij, uporabljenih v tehnikah objektnega načrtovanja razvoja

programske opreme, ki so bile objavljene v devetdesetih letih prejšnjega stoletja in v katerih so sodelovali različni avtorji, in sicer (Wieringa, 1998, str. 512-524): (Mellor, Schlaer, 1988), (Coad, Yourdon, 1990), (Booch, 1991), (Rumbaugh, 1991), (Jacobson, Martin, Odell, 1992). Namen vzpostavitve poenotenega jezika je bila želja po standardni notaciji, ki bi bila uporabna za vse tedaj znane objektne metode in bi vsebovala njihove najboljše sestavine. Številni avtorji do tedaj objavljenih različnih metodologij objektnega razvoja so ugotovili, da za razvoj računalniških rešitev ni treba predpisovati celotnih metodologij, pač pa je dovolj vzpostaviti notacijo in semantiko, ki podpira razvojni proces programske opreme. Kot rezultat je nastal poenoteni jezik modeliranja, ki ga je organizacija OMG (angleško: Object Management Group) sprva obravnavala kot predlog standarda in leta 1997 tudi objavila kot standard (UML1.1).

Danes se na področju projektov razvoja programske opreme, kot sredstvo načrtovanja in komunikacije, v večini primerov uporablja prav poenoteni jezik modeliranja. Jezik se iz začetnega standarda razvija in dopolnjuje. Za njegove dopolnitve in nove definicije skrbi posebna skupina (angleško: Revision task Force) znotraj organizacije OMG. Danes se jezik, poleg modeliranja zgradbe programskih rešitev in arhitekture sistema, uporablja tudi na področju modeliranja poslovnih procesov in modeliranja podatkov. Trenutno je aktualna verzija jezika UML 2.1, kot industrijski standard ISO/IEC 19501 pa je uporabljena verzija UML 1.4.2.

Pomen uporabnosti poenotenega jezika modeliranja kot sredstva za komunikacijo v razvoju programske opreme lahko utemeljujemo z njegovimi naslednjimi možnostmi (Milićev, 2001, str. 7):

- vizualizacija modelov,
- specifikacija, navedba nedvoumnih modelov,
- predstavitev zgradbe programske rešitve, ki jo pozneje izdelamo,
- dokumentiranje zahtev, zgradba projekta in izvorne kode.

Izvorno je jezik zasnovan na treh elementih, in sicer na (Jacobson, Booch, Rumbaugh, 1999b, str. 17-18):

- osnovnih gradnikih,
- pravilih, ki usmerjajo način povezave gradnikov,
- nekaterih splošnih mehanizmih, ki veljajo za celoten jezik.

Med gradnike poenotenega modelirnega jezika štejemo:

- stvari (angleško: things),
- relacije (angleško: relations),
- diagrame (angleško: diagrams).

Stvari so temeljni gradniki modela, relacije jih medsebojno povezujejo. Z diagrami povezujemo skupine stvari. Stvari delimo na več vrst:

- sestavne (angleško: structural),
- vedenjske (angleško: behavioral),
- povezovalne (angleško: grouping),
- pojasnjevalne, označevalne (angleško: annotational).

Sestavne stvari predstavljamo kot samostalnike v modelu. Predstavljajo statični del modela. Ločimo sedem vrst sestavnih predmetov (Jacobson, Booch, Rumbaugh, 1999b, str. 18-19):

- *Razred* predstavlja opis skupine objektov, ki si delijo enake atribute, operacije, relacije in pomen (semantiko). Razred izvaja enega ali več vmesnikov.
- *Vmesnik* predstavlja skupino operacij, s katero so določene storitve razreda ali komponente. Z vmesnikom je opisano obnašanje elementa kot ga vidimo od zunaj. Vmesnik opredeli specifikacijo operacije, ne pa konkretne izvedbe operacije.
- *Sodelovanje (kolaboracija)*. Z njim definiramo sodelovanje (participiranje) v vzajemnem učinkovanju skupine povezanih elementov. Sodelovanje je združba elementov, ki delujejo vzajemno in skupaj zagotavljajo učinek, ki presega zgolj seštevek delovanja posameznih delov. Ima tako strukturni kot vedenjski značaj.
- *Primer uporabe*. Predstavlja opis skupine zaporednih aktivnosti, ki jih izvaja sistem, da bi končni rezultat njihove izvedbe pomenil uporabno vrednost za akterja oziroma soudeleženca. Izvedba primera uporabe je realizirana s pomočjo sodelovanja.

Preostale tri vrste sestavnih predmetov so podobne razredu, saj si tako kot razred delijo enake atribute, operacije, relacije in pomen. Kljub vsemu se med seboj dovolj razlikujejo, da jih opredelimo kot dodatne samostojne elemente za predstavitev nekaterih vidikov objektnega sistema, ki ga modeliramo. Mednje štejemo (Jacobson, Booch, Rumbaugh, 1999b, str. 20):

- *Aktivni razred* predstavlja razred, katerega objekti vsebujejo enega ali več procesov in lahko posledično sprožijo kontrolno aktivnost. V bistvu je ta sestavni element enak kot razred, le s to razliko, da lahko njegovi objekti izvajajo proces, ki se odvija istočasno z drugimi procesi.
- *Komponenta* predstavlja fizični in izmenljiv del sistema, ki mu je prilagojen in zagotavlja izvedbo skupine vmesnikov. Praviloma predstavlja komponenta fizično samostojen paket skupine logičnih elementov, kot so razredi, vmesniki in sodelovanje.
- *Vozlišče* predstavlja fizični element, ki se nahaja na mestu izvajanja in predstavlja računalniški vir za izvedbo. Ta ima v splošnem vsaj nekaj spomina in pogosto tudi zmožnost procesiranja. Na vozlišču je lahko več komponent, ki se lahko tudi selijo med njimi.

Vedenjske stvari so, kot že ime pove, namenjene obnašanju in predstavljajo dinamični del modela. V modelu so to povedi (glagoli), ki predstavljajo obnašanje v času in prostoru. Ločimo dve osnovni vrsti vedenjskih stvari (Jacobson, Booch, Rumbaugh, 1999b, str. 21):

- *Interakcija* je obnašanje, ki vsebuje skupino sporočil. Ob izmenjavi teh sporočil med objekti znotraj izbranega konteksta se doseže določen namen. Z interakcijo lahko opredelimo bodisi posamično operacijo objekta bodisi obnašanje skupine objektov. Interakcija se navezuje na več elementov, vključno s sporočili, zaporedje izvedenih aktivnosti in povezave med objekti.
- *Stanje* je obnašanje, ki opredeli zaporedje stanj, skozi katera gre bodisi objekt bodisi interakcija med svojim življenjskim ciklom, to je v času odzivanja na dogodke. S stanjem stroja lahko opredelimo obnašanje posameznega razreda ali razredov, ki medsebojno sodelujejo. Stanje stroja se navezuje na več drugih elementov, vključno s stanji, tranzicijami (tokom dogajanja med stanji), dogodki (stvarmi, ki sprožijo tranzicijo) in aktivnostmi (odgovori na tranzicijo).

Povezovalne stvari so sestavine za organizacijo modelov. So kot škatle, v katere lahko razstavimo model. V splošnem predstavlja povezovalni predmet paket. *Paket* je splošnonamenski mehanizem za organiziranje elementov v skupine. Za razliko od komponent, ki obstajajo med izvedbo, so paketi zgolj konceptualne narave, se pravi, da obstajajo samo v času razvoja.

Pojasnjevalne stvari sodijo med tiste dele modela, ki pojasnjujejo. To so različni komentarji, namenjeni opisovanju, pojasnjevanju ali pripombam elementov v modelu. Pojasnjevalni element je *opomba*. Opomba kot element je simbol, v katerega vpišemo zabeležke in komentarje, vezane na posamezen element ali skupino elementov.

Druga temeljna skupina gradnikov jezika so relacije. Razlikujemo štiri vrste relacij (Jacobson, Booch, Rumbaugh, 1999b, str. 23-24):

- *Odvisnost* je pomenska (semantična) relacija med dvema predmetoma, kjer sprememba v enem (neodvisnem) povzroči spremembo v drugem (odvisnem
- *Asociacija* ima tvorni značaj in predstavlja relacijo, ki opisuje povezavo med objekti. Posebna vrsta asociacije je agregacija, ki predstavlja tvorne relacije med celoto in njenimi deli.
- Tretja relacija je *generalizacija*. Predstavlja relacijo specializacije oziroma generalizacije, v kateri so objekti specializiranega elementa (otroci) nadomestljivi z objekti generaliziranega elementa (starši). Na ta način otroci delijo zgradbo in obnašanje staršev.
- *Realizacija* je pomenska (semantična) relacija med klasifikatorji, kjer en klasifikator opredeli dogovor, drugi klasifikator pa jamči izvedbo dogovora. Uporablja se med vmesniki in razredi (ali komponentami), ki jih realizirajo. Prav tako se uporabljajo med primeri uporabe in sodelovanji, ki jih realizirajo.

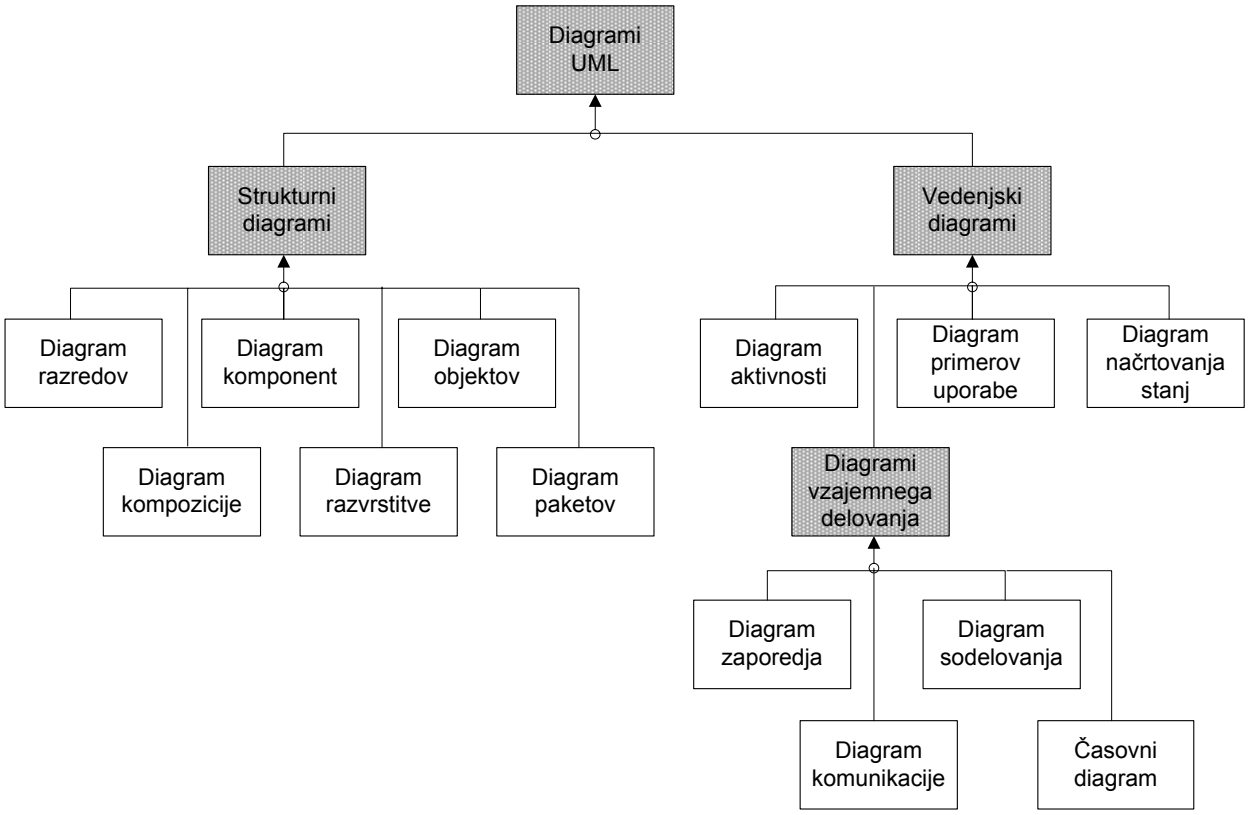
Tretja temeljna skupina gradnikov poenotenega modelirnega jezika so *diagrami*. Diagrami so grafična predstavitev skupin elementov, najpogosteje prikazani kot povezan graf točk (predmetov) in povezav (relacij) med njimi. Z diagrami vizualno predstavimo sistem z različnih perspektiv. Posamezen element se lahko pojavi na enem ali več diagramih. Načelno lahko diagram vsebuje poljubno kombinacijo predmetov in relacij. V praksi uporabljamo nekaj tipičnih kombinacij predmetov in relacij na diagramih. Med osnovne diagrame uvrščamo (Jacobson, Booch, Rumbaugh, 1999b, str. 25-26):

- diagram razredov, s katerim predstavimo statično zgradbo razredov in njihovih medsebojnih povezav,
- diagram objektov, ki predstavlja objekte in njihovo povezanost,
- diagram primerov uporabe, s katerim predstavimo funkcionalnost sistema z gledišča uporabnika,
- diagram zaporedja, s katerim predstavimo dinamiko sodelovanja med objekti,
- diagram sodelovanja je prav tako namenjen predstavitvi sodelovanja med objekti,
- diagram načrtovanja stanj je namenjen predstavitvi vseh možnih stanj objektov in dogodkov, ki povzročijo spremembe,
- diagram aktivnosti je namenjen predstavitvi zaporedja toka aktivnosti sistema,
- diagram komponent omogoča prikaz komponent sistema,
- diagram razvrstitve prikaže vzpostavitev komponent na posamezni strojni opremi.

Modelirni jezik dovoljuje tudi izdelavo drugih vrst diagramov glede na potrebe uporabnika (Milićev, 2001, str. 11). Skladno s to zamislijo je bil osnovni nabor diagramov pozneje dopolnjen z naslednjimi diagrami:

- diagram kompozicije predstavlja notranjo zgradbo razreda,
- diagram paketov opiše delitev sistema na logične skupine,
- diagram komunikacije je poenostavljena verzija diagrama sodelovanja,
- časovni diagram prikazuje specifične časovne omejitve v sistemu.

Slika 1: Skupine diagramov poenotenega jezika za modeliranje (UML)




Vir: Povzeto po [URL: http: //en.wikipedia.org/wiki/Unified_Modeling_Language]

Navedene skupine osnovnih gradnikov predstavljajo prvi koncept poenotenega modelirnega jezika. Drugi temeljni koncept predstavljajo pravila. Osnovnih gradnikov namreč ne moremo poljubno povezovati med seboj. Zato poenoteni modelirni jezik vsebuje vrsto pravil. Ta pravila opredeljujejo, kako naj bi izgledal dobro zasnovan model. Dobro zasnovan model je namreč tisti, ki je pomensko konsistenten in skladen z modeli, ki so z njim v relaciji. Jezik ima vrsto pravil za (Jacobson, Booch, Rumbaugh, 1999b, str. 26):

- imena - poimenovanje predmetov, relacij, diagramov,
- doseg - kontekst, ki opredeli specifičen pomen imen,
- vidnost - kako so imena med seboj vidna,
- celovitost - kako se predmeti pravilno in skladno navezujejo med seboj,
- izvedbo - kaj pomeni simulacija dinamičnega sistematičnega.

Modeliranje je zamišljeno kot dinamičen proces, v katerem so v fazi izgradnje možni različni pogledi njegovih izdelovalcev. V tem smislu so pravila bolj usmerjevalne kot deklarativne narave. Glede na običajno prakso je proces modeliranja običajno iterativen in inkrementalen postopek, zato pot do dobro zasnovanega modela sloni na ponovitvah definicij gradnikov, njihovem spreminjanju in prilagajanju.

Tretji koncept, ki nastopa v poenotenem modelirnem jeziku, so splošni mehanizmi, ki predstavljajo splošen vzorec v uporabi jezika. V jeziku se pojavljajo naslednji splošni mehanizmi (Jacobson, Booch, Rumbaugh, 1999b, str. 27):

- specifikacije,
- dodatki,

- delitev pomena,
- mehanizmi razširitve.

Za vsako grafično predstavitvijo posameznega dela obstaja *specifikacija*, ki predstavlja tekstni opis sintakse (pravila uporabe) in pomena sestavine. Večina elementov ima unikatno in neposredno grafično predstavitev namenjeno njegovi vizualizaciji. Elementi imajo nek osnovni izgled, ki pa ga lahko po presoji dopolnimo s paleto specifičnih *dodatkov*. V objektno orientiranih sistemih realnost delimo na več načinov. Ločimo razrede in objekte. Razred je abstrakcija, objekt je konkreten predstavnik te abstrakcije. Lahko imamo primere uporabe in primerke primerov uporabe. Ta *delitev pomena* se med gradniki večkrat pojavi. Aktualno lahko postane ločevanje vmesnika in izvedbe. Vmesnik opredeljuje dogovor, izvedba pa predstavlja posamično konkretno realizacijo tega dogovora. Poenoteni jezik modeliranja je orodje za načrtovanje programske opreme. Z zaprtim, končno deklariranim jezikom bi bilo nemogoče modelirati paleto nians modelov različnih domen. Zato jezik vsebuje *mehanizme razširitve*, ki omogočajo uporabniku odprtost in možnost razširitve jezika. Med mehanizme razširitve štejemo (Jacobson, Booch, Rumbaugh, 1999b, str. 29):

- stereotipe,
- etiketiranje vrednosti,
- prilagoditev.

*Stereotip* omogoča razširitev besednjaka jezika tako, da sestavimo nov element. Slednjega izpeljemo iz obstoječega na način, ki najbolj ustreza našim potrebam. *Etiketiranje vrednosti* omogoča razširitev lastnosti gradnika jezika, kjer dodamo novo informacijo v specifikacijo elementa. *Prilagoditev* (angleško: constraint) omogoča razširiti pomen (semantiko) sestavnega elementa jezika tako, da bodisi dodamo nova pravila, bodisi prilagodimo obstoječa.

### 4.3.4 Proces razvoja programske opreme, metodologije

Metodologije razvoja programske opreme so „priporočene zbirke delovnih faz, procedur, pravil, tehnik, orodij, dokumentacije, menedžmenta in usposabljanja, namenjenega razvoju sistema“ (Avison, 2003).

Danes poznamo vrsto metodologij za razvoj programske opreme oziroma računalniške rešitve. Med strokovnjaki in poznavalci ni enotne ocene o tem, katera je najboljša. Med nekaterimi strokovnjaki se vnemajo celo razprave o negativnih vplivih uporabe razvojne metodologije, saj trdijo, da predpisane tehnike lahko omejujejo načine mišljenja udeležencev razvoja o problemih v obravnavani domeni. Pojavljajo se trditve, da način predstavitve problema, kot ga predpostavljajo predpisane tehnike, lahko bistveno vpliva na razumevanje problema in posledično na sprejemanje odločitev (Avison, Adams, 2003).

Ne glede na morebitne dvome o uporabnosti metodologij, lahko na osnovi njihove množične razširjenosti in citiranosti zaključimo, da poznavanje razvojnih oblik metodologij in značilnih razlik med njimi sodi v osnovno znanje tistih, ki načrtujejo in vodijo projekte razvoja programske opreme. Skušajmo skozi nekatere razvojne oblike procesa razumeti značilnosti oziroma različnosti predlaganih metodoloških pristopov.

Če predpostavimo, da je programska oprema sistem, lahko njeno izdelavo izvajamo po načelih sistemskega inženiringa. Izvajanje sistemskega inženiringa je definirano kot splošen proces, ki

zagotavlja mehanizme za prepoznavanje in razvijanje proizvodov ter definicij procesov sistema. Razvoj sistemov se običajno odvija skozi življenjski razvojni cikel.

Eden od standardov IEEE opisuje procese življenjskega cikla programske opreme [IEEE Std. 1074-1997]. Standard predpisuje sedemnajst obveznih procesov v času razvoja in vzdrževanja programske opreme. Razvojne procese obravnava kot skupine aktivnosti, posamezne aktivnosti pa opredeljuje kot naloge, sestavljene iz podaktivnosti, ki jih mora izvesti posameznik ali skupina, da doseže določen namen. Naloga je opredeljena kot potrošnik virov (osebje, čas, denar), ki producira delovni izdelek. Posamezna naloga predstavlja najmanjšo delovno enoto, ki je relevantna za ravnanje projekta. Predstavlja osnovo za načrtovanje ter spremljanje dela in ukrepanje v primeru težav. Na osnovi aktivnosti, potrebnih za končno izvedbo projekta, projektni vodja vzpostavi projekt življenjskega cikla programske opreme. Nabor aktivnosti je lahko selektiven, standard je tudi neobvezujoč v zaporednosti njihove izvedbe. V času celotne izvedbe projekta se hkrati in vzporedno izvaja še nekaj celovitih oziroma integralnih procesov, to so proces verifikacije skladnosti modelov s specifikacijo, proces nadzora konfiguracije, ki nadzira razvoj sprememb in verzij, proces dokumentacije, ki se ukvarja z dokumentiranjem delovnih izdelkov (razen kode) in proces izobraževanja, v katerem načrtujemo in razvijamo program izobraževanja.

Aktivnosti v izvajanju sistemskega inženiringa potekajo skozi paleto različnih medsebojno prepletenih procesov v domeni različnih strokovnjakov. V razvoju programske opreme medsebojno prepletenost procesov usklajujejo različni standardi. Med tiste, ki jih pri razvoju programske opreme ne bi smeli spregledati, zanesljivo sodita ISO/IEC 15288 Systems engineering-System life cycle processes in IEEE Standard for Application and Management of the Systems Engineering Process [IEEE-1220-2005].

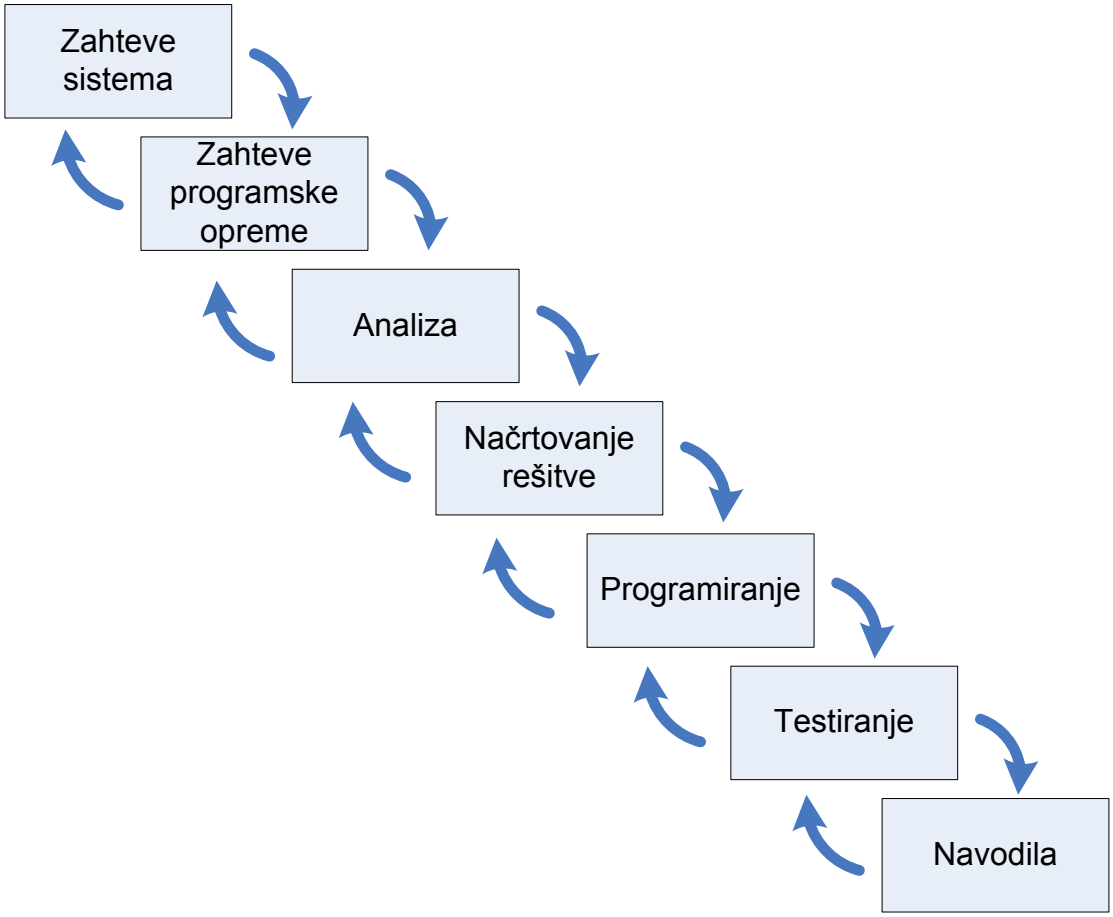
Razvoj programske opreme se odvija v življenjskem krogu, za katerega so značilne faze začetek, razvoj, uvajanje in izvajanje ter vzdrževanje. Razvojni pristop izhaja iz inženirske delovne metode, poimenovane življenjski cikel sistema. Njena prvotna značilnost je zaporednost izvajanja faz. Iz nje izhaja veliko modificiranih delovnih metodologij, ki ohranjajo osnovne fazne značilnosti, spreminjajo pa način izvedbe ciklov oziroma razvojnih faz. Sodobnejši pristopi zanikajo pomen podrobnega načrtovanja projektov razvoja programske opreme in ponujajo drugačne pristope. V grobem lahko klasične in sodobne pristope razdelimo na dve večji skupini, kjer lahko predstavimo tipične značilnosti enega ali drugega metodološkega koncepta.

#### 4.3.4.1 Kaskadne metodologije (metodologije vodnih slapov)

Metodologije razvoja programske opreme so se razvile v želji, da bi izdelava slednje postala bolj predvidljiva in učinkovita. Končni cilj, ki ga zasledujemo v projektu razvoja, je izdelava kakovostne programske opreme. To pomeni, da bo zadovoljila zahteve naročnika (uporabnika) in bo izdelana v predvidenem časovnem in stroškovnem okviru.

V sedemdesetih letih prejšnjega stoletja je Winston W. Royce objavil svoj pogled na ravnanje pri razvoju obsežne programske opreme (Royce, 1970). Izhaja iz dveh skupin opravil, ki so po njegovem prisotne v vsakem razvoju programske opreme, in sicer iz analize in programiranja. Pri izdelavi obsežnejših programov Royce predlaga pred izvedbo analize še izvedbo faze vzpostavitve zahtev sistema in zahtev programske opreme. Med analizo in programiranjem predlaga izvedbo faze načrtovanja programske opreme. Fazi izdelave programske opreme sledi faza testiranja. Za izboljšanje predlaganega procesa je avtor predlagal ponavljanje posameznih faz na način vračanja v predhodne razvojne faze.

Slika 2: Izvedbeni koraki in ponavljanje faz v razvoju obsežne programske opreme




Vir: povzeto po Royce, 1987

Za uspešnejšo izvedbo projekta naj bi posamezne faze razvoja izvajalo specializirano osebje. Predlagani način razvoja programske opreme je zahteval obsežno spremljajočo dokumentacijo v vsaki razvojni fazi (slika 3.).

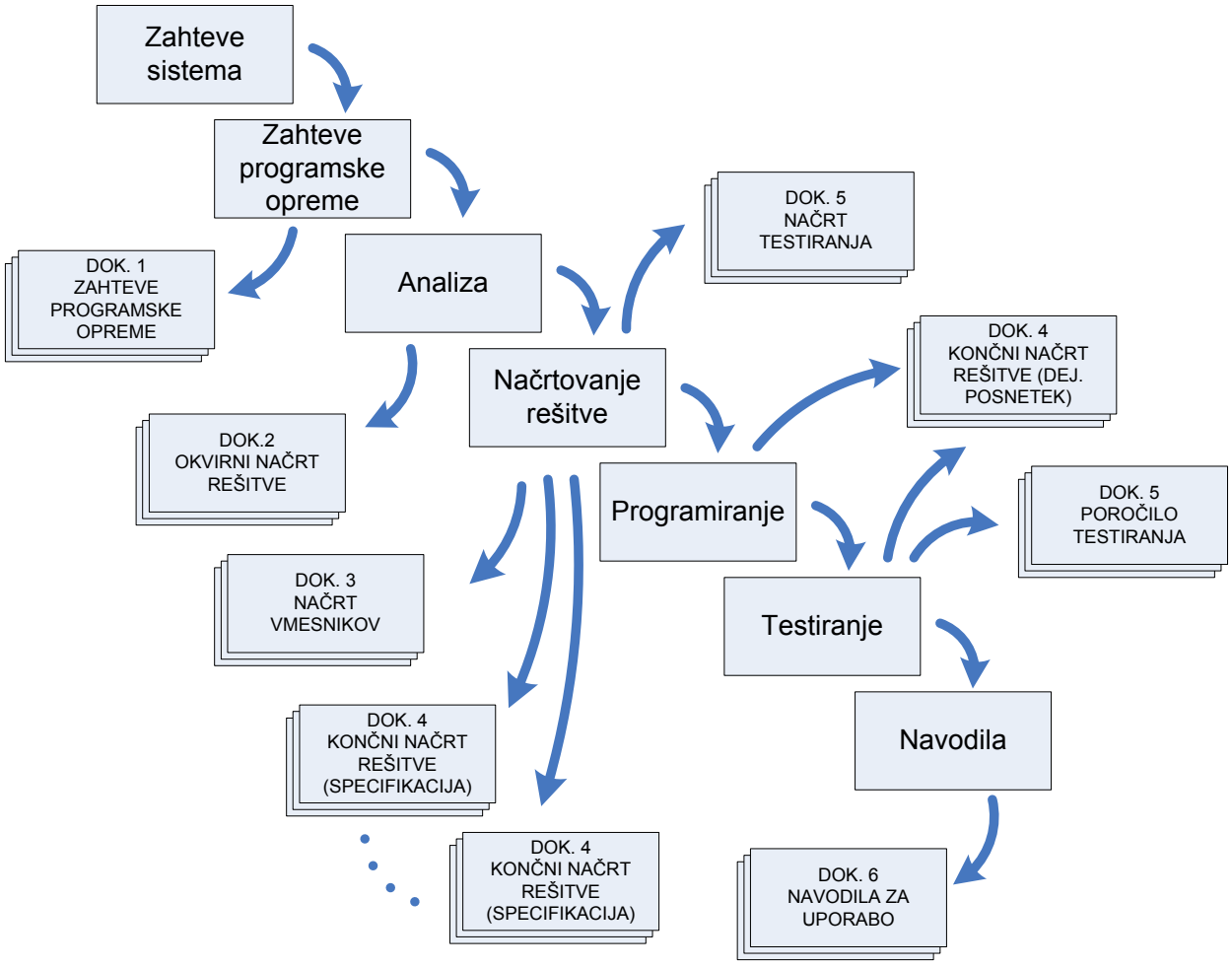
Začetni model zaporednega izvajanja razvojnega procesa in aktivnosti v življenjskem obdobju razvoja programske opreme so kmalu začeli nadomeščati bolj prožni modeli izvedbe razvojnega procesa. Osnovne značilnosti razvoja lahko povzamemo v spodnjem pregledu:

- Model izvedbe v obliki vodnih slapov ali kaskadni način: predvideva aktivnostno naravnan življenjski cikel, ki predpisuje zaporedno izvedbo procesov. Vse aktivnosti izbranega procesa morajo biti izvedene pred začetkom naslednjega. Ta model izvedbe ne predvideva vračanja v predhodne aktivnosti. Pravilnost izvedbe poteka skozi trajne nadzorne aktivnosti (Royce, 1970).
- V-model: je variacija kaskadnega modela, ki eksplicitno izpostavlja odvisnost med razvojnimi aktivnostmi in nadzornimi aktivnostmi. Pojavil se je konec sedemdesetih let prejšnjega stoletja. Vzpostavlja sledljivost med zahtevami in načrtovanimi elementi ter testi sprejemljivosti. Na ta način ne delamo ničesar, kar ni posebej zahtevano in sprejemljivo (Bruegge, Dutoit, 2004, str. 630). V projektnem inženiringu se takšen pristop omenja za primerljivega z metodo PRINCE2.
- Spiralni model: je predlagan (Boehm, 1986) za odpravljanje nekaterih slabosti kaskadnega modela. Sloni na ponavljanju oziroma iteraciji posameznih aktivnosti.

Prevzema aktivnosti kaskadnega modela, ki jim doda nove aktivnosti. Dodane aktivnosti, kot so nadzor tveganja, ponovna uporaba in prototipiranje, izvajamo v spiralnem ciklu. Model obravnava tveganje na postopen, inkrementalen način, po prioritetnem zaporedju. Predvideva podrobno zbiranje zahtev, izdelavo okvirnega načrta rešitve ter izdelavo prototipa. Pred izdelavo drugega podrobnejšega prototipa je izvedeno vrednotenje prvega ter podana ocena tveganja. Iz te ocene izhaja definicija zahtev ter načrtovanje drugega prototipa. Sledi izdelava in testiranje drugega prototipa.

- Poenoteni razvojni proces: (Jacobsen, Booch, Rumbaugh, 1999c). Arhitektura procesa predpostavlja, da se neka programska rešitev spreminja od rojstva do smrti. Obdobja v tem razvoju (spreminjanju) programske opreme razdeli na cikle. Tu gre za drugačen pomen teh ciklov kot pri spiralnem modelu. Rezultat vsakega cikla je namreč nova uporabna verzija programske opreme. Vsak cikel tvorijo štiri stanja oziroma faze, skozi katere izvajamo razvoj programske rešitve. Te faze so idejna faza, faza obdelave, faza konstruiranja in prehodna faza. Vsako fazo izvedemo skozi pet delovnih tokov (angleško: workflow). Ti deli celotnega procesa so zbiranje, analiza, načrtovanje, izvedba in testiranje. Proces predpostavlja inkrementalno in iterativno izvedbo delovnih tokov. Danes predstavlja de facto standard na aktivnostih utemeljenega iterativnega razvojnega modela.

Slika 3: Zahtevana dokumentacija v fazah razvoja programske opreme




Vir: povzeto po Royce, 1987

Na osnovi navedenih modelov je bila vzpostavljena vrsta metodologij razvoja računalniških rešitev, ki opredeljujejo različne načine izvedbe aktivnosti in predpisujejo različne spremljajoče delovne pripomočke ter rezultate dela. Za temeljitejše razumevanje metodoloških izhodišč je dobro, če ločimo med izrazoma metoda in metodologija.

Metoda je ponovljiva tehnika, ki določa korake, potrebne za rešitev specifičnega problema. Metodologija je na drugi strani zbirka metod za reševanje skupine sorodnih problemov, ki opredeljuje, kje in kdaj, odvisno od okoliščin, naj uporabimo posamezno metodo (Bruegge, Dutoit, 2004, str. 15).

S stališča izvajanja posameznih metodologij običajno eden od pogledov izhaja iz faze življenjskega cikla razvoja, znotraj katere priporoča izvedbo skupine povezanih aktivnosti. Drugi pogled na isti življenjski cikel pa je osredotočen na delovne izdelke in načrtovalske pripomočke, ki jih izdelamo skozi aktivnosti. Prvi pogled usmerja udeležence na način izvedbe dela, drugi na vsebino ter zgradbo izdelkov in pripomočkov, potrebnih za poznejše faze dela.

Preučevanje standarda za razvoj programske opreme pokaže, da je poleg razvojnih procesov, namenjenih končni programski rešitvi, potrebno izvajati še vzporedne upravljalske in integralne procese. Vsa ta množica procesov, ki se med seboj prepletajo, redkokdaj dopušča njihovo zaporedno izvajanje (Bruegge, Dutoit, 2004, str. 628). Prav prepletenost procesov in njihovih aktivnosti pa že tako zahtevno nalogo razvoja programske opreme naredi še težje obvladljivo.

Danes je dokaj razširjena ocena, da kaskadne metodologije niso dosegle širšega razmaha zaradi svoje relativne obsežnosti, dokaj birokratske narave izvajanja in zahtev po širokem spektru kadrovskih virov. Uporabljajo se predvsem za gradnjo obsežnejših sistemov in v okoljih, ki svojo organiziranost in kadrovsko zasedbo zmorejo prilagoditi zahtevam uporabljene metodologije. Njihova značilnost je skrbno načrtovanje razvoja v vseh fazah življenjskega razvojnega kroga ter nadzor in dokumentiranje izvedbe z namenom preprečevanja večjih napak. Razvoj sistema običajno traja dolgo in zahteva sodelovanje večjega števila uporabnikov in informatikov (Gradišar, Resinovič, 2001, str. 429).

Obstaja tudi ocena, da je v specifičnih pogojih njihova uporaba celo neprimerna. V primeru, ko je čas med spremembami znatno krajši kot je čas trajanja aktivnosti in se spremembe lahko pojavijo tako na področju splošne uporabe kot na področju končnih rešitev, V-model in osnovni kaskadni razvojni model izkazujeta vrsto slabosti (Bruegge, Dutoit, 2004, str. 628).

Skozi čas in na osnovi praktičnih izkušenj uporabe različnih metodologij razvoja se je pokazalo, da v trendu hitrega tehnološkega razvoja metodologije kaskadnega tipa izkazujejo dokaj variabilno uporabno vrednost. To je posledično pripeljalo do vrste metodoloških izpeljank in recikliranja že videnih metodoloških pristopov.

Razvoj računalniških rešitev se izvaja po načelih sistemskega inženiringa, kjer je potrebno formulirati problem, ga analizirati, iskati rešitve, sprejeti odločitev o ustrezni rešitvi in predlagati rešitev. Uporabnika je treba zadovoljiti z rešitvijo, ki je izvedena z načrtnimi količinami, v predvidenem času in v okviru razpoložljivosti virov. Ne glede na številne kritike o zahtevnosti in omejeni uporabnosti izvajanja kaskadnih metodologij ostaja dejstvo, da so prav slednje največ prispevale k razumevanju udeleženih aktivnosti in prepletenosti procesov v projektih razvoja novih računalniških rešitev. Standardi, navedeni na začetku tega poglavja, v veliki meri temeljijo na kaskadnih metodah razvoja računalniških rešitev, poleg tega pa poenoteni razvojni proces nekateri obravnavajo kot de facto standard za iterativni model razvoja programske opreme (Bruegge, Dutoit, 2004, str. 639-640).

Pri vsaki večji nalogi, tudi v razvoju programske opreme, se praviloma držimo nekega koncepta, ki usmerja zasnovo in izvedbo projekta. Razvoj programske opreme predstavlja projekt skupinskega dela, v katerem opredelimo posamezne naloge, časovne roke in odgovornosti za uspešno izvedbo končnega izdelka. Tudi posamezne dele celotne rešitve lahko izdelujemo skozi življenjski razvojni krog in ponovitve njegovih faz. V tem kontekstu lahko metodološka navodila obravnavamo kot nepogrešljiv delovni pripomoček. Tudi preučevanje različnih kaskadnih metodologij razvoja pred njihovo konkretno uporabo v razvoju računalniške rešitve lahko, glede na povedano, obravnavamo kot pomemben dejavnik za zmanjševanje tveganja uspešnosti projektov razvoja programske opreme.

Spremljevalec tradicionalnih razvojnih metodologij, od katerih so nekatere postale de facto standard, so tudi alternativne poti izdelave računalniških rešitev, med katere prištevamo neformalen pristop, prototipiranje, razvoj s strani končnih uporabnikov in uporabo programskih paketov. Običajno jih srečujemo pri razvoju manj obsežnih rešitev, ki dopolnjujejo posamično funkcionalnost obsežnejšega informacijskega sistema (Gradišar, Resinovič, 2001).

#### 4.3.4.2 Agilne metodologije

Gibanje agilnega razvoja računalniških rešitev se je razvilo kot alternativa tradicionalnim razvojnim metodologijam (Runeson, Greberg, 2000). Kot odziv na dokaj togo zasnovo kaskadnih metodologij se je v zadnjem desetletju uveljavil model agilnega razvoja programske opreme. Tudi ta model izdelavo programske opreme obravnava kot projekt, ki ga je treba načrtovati, a mnogo manj kot pri klasičnih metodologijah. Osredotoča se na zmanjševanje stroškov, povezanih z načrtovanjem. Model sloni na ponavljanju razvojnega cikla, v katerem izdelamo delujočo računalniško rešitev. Vsaka ponovitev predstavlja celoten razvojni projekt, vključno z analizo (dopolnjevanjem) zahtev, načrtovanjem, kodiranjem, testiranjem in dokumentiranjem. Agilne metodologije razvoja računalniških rešitev izhajajo iz manifesta, ki je osnovan na naslednjih predpostavkah [URL: http: //agilemanifesto.org/]:

- delovanje posameznika in vzajemno sodelovanje z ostalimi udeleženci razvoja ima prednost pred procesom in uporabo orodij,
- delujoča programska oprema ima prednost pred obsežno dokumentacijo,
- sodelovanje z naročnikom ima prednost pred pogajanji o sporazumu naročila,
- odzivanje na spremembe ima prednost pred izpolnjevanjem načrta.

Metodologije, ki slonijo na modelu agilnega razvoja, so predlagane z namenom povečevanja produktivnosti, prijaznejše uporabe udeležencem v razvoju, spreminjanja delovne kulture in vpeljave novih načinov dela. Med osnovnimi nameni novih agilnih metodologij je zmanjševanje stroškov, povezanih s spremembami zahtev. To prinaša iterativnost razvojnega cikla, ki omogoča sprotno odzivanje na spreminjajoče zahteve naročnika. V takšnem načinu dela sta poudarjena sodelovanje z naročnikom (uporabnikom) in redukcija izdelave načrtovalskih pripomočkov, ki so ocenjeni kot časovno potratni. To pomeni tudi redukcijo pisne dokumentacije in podrobnejših načrtov računalniške rešitve. Razvijalci ne načrtujejo splošne zasnove programa, ki bi omogočala poznejše dopolnjevanje (Bruegge, Dutoit, 2004, str. 661). Razvijalci se osredotočajo na načrtovanje in reševanje posamičnih sprememb, ne pa na hkratno reševanje več sprememb. Izpostavljen je pomen skupinskega dela in pogoste komunikacije z naročnikom. Merilo uspešnosti je trenutna zadovoljitev uporabnikovih zahtev.

Uporabniki agilnih metodologij ne zanikajo uporabne vrednosti različnih načrtovalskih pripomočkov, vendar jih uvrščajo za delujočo programsko kodo. Ena od spletnih anket o uporabi agilnih tehnik razvoja v projektih izdelave računalniških rešitev je pripeljala do zaključka, da podrobna dokumentacija nima velike uporabne vrednosti v skupinah, ki uporablja agilni pristop. Ni pa bil zanikan pomen dokumentacije, kot jo predpisujejo agilni pristopi [URL: http: //www.ambysoft.com/surveys/agileMarch2007.html]. (Anketa sloni na vzorcu 781 anketirancev, 69 % vzorca je uporabnikov agilnih tehnik, 52 % vzorca je razvijalcev programske opreme) Na osnovi ocenjevanja, kjer je bila možna najvišja ocena pet, so anketiranci, ki uporabljajo agilni razvojni pristop, rangirali uporabnost različnih delovnih pripomočkov, kot je razvidno iz preglednice (tabela 2.).

Tabela 2: Ocena uporabnosti nekaterih pripomočkov v agilnih metodologijah

| Delovni pripomoček | Ocena (najvišja = 5) |
|---|---|
| Delujoči program (angleško: Working Software) | 4.22 |
| Izvorna koda (angleško: Source Code) | 4.22 |
| Testi razvijalcev (angleško: Developer Tests) | 3.96 |
| Skiciranje na tabli (angleško: Whiteboard Sketches) | 3.90 |
| Seznam zahtevanih opravil (angleško: Iteration Task List) | 3.87 |
| Testi uporabnikov (angleško: Customer Tests) | 3.87 |
| Tehnična specifikacija (angleško: Arch Spec - High Level) | 3.66 |
| Poročila o napakah (angleško: Defect Reports) | 3.61 |
| Specifikacija zahtev (angleško: Requirements Spec - High Level) | 3.52 |
| Primeri uporabe (angleško: Use Cases - Light) | 3.52 |
| Načrt testiranja (angleško: Test Plan) | 3.39 |
| Dinamično spremljanje projekta (angleško: Burn Down Chart) | 3.35 |
| Izdelava maket (angleško: Paper models) | 3.22 |
| Podrobni primeri uporabe (angleško: Use Cases - Detailed) | 2.96 |
| Podrobna specifikacija zahtev (angleško: Requirements Spec - Detailed) | 2.90 |
| Podrobna tehnična specifikacija (angleško: Arch Spec - Detailed) | 2.88 |
| Gantogrami (angleško: Gantt Chart - High Level) | 2.70 |
| Podrobni gantogrami (angleško: Gantt Chart - Detailed) | 2.21 |

Vir: [URL: http: //www.ambysoft.com/surveys/agileMarch2007.html]

Agilne tehnike izdelave računalniških rešitev tudi v izdelavi dokumentacije postavljajo v ospredje njeno učinkovitost. Med merila učinkovitosti se uvrščajo naslednji kriteriji: [URL: http: //www.agilemodeling.com/essays/agileDocumentation.htm#WhenIsADocumentAgile]

- agilni dokumenti omogočajo optimizacijo vložkov naročnika,
- naročniki morajo poznati celoten strošek zahtevane dokumentacije,
- agilna dokumentacija je oskubljena in zadostna,
- zadošča namenu,
- opisuje stvari, ki jih je dobro vedeti,
- agilni dokumenti imajo ciljnega uporabnika in olajšujejo njegovo delo,
- agilni dokumenti so dovolj natančni, skladni in podrobni za svoj namen.

Na osnovi navedenih kriterijev lahko sklepamo, da dokumentacija agilnih metodologij, kolikor je sploh je, v glavnem nastaja v poznejših razvojnih fazah, kot so kodiranje in dokumentiranje navodil, manj pa v fazah načrtovanja ali testiranja rešitve.

Lahko bi rekli, da je značilnost agilnih metodologij izdelava prototipov, kjer so slednji delujoče računalniške rešitve. Prototip je orodje modeliranja, na katerem preizkušamo ideje, konkretiziramo razmišljanja, preizkušamo odločitve in ustreznost končnih programskih rešitev (Kovačič, Bosilj-Vukšić, 2005, str. 266). Model prototipnega razvoja računalniških rešitev je bil poznan še pred objavo agilnega manifesta. Lahko bi rekli, da so agilne metodologije njegova nadgradnja, zato si oglejmo nekaj načel prototipnega razvoja računalniških rešitev. Po metodi prototipa razvoj poteka v štirih fazah (Kovačič, Bosilj-Vukšić, 2005, str. 267):

- definicija izhodiščnih informacijskih potreb oziroma potreb uporabnika,
- razvoj prototipne rešitve,
- uporaba in ocena prototipa za izpopolnitev uporabnikovih zahtev,
- spreminjanje oziroma izboljšava prototipa.

Prototipni pristop ima uporabno vrednost tudi pri razvoju baze podatkov in delujočih računalniških rešitev. Predpostavlja uporabo določene tehnologije (programskih jezikov in baz podatkov). Zagovorniki takšnega pristopa pogosto izhajajo iz predpostavke, da uporabniki sami niso sposobni opredelitve poslovnih pravil in informacijskih potreb, dokler rešitve v takšni ali drugačni obliki ni mogoče preizkusiti (Kovačič, Bosilj-Vukšić, 2005, str. 266).
Metoda je smotrna in priporočljiva (Gradišar, Resinovič, 2001, str. 432):

- kadar druge metode ne omogočajo pravočasne izdelave sistema,
- kadar uporabnik ne ve natančno, kaj potrebuje,
- kadar sistem lahko omejimo na obvladljivo velikost ali podsisteme,
- kadar je pomembna preprostost in prijaznost uporabe,
- kadar želimo testirati idejo,
- kadar je sistem predviden za generiranje poročil ali kot podpora za odločanje,
- kadar je predvidena manj pogosta uporaba.

Njeno uporabo se odsvetuje v naslednjih primerih:

- kadar informatik ne pozna dovolj dobro orodja, s katerim gradi rešitev,
- kadar podatki, potrebni za rešitev, niso preprosto dosegljivi (hiter dostop do podatkov je zelo pomemben za metodo),
- kadar obstoječa programska oprema v organizaciji ni sistematično vzdrževana,
- kadar vodstvo organizacije ne kaže zanimanja za takšen način dela,
- kadar uporabnik ni pripravljen sodelovati pri razvoju rešitve ali mu posvetiti dovolj časa.

Prototipi so lahko vsestransko uporaben pripomoček pri razvoju programske opreme, neodvisen od izbrane metodologije razvoja. Metoda izdelave prototipov se pogosto uporablja tudi kot

začetna faza pri razvoju sistema po načelu življenjskega cikla (Gradišar, Resinovič, 2001, str. 432). Praktične izkušnje s področja informatizacije poslovanja kažejo na učinkovitost in smotrnost prototipnega pristopa v povezavi z modeliranjem podatkov. Uporabno vrednost ima tudi v raziskovalnem okolju kot metoda preizkušanja na modelih in kompleksnih sistemih (Kovačič, Bosilj-Vukšić, 2005, str. 267).

Kakor koli že gledamo na oba razvojna pristopa, lahko rečemo, da imata veliko skupnega. Njuna temeljna skupna lastnost je intenzivno sodelovanje izvajalcev končne rešitve z naročnikom oziroma uporabnikom in ciklično izboljševanje funkcionalnosti delujoče računalniške rešitve.

#### 4.3.4.3 Katera izbira je prava

Kaskadne in agilne razvojne pristope danes pogosto primerjamo med seboj in ocenjujemo njihovo primernost. Prve pogosto opredeljujemo kot težke, disciplinirane, predvidljive, druge pa kot lahke, razpuščene in prilagodljive. Predvidljive metode so osredotočene na podrobno načrtovanje realizacije programske opreme, ki bo narejena v prihodnosti. Razvojni proces v času svojega trajanja ponuja pregled nad vsemi nalogami, ki jih izvajamo, in lastnostmi prihodnje programske opreme. Spremembe zahtev so nezaželene in jih težko uvajamo. To je tudi razlog, da morajo novi zahtevki ali spremembe običajno prestati strogo presojo in dokazovanje svoje upravičenosti. Prilagodljive (agilne) metodologije se posvečajo predvsem odzivnosti na spremenjene zahteve. V uporabi teh metodologij je dokončno realizacijo težko napovedati. Potrebne funkcije in prihodnje lastnosti programske opreme lahko načrtujemo le za kratek čas naprej.

Poleg zagovornikov agilnih pristopov razvoja, se pojavljajo tudi kritike. Pojavljajo se očitki o pomanjkanju izkustvenih dokazov, ki bi podpirali trditve njihove uporabnosti ter njihovo pomanjkanje teoretične utemeljenosti (Abrahamsson, 2003).

Na osnovi študije primerov projektov, ki so uporabljali XP razvojno agilno metodologijo (Bruegge, Dutoit, 2004, str. 665-684) in neizkušenega projektnega vodjo, ki je lahko izbiral med različnimi metodami, je bilo zaključeno, da so agilni pristopi bolj učinkoviti takrat, ko vsi ključni udeleženci delijo isti cilj in koncept zasnove sistema bodisi zaradi vizije naročnika bodisi zaradi sistema samega. Po drugi strani je bilo ocenjeno, da je hierarhična organiziranost projekta bolj primerna v pogojih, ko udeleženci vzpostavljajo množico navzkrižnih zahtev in tekmujejo za omejene vire.

V splošnem lahko črpamo nekaj usmeritev pri izbiri prave metodologije tudi iz hevristike oziroma spoznanj pri uporabi različnih metodologij, od koder izhajajo naslednje ugotovitve (Bruegge, Dutoit, 2004, str. 688):

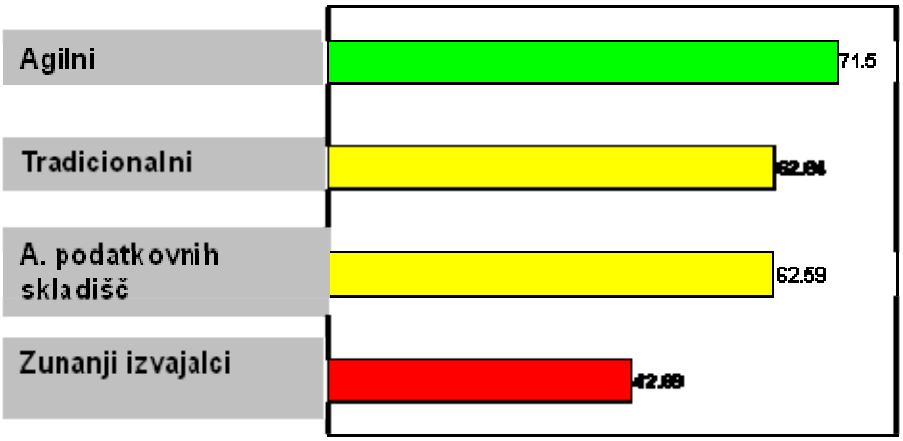
- bodimo pripravljeni spremeniti sistem: spremembe se bodo v projektu vedno dogajale tako s strani uporabnikov kot izvajalcev;
- bodimo pripravljeni spremeniti organizacijo projekta: hierarhične organizacije so bolj predvidljive in učinkovite v izvedbi projektov;
- bodimo pripravljeni spremeniti proces: vizija na osnovi izkušenj olajšuje prilagajanje procesa;
- skrajšajmo odzivni čas naročnika: naročnik (uporabnik) je pogosto udeležen v večjih spremembah projekta. To zahteva učinkovit mehanizem komunikacije in odzivnost uporabnikov, ki se spoznajo na domeno problema;

- ustvarjajmo zaupanje: udeleženci se pogosto izogibajo javljanju napak, še zlasti, če so sami krivci za njih. Zgodnje javljanje napak je lahko ločnica med uspehom in neuspehom projekta, zato je bolj kot za kaznovanje treba poskrbeti za nagrajevanje odkrivanja napak;
- naučimo se pravih stvari: koncentracija znanja samo na omejenem številu udeležencev lahko projektu škodi. Odmik od organizacije projekta v slogu zahteva-nadzor v organizacijo vzajemnega sodelovanja lahko zahteva spremembo vloge iz projektnega vodje v učitelja, ki poskrbi za ustrezno znanje udeležencev.

V razvoju programske opreme sledimo končnemu cilju, izdelavi kakovostne opreme, izdelane v predvidenih časovnih in finančnih okvirih. Za doseganje tega cilja lahko uporabimo takšno ali drugačno metodologijo, ki predlaga uporabo določenih metod (receptov) in orodij v odvisnosti od razmer, v katerih smo. Lahko zaključimo, da vsaka metodologija omogoča prilagajanje aktivnosti in uporabe orodij okolju, v katerem izvajamo razvoj.

Na osnovi spletne raziskave, v kateri je sodelovalo 586 anketirancev, [URL: http: //www.ambysoft.com/surveys/success2007.html] o stopnji uspešnosti projektov informacijske tehnologije, je bilo po mnenju anketiranih ugotovljeno, da agilno zasnovani razvojni projekti izkazujejo 71,5 % uspešnost, projekti, zasnovani na tradicionalnem razvojnem ciklu, 62,8 % uspešnost, projekti, zasnovani na analizi podatkovnih skladišč, 62.59 %, zunanja izvedba razvojnih projektov pa izkazuje 42,7 % uspešnost.

Slika 4: Ocena uspešnosti projektov IT v odvisnosti od metodološke zasnove in izvedbe (v %)




Vir: [URL: http: //www.ambysoft.com/surveys/success2007.html]

Vzorca anketirancev navedene spletne ankete sicer ne moremo obravnavati kot reprezentativnega vzorca uporabnikov razvojnih metodologij. Lahko pa bi rekli, da je ocena uspešnosti projektov, izvedena po tradicionalnih ali agilnih metodologijah, na prvi pogled medsebojno primerljiva.

Iz povedanega lahko zaključimo, da je izbiro metodologije treba predvsem prilagajati okolju, v katerem jo izvajamo. Okolje določa mejne elemente, s katerimi se sooča projektni vodja ob začetku projekta. Med te elemente štejemo uporabnike, naročnike, bolj ali manj izkušene razvijalce, izkušnje drugih udeležencev, časovne roke, organizacijo izvedbe projekta, rezervni načrt za izvedbo projekta, vrsto problema, ki ga rešujemo, in lokacijo projekta. Utemeljeno lahko predpostavljamo, da bo na značaj izbrane metodologije vplivalo tudi znanje ključnih nosilcev izvedbe projekta. Nosilci izvedbe, ki imajo več programerskih izkušenj, bodo favorizirali agilne

pristope. V okoljih s tradicijo, usmerjeno v načrtovanje izvedbe projektov, bodo verjetno bolje sprejete tradicionalne metode. S pomočjo selektivne izbire metod, predlaganih znotraj različnih metodologij, možnostjo njihove izbire in prilagoditve izbranemu okolju se projektni vodja lahko dokoplje do prave izbire, ki bo vodila do uspešnega zaključka projekta (Bruegge, Dutoit, 2004, str. 646-47). Lahko bi rekli, da je eden ključnih dejavnikov uspešne izvedbe projekta razvoja programske opreme še vedno vodja projekta, ki organizira, nadzira, usmerja in pomaga udeležencem pri izvedbi projekta.

Razvoj informacijskih sistemov združuje sodelovanje poklicnih informatikov, prihodnjih uporabnikov in vodij procesov razvoja. Slednji so odgovorni tudi za skladnost vsebine prihodnjega sistema s strateškimi in taktičnimi načrti organizacije. Prihodnji uporabniki sodelujejo v razvoju tako, da podrobno določijo svoje zahteve in s tem opredelijo vsebino sistema oziroma računalniške rešitve. Ta vključuje opredelitev vhodnih in izhodnih podatkov ter pravila njihovega preoblikovanja. Informatiki poskrbijo za uporabo ustrezne tehnologije in izdelavo računalniških rešitev (Gradišar, Resinovič, 2001, str. 422). Če končni uporabnik ni točno znan (spletne storitve za širši krog uporabnikov), je treba vloge in zahteve funkcionalnosti, ki jim jo pripišemo, utemeljiti (vzpostaviti) na drug način. Morda bo treba zasnovo računalniške podpore podrobneje načrtovati in izpeljati v skladu z drugimi načrti organizacije.

V organizaciji, ki živi in je odvisna od razvoja računalniških rešitev, izbiranje in uporabo ene od razvojnih metodologij lahko uvrstimo na raven strateškega iskanja novih razvojnih priložnosti. Kadar bi za način delovanja svoje organizacije vzpostavili kriterije, kot so doseganje visoke stopnje tehnološkega (informacijskega) znanja, možnost hitrega odzivanja na spremenjene zahteve naročnika, učinkovitost komunikacije z naročnikom, možnost hitre izvedbe računalniških rešitev, potem bi zanesljivo morali preučiti možnost uvedbe ene od agilnih razvojnih metodologij.

Če razmišljamo o uporabi metodologije za razvoj računalniških rešitev v okviru potreb organizacije, v kateri delujemo, bomo primernost razvojne metodologije praviloma presojali tudi s stališča njenih zahtev in kadrovskih virov, ki jih imamo na voljo. Uporaba metodologije, najbolj pa njena izvedba, je namreč v tesni povezavi s kulturo organizacije, razpoložljivimi kadri, stopnjo računalniškega (tehnološkega) in projektnega znanja, izkušnjami skupinskega dela, načini vodenja ter učinkovitostjo mehanizmov notranje komunikacije in izmenjave znanja.

Dodaten kriterij, ki bi lahko pomembnejše vplival na primernost določene razvojne metodologije, je morebitna zahteva po obsežnejši izdelavi spremljajoče dokumentacije. Sem štejemo tako dokumentiranje zahtev naročnika, podrobne načrte končne rešitve, dokumentacijo, ki spremlja programsko kodo, kot poročila testiranja in dodatna procesna navodila, s katerimi lahko kombiniramo navodila za uporabo računalniške rešitve. Takšne zahteve se lahko pojavijo, če želimo z razvojem računalniške rešitve podpreti še kakšen drug proces. Kot primer lahko obravnavamo zahtevo po razvoju računalniške rešitve in sočasno prenovo poslovnih procesov.

Sam proces razvoja računalniške rešitve lahko tudi razdelimo na podprocese, ki jih želimo izvajati kombinirano, v lastni režiji in ob zunanji pomoči. Drug primer je lahko proces povečevanja baze znanja v organizaciji. Z načrtno izdelavo dokumentacije, ki jo zahtevajo nekatere razvojne metodologije, lahko zagotovimo višjo raven skupinskega in organizacijskega znanja (način izvedbe procesov, navodila, vzorci delovanja). V podobnem kontekstu lahko obravnavamo modele delovanja organizacije ter uporabo poenotenega jezika za modeliranje (UML), ki ga favorizirajo nekatere razvojne metodologije. Uporaba izbrane metodologije razvoja namreč lahko s pripomočki, ki jih predpisuje, na različne ravni delovanja organizacije vnese zahteve po enotnih pripomočkih za komunikacijo in izmenjavo znanja.

Pri odločanju o uvedbi ene od razvojnih metodologij v lasten razvoj računalniških rešitev, bi lahko zaključili, da ima izbira ene od klasičnih razvojnih metodologij v nekaterih primerih našega delovanja določene prednosti. V tem pogledu bi lažje utemeljili večjo uporabno vrednost ene od klasičnih metodologij razvojnega cikla (kaskadne), če povežemo proces razvoja izdelave računalniške rešitve še z drugimi procesi v organizaciji ali, če delujemo v organizaciji, ki v svojem delovanju sledi modelu učeče organizacije.

### 4.3.5 Procesni vzorci (angleško: process patterns)

Nekateri avtorji predlagajo izvajanje procesa razvoja programske opreme v obliki procesnih vzorcev (Ambler, 1998c). Osnovna ideja predpostavlja razvoj kot proces, ki ga ne moremo natančno predpisati. Lahko pa izvedbo procesa predpišemo na splošno s stališča splošnih problemov ali opravil, ki jih je potrebno izvesti za uspešno dokončanje razvojne naloge. Temu so namenjeni procesni vzorci. Ambler priznava tudi druge skupine vzorcev, na primer vzorce analize, vzorce načrtovanja in organizacijske vzorce. Prav slednje omenja kot tiste, na katere se je treba opirati pri uporabi procesnih vzorcev. Z uporabo organizacijskih vzorcev, ki opisujejo splošne organizacijske in izvedbene pristope, naj bi namreč uspešneje upravljali s človeškimi viri, udeleženimi v razvoju programske opreme.

Če se vrnemo na procesne vzorce, naj bi slednji predstavljali preizkušen in uspešen pristop, ki opisujejo niz aktivnosti za izvedbo določene faze v razvoju programske opreme. Njihovo nasprotje so procesni antivzorci, ki opisujejo skupine aktivnosti v razvoju, za katere je izkustveno dokazano, da so neučinkovite. Procesne vzorce lahko delimo na nekaj tipov (Ambler, 1998c, str. 6):

- vzorec izvedbe projektne faze (angleško: phase pattern): Projektne faze so vrhnje (najbolj abstraktne) komponente razvojnega procesa. Sem sodijo začetna faza, faza sestavljanja, faza izdelave ter faza vzdrževanja in podpore. Izvedba posamične faze je opisana z izvedbo projektnih etap. Tako so, na primer, za izvedbo projektne faze sestavljanja predlagane etape modeliranje, posplošitev, programiranje in delno testiranje;
- vzorec izvedbe projektne etape (angleško: stage pattern): procesni vzorec, s katerim predvidimo izvedbo etape znotraj projektne faze. Izvedba vsake vrhnje komponente procesa (projektne faze) je deljena na več etap. Etape izvajamo v iteracijah korakov, predvidenih z vzorcem izvedbe projektne etape;
- vzorec izvedbe opravila (angleško: task pattern): ta tip procesnega vzorca opiše podrobne korake za izvedbo specifične naloge.

Vendar pa Ambler izhaja iz predpostavke, da je vzpostavitev projektnih faz in etap samo sestavni del, ki sam po sebi ne zadošča za izvedbo objektnega procesa razvoja programske opreme. Kot ključne elemente za uspešno izvedbo projekta avtor navaja vzporedno izvajanje vzajemno prepletenih nalog s področij podpornih procesov (Ambler, 1998c, str. 43-48):

- zagotavljanja kakovosti: upoštevanje standardov, usmerjanje izdelave pravih stvari na pravi način;
- projektnega ravnanja: organiziranje, nadziranje in usmerjanje izvedbe projekta;
- upravljanja s človeškimi viri: organizacija, nadzor, vodenje in motiviranje udeležencev na povezovalen način;

- upravljanja tveganja: odkrivanje, spremljanje in odpravljanje tveganj, ki jih zaznajo udeleženci;
- upravljanja ponovne rabe: organizacija, nadziranje in usmerjanje izvedbenih podrobnosti ali izdelkov projekta na način, ki bo omogočal ponovno rabo;
- izobraževanja in usposabljanja: z razširjanjem znanja vnašamo razumevanje o potrebnih nalogah med udeležence, usposabljanje pa omogoča pridobivanje potrebnih veščin za izvedbo dela;
- merjenja doseženih ciljev: merjenje in analiza omogočata nadzor nad kakovostjo in učinkovitostjo opravljenega dela z namenom izboljšave tako procesov kot njihovih končnih produktov;
- upravljanja izdelkov in pripomočkov projekta: vse oprijemljive, vmesne in končne pripomočke, ki so nastali med izvedbo projekta, smatramo za ključne sestavine uspešnega vzdrževanja in poznejše podpore programske rešitve;
- upravljanja z infrastrukturo projekta: infrastruktura projekta so viri, s katerimi razpolagamo - osebje, razpoložljiva orodja, privzeti standardi in navodila, standardni razvojni proces, če ga uporabljamo. Pri vzpostavitvi infrastrukture opredelimo udeležence projektne skupine, njihove vloge in odgovornosti, seznam orodij, vzpostavitev (obnova) skupinskega znanja, potrebnega za izvedbo projekta, in prikrojitev izvedbe procesa projektni skupini.

Avtor še zlasti opozarja na pomembnost kompleksne obravnave razvojnega procesa z vidika izvajanja navedenih nalog. Navzkrižno izvajanje nalog podpornih procesov znotraj faz projekta po avtorjevem mnenju predstavlja tisto vezivo, ki omogoča uspešno dokončanje projekta. Skladno s predstavljenimi koncepti avtor predlaga različne vzorce za izvajanje etap v fazah projekta. Vzorci vsebujejo izvedbena opravila, ki jih iterativno izvajamo (ponavljamo) znotraj vsake etape. Seznam predlaganih opravil okvirno sledi načelom izvedbe dela po modelu razvojnega cikla. Tipični vzorec izvedbe etape obsega naslednje sestavine (Ambler, 1998c, str. 277-345):

- opredelitev vhodnega konteksta: opredelitev vhodnih elementov oziroma pogojev, ki jih je treba upoštevati v izvedbi etape;
- predstavitev rešitve: opredelitev metod, pripomočkov in konceptov, ki jih bomo uporabljali znotraj etape;
- predstavitev projektnih nalog: izdelava seznama projektnih nalog. Iz nabora podpornih procesov oziroma iz celotnega seznama podpornih nalog izberemo tiste, ki po svojem namenu in vsebini sodijo v izvedbo izbrane etape;
- opredelitev izhodnega koncepta: opredelitev izdelkov in rezultatov, proizvedenih v posamezni etapi, in opredelitev pogojev, pod katerimi ocenjujemo, ali je etapa uspešno zaključena;
- seznam dejavnikov uspeha: navedeni so dejavniki, ki po mnenju avtorja bistveno pripomorejo k uspešni izvedbi etape;
- kontrolni seznam aktivnosti in pripomočkov: vsak vzorec vsebuje celotni seznam vhodnih elementov, pripomočkov, predvidenih procesov oziroma nalog in izhodnih elementov, udeleženih v etapi. Seznam predstavlja referenčno zbirko entitet, udeleženih v izvedbi etape. Uporabljamo ga kot kontrolni mehanizem.

Procesni vzorci so lahko potencialno izjemno koristen pripomoček za tista razvojna okolja, ki želijo izdelati lasten vzorec razvoja računalniške rešitve v obliki projektne naloge, okolje samo pa nima veliko projektnih izkušenj. Uporaba procesnih vzorcev, glede na svoj koncept, omogoča (načeloma poljubno) kombiniranje izbranih metod razvoja pod okriljem etap projekta, z uporabo

(načeloma poljubnih) nalog različnih podpornih procesov (zagotavljanje kakovosti, dokumentacije, ponovne rabe elementov, merjenja doseženih ciljev itn). Kontrolni seznam aktivnosti in pripomočkov, predlagan v vsaki etapi, lahko v končni fazi odigra dvojno vlogo, prvič pri organizaciji, nadzoru in usmerjanju dela, drugič kot preizkušena predloga (vzorec) za izvedbo novih projektov.

Ocenjujem, da v organizaciji, kjer projektno delo nima tradicije ali kjer zaposleni zavračajo sodelovanje na projektih zaradi tega, ker ocenjujejo svoje znanje kot pomanjkljivo, lahko procesne vzorce uporabimo kot koristen pripomoček. Uporabimo jih lahko kot praktičen (podrobno opisan) primer ogrodja za organizacijo projekta razvoja računalniške rešitve. Z njihovo pomočjo lahko izdelamo lasten procesni vzorec, prilagojen znanju in zastavljenim ciljem udeležencev razvoja računalniške rešitve. Procesni vzorci nas učijo, kako lahko razgradimo in ponovno medsebojno povežemo kompleksne in vzajemno povezane procese na pregleden in obvladljiv način. Na ta način kompleksna naloga postane pregledna in na prvi pogled bolj obvladljiva. Kompleksna znanja (izvedba procesov) razpadejo na nabor nalog, ki jih izvajamo v kontekstu izbranega glavnega in vključenih podpornih procesov. Tudi obseg manjkajočega znanja lahko postane s tem bolj obvladljiv, saj dobi izvajalec, ki mu delegiramo del nalog, boljši vpogled v obseg pričakovanega (manjkajočega) znanja.

### 4.3.6 Procesna ogrodja (angleško: process frameworks)

Nekateri avtorji (Graham, Henderson-Sellers, Younessi, 1999) izpostavljajo, da je treba proces razvoja programske opreme že v samem začetku prilagoditi z vzpostavitvijo organizaciji prirejenega specifičnega procesa. Prikrojitev razvojnega procesa specifičnim pogojem sloni na predpostavki, da v projektu razvoja programske opreme nastopa na desetine spremenljivk, med katere uvrščamo orodja, programske jezike, ljudi, procese, cilje kakovosti ter velikost projekta in jih ne moremo uskladiti na nivoju predpisa enega samega univerzalnega procesa (Henderson-Sellers, 2000, str. 56).

Slika 5: Proces razvoja programske opreme, metodološki okvir OPEN

| **Raven poslovnih procesov** | **Proces izdelave programske opreme** | **Proces modeliranja** |
|---|---|---|
| Poslovne funkcije<br>+<br>Proces izdelave programske opreme<br>+<br>Uvedba računalniške rešitve v uporabo | Osebje / organizacijska kultura<br>+<br>Orodje / razpoložljiva tehnologija<br>+<br>Metodologija | Modeliranje<br>+<br>Tehnike<br>+<br>Jezik modeliranja |

Vir: Prirejeno po Henderson-Sellers, 2000, str. 55, Fig. 1

Konec devetdesetih let prejšnjega stoletja je bil vzpostavljen splošen metodološki okvir razvoja programske opreme tretje generacije pod imenom OPEN (angleško: Object-oriented Processs, Environment, Notation). Vzpostavljeno je bilo tudi neprofitno združenje (OPEN) z namenom, da člani prispevajo razvojne metode za splošno uporabno, standardizirano metodologijo razvoja programske opreme.

Člani konzorcija izhajajo iz predpostavke, da je metodologija lahko zasnovana le kot skupina ohlapno povezanih procesov, ki jih je mogoče prilagajati izkušnjam izvajalcev, organizacijski kulturi in specifičnim zahtevam okolja, kjer je uporabljena.

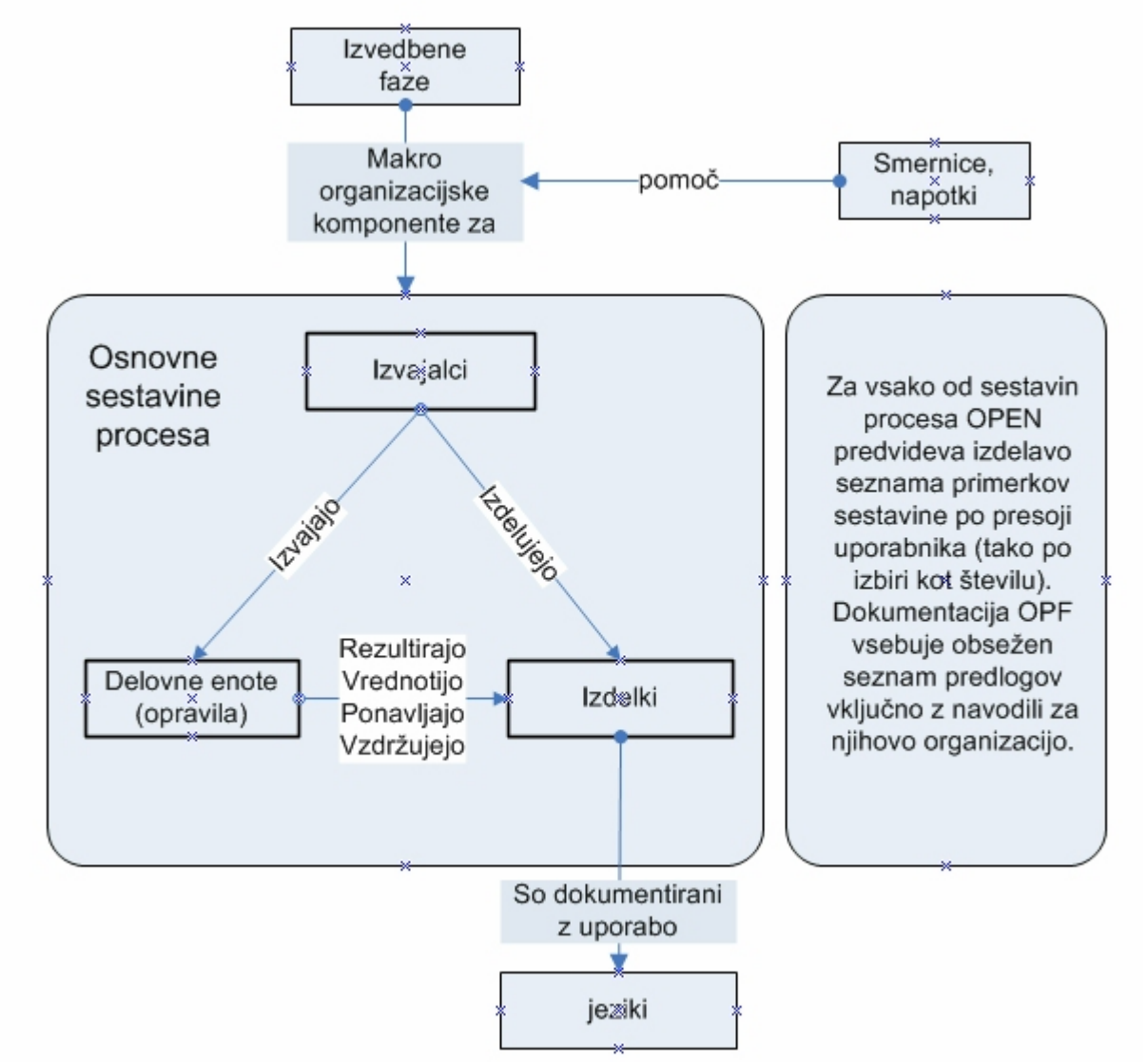
Izhodišče za prilagoditev metodologije specifičnemu okolju predstavlja odprto procesno ogrodje (angleško: Open process framework, OPF). Zamišljeno je kot standardiziran pristop za izdelavo metodologije življenjskega cikla. Temelji na knjižnici komponent, ki jih lahko dopolnjujemo in prikrojimo za uporabo v okviru specifičnega projekta. Knjižnica komponent za izdelavo metodologije vsebuje naslednje tipe procesnih komponent (Zowghi, Firesmith, Henderson-Sellers, 2005, str. 61-63):

- celovita rešitev oziroma končna naloga (angleško: endeavor): abstraktna komponenta, s katero modeliramo ciljno usmerjeno delovanje, v katerem producenti vzajemno sodelujejo in s svojim vložkom delovnih enot razvijajo delovne produkte oziroma storitve. Z opredelitvijo celovite naloge običajno prevzamemo odgovornost za izvedbo nekega končnega proizvoda (storitve), ki ga lahko zasnujemo na enem ali več delovnih programih, slednje pa na enem ali več projektih;
- izvedbena faza (angleško: stage): abstraktna delovna komponenta za modeliranje časovnega okvira izvedbe časovno zaključenih in povezanih nizov delovnih enot. Med izvedbene dobe uvrščamo tiste, ki jim lahko pripišemo trajanje (cikel, faza, izvod) in tiste, ki nimajo trajanja (mejnik, korak);
- delovni proizvodi oziroma izdelki (angleško: work products): abstraktne komponente, ki so rezultat izvedbe delovnih enot oziroma vloženega dela izvajalcev in imajo uporabno vrednost za izvedbo končne naloge. Ločimo paket izdelkov in verzijo izdelka. Paket izdelkov predstavlja vzajemno povezan nabor izdelkov (na primer tistih, ki nastanejo ob izvedbi ene od aktivnosti). Verzija proizvoda predstavlja oznako končnega izdelka v določenem trenutku razvojnega procesa. Kot proizvod lahko obravnavamo enega od pojmov, kot so: aplikacija, arhitektura, zahteva, model, diagram, dokument, podatkovna, programska ali strojna komponenta. Paketi izdelkov sodijo skupaj po vsebini in so običajno rezultat aktivnosti, kot so zasnova arhitekture, izdelava načrta, diagramov, modelov, zahtev in drugih;
- jezik (angleško: languages): abstraktna komponenta, ki sloni na slovarju in slovničnih pravilih (sintakse in semantike) in jo uporabljamo za izdelavo delovnih proizvodov oziroma izdelkov. Na osnovi uporabljenih jezikov omogočimo delovnim proizvodom komunikacijo z ljudmi ali aplikacijami. Med jezike uvrščamo pogovorne, modelirne, izvedbene in druge jezike;
- izvajalec (angleško: producer): abstraktna komponenta, s katero modeliramo nekoga ali nekaj, kar izvaja delovne enote, na osnovi katerih izdelamo delovne produkte oziroma storitve. Ločimo posredne (organizacija, skupina, vloga) in neposredne (oseba, orodje) izvajalce. Z izvajalci modeliramo odgovornosti udeležencev in njihovo sodelovanje v izvedbi končne naloge;
- delovna enota ali opravilo (angleško: work unit): abstraktna komponenta predstavlja delovno enoto, s katero modeliramo izvedbo funkcionalno zaključenega obsega dela izvajalcev. Delovno enoto (opravilo) lahko delimo na manjše delovne enote, kot so aktivnosti in zadolžitve. Aktivnost predstavlja zbirko medsebojno povezanih zadolžitev,

ki jih opravljajo izvajalci. Rezultat izvedbe aktivnosti je medsebojno povezan nabor delovnih proizvodov. Med delovne enote sta uvrščena tudi delovni tok in tehnika. Delovni tok predstavlja urejeno zaporedje zadolžitev, s katerimi izdelamo posamezen delovni proizvod. Tehnika predstavlja način izvedbe zadolžitve.

Procesno ogrodje je metamodel, na podlagi katerega lahko vzpostavimo organizaciji prirejen specifičen proces (slika 6.). Na nivoju metamodela so temeljni abstraktni gradniki procesa celovita rešitev, izvedbena faza, izvajalec, opravilo, izdelek in jezik. Predstavljajo standardiziran terminološki koncept za izvedbo procesnega inženiringa. Z njihovo pomočjo izpeljemo seznam konkretnih komponent tako za celoten proces kot posamezne metode. Glede na razmere in pogoje, v katerih izvajamo razvoj programske opreme, si lahko pripravimo specifičen razvojni proces, ki sloni na našem izboru vlog udeležencev, aktivnosti, nalog, izdelkov in tehnik.

Slika 6: Osnovne sestavine procesnega ogrodja OPEN




Vir: Povzeto po Firesmith and Henderson-Sellers, 2000

Predlagan pristop prikrojevanja procesa našim potrebam izhaja iz predpostavke, da je proces razvoja programske opreme projektna naloga, ki jo izvajamo na osnovi merljivih opravil ob

uporabi ustreznih orodij. Aktivnost v tem procesu predstavlja smernico oziroma dolgoročni cilj, primerljiv s pojmom faza v inkrementalnem ciklu razvoja. Označuje vsebino, kaj moramo narediti, ne pa tega, kako to naredimo. Zaradi daljšega trajanja in širšega pomena jih je težje upravljati, zato potrebujemo bolj podrobno členitev vsebine. To lahko naredimo z navedbo seznama zadolžitev (angleško: tasks). So najmanjše merljive enote dela, določene znotraj razvojnega projekta. Vendar naloge še vedno ne povedo, kako naj delo opravimo. To vlogo v predlaganem procesu ima tehnika (angleško: technique). S tehniko opišemo, kako lahko uporabimo metodo objektne tehnologije oziroma njen koncept. Povezovanje nalog in tehnik izvajamo s pomočjo predpisane prikrojevalne matrike.

Prikrojevalna matrika prikazuje relacije med nalogami in tehnikami. V matriki označimo obvezno kombinacijo (angleško: M-madatory), priporočljivo (angleško: R- recommended), opcijsko (angleško: O-optional), neprimerno (angleško: D-discouraged) in prepovedano (F-forbidden). Izdelava matrike sodi med pomembnejše opravilo v izdelavi procesa na osnovi ogrodja OPEN.

Delovni produkti (angleško: work products) so rezultat v aktivnostih izvedenega dela. To so lahko dokumenti, komponente, programska oprema. Gleda na to, da je tudi v OPEN procesnem pristopu predlagan inkrementalen in iterativen pristop dela, imajo delovni produkti značaj delnega končnega izdelka. Izdelki nastajajo skozi uporabo različnih tehnik v času ponavljanja različnih aktivnosti. Delni izdelki so izhodno-vhodne količine v nizu povezanih aktivnosti. Uporaba modelirnega jezika je obvezna. Avtorji predlagajo OML (OPEN modeling language) ali pa uporabo standardnega jezika modeliranja UML.

Slika 7: OPF, prikrojevalna matrika zadolžitev in tehnik

| | Zadolžitve | | | | | |
|---|---|---|---|---|---|---|
| Tehnike | A | B | C | D | E | . . |
| 1 | M | D | F | F | F | . |
| 2 | D | D | F | F | D | . |
| 3 | D | D | O | O | D | . |
| 4 | F | O | O | O | F | . |
| 5 | F | M | O | D | F | . |
| 6 | R | R | M | R | O | . |
| 7 | D | R | F | M | O | . |
| 8 | D | F | M | D | D | . |
| 9 | R | R | D | R | R | . |
| 10 | O | D | O | O | R | . |
| 11 | F | M | O | F | D | . |
| . | . | . | . | . | . | . |

Vir: Povzeto po Henderson-Sellers, 2000, str. 57, Fig. 2

Na osnovi razumevanja procesnega ogrodja OPEN lahko za specifično okolje in potrebe (oblikovanja lastnega procesa razvoja) uporabimo komponente iz obsežnega repozitorija komponent, ki so objavljene na spletu. Različne komponente so uporabnikom prosto dosegljive preko metamodela procesnega ogrodja OPEN. Njegovi avtorji opozarjajo na univerzalnost predlaganega metamodela, ki je uporaben tako za opisovanje splošnih razvojnih metodologij (od manjših agilnih do večjih kompleksnejših) kot tudi za oblikovanje specifičnih, prikrojenih zahtev okolja, v katerem jih izvajamo.

Predlagatelji podajajo tudi smernice, s katerimi v okviru oblikovanja (angleško: construction) lastnega procesa za razvoj programske opreme opravimo ustrezen izbor med komponentami, s pomočjo katerih si obetamo doseči zastavljeni končni cilj (Zowghi, Firesmith, Henderson-Sellers, 2005, str. 65).

Če med seboj primerjamo pristop razvoja programske opreme na osnovi procesnih vzorcev in predlagani pristop na osnovi procesnega ogrodja, lahko rečem, da sta pristopa relativno podobna. Razvoj na osnovi procesnih vzorcev temelji na predpisovanju izvedbe preizkušenih delnih procesov (faz) in povezanih aktivnosti (etap) in opozarja na ključne aktivnosti (korake) za uspešno izvedbo razvojnih etap. Model razvoja s pomočjo procesnega vzorca v osnovi pušča proste roke pri zasnovi celotnega procesa. Zasnovo specifičnega procesa usmerja zgolj z navedbo priporočil o sestavinah (komponentah) in opozarjanjem na povezane aktivnosti. Res pa je, da bolj eksplicitno usmerja (predlaga) izdelavo delnih proizvodov in pripomočkov, izdelanih s pomočjo modelirnega jezika.

Bolj kot na propagiranje že preizkušenega procesa, se osredotoča na propagiranje in razlago uporabe preizkušenih komponent (sestavin), s katerimi sami zgradimo proces. Zaradi neprestanega osveževanja obsežnega repozitorija komponent, ki so dosegljive preko spleta, predstavlja poznavanje in uporaba procesnega ogrodja OPEN koristen pripomoček pri kombiniranem snovanju procesa in razvojne metodologije v izdelavi lastnih računalniških rešitev.

### 4.3.7 Zbiranje in analiza zahtev

Ožje področje aktivnosti, ki se ukvarja z odkrivanjem, analizo, dokumentiranjem in vzdrževanjem zahtev sistema razumemo pod pojmom inženiring zahtev (angleško: requirements engineering). Pri nekaterih metodoloških predlogih poteka ta skupina aktivnosti v obliki ene faze procesa, pri drugih pa v obliki dveh faz procesa, in sicer v obliki specifikacije in analize zahtev (Parviainen et al. 2003, str. 16-21).

Osnovni namen specifikacije zahtev je opredelitev funkcionalnosti in mejnih pogojev za delovanje sistema. Cilj zbiranja zahtev uporabnikov je doseganje dogovora z uporabniki oziroma naročniki, kaj naj bi sistem delal. Cilj zbiranja zahtev sistema je opredelitev pogojev za delovanje sistema. Izvedba specifikacije zahtev je opredeljena tudi s standardi. Vodnik za specifikacijo zahtev sistema predstavlja IEEE 1233-1998 standard. Opredeljuje prepoznavanje, organizacijo, predstavitev in dopolnjevanje zahtev z vidika poznejše vgradnje v faze načrtovanja in izvedbe.

Z zbiranjem zahtev definiramo obseg problema, ki ga rešujemo. Zahteva predstavlja posamično, stvarno funkcionalnost, za katero pričakujemo, da jo bo izvedel sistem ali ponujena programska rešitev. Z zahtevami opredelimo tudi nefunkcionalne lastnosti ali značilnost, za katero želimo, da jo sistem vsebuje. Z navedbami zahtev opredelimo samo vsebino (kaj) in ne predpisujemo izvedbe (kako). Ugotavljanje in razvrščanje zahtev sistema izvajamo v fazi razvojnega procesa, ki jo imenujemo analiza zahtev.

Seznam zbranih zahtev je sredstvo, s katerim omogočimo vsakemu preučevalcu enak pogled na problem, ki ga rešujemo. Predstavlja osnovno komunikacijsko sredstvo med naročnikom in izvajalcem končne rešitve. Primaren način izražanja zahtev je navajanje v obliki tekstovnega dokumenta, ki ga lahko spremljajo priloge s podrobnejšimi ilustracijami in razlagami. Proces

zbiranja in analize zahtev je sestavni del vseh metodologij razvoja programske opreme. Lahko je zelo obsežen in naporen za izvedbo, zato je neprestano tarča pripomb in sprememb. Kritike navajajo veliko časovno potratnost in paralizo analize zaradi navajanja prevelikih podrobnosti. Spremembe segajo od zmanjševanja navajanja podrobnosti do njihovega postopnega opuščanja.

Zbiranje zahtev in analizo lahko obravnavamo kot proces, ki je odvisen od dejavnikov, kot so organizacija, vrsta programske opreme, izbrani proces sistemskega inženiringa in razvojnega procesa izdelave programske opreme.

Med glavne aktivnosti procesa zbiranja zahtev se uvrščajo (Parviainen et al. 2003, str. 12):

- izločanje zahtev skozi razgovore in sodelovanje z udeleženci v obravnavanem sistemu, zbiranje podatkov na osnovi razpoložljive dokumentacije, zbiranje znanja o problemski domeni, preučevanje tržnih raziskav in podobno;
- analiza zahtev in pogajanja, kjer zahteve podrobno preučimo in jih udeleženci v obravnavanem sistemu sprejmejo;
- preverjanje zahtev, kjer dokončno preverimo popolnost in skladnost ugotovljenih zahtev.

Pri novejših agilnih pristopih razvijalci in arhitekti sistema praviloma niso prisotni v fazi zbiranja zahtev kot jo poznajo tradicionalni pristopi (Withall, 2007, str. 8). Eden od osnovnih principov, ki izhaja iz manifesta agilnih metod, pravi, da izdelaj več programske opreme in manj dokumentacije. Agilne metode ne zanikajo koristnosti zbiranja zahtev sistema, vendar se ga lotevajo na drug način, s pozivom uporabnikom, naj napišejo svojo izjavo (angleško: user story) o tem, kaj naj bi sistem delal. Na osnovi teh izjav dobijo razvijalci in snovalci vpogled v funkcijske zahteve pričakovanega sistema. Agilne metode slonijo na vgradnji prepoznane funkcionalnosti v osnutek končne rešitve in na njenem postopnem izboljševanju glede na odzive uporabnika. Tudi agilno zbiranje zahtev temelji na postopnem (inkrementalnem) dopolnjevanju zahtev uporabnika (Withall, 2007, str. 9).

Ne glede na uporabljeno metodologijo razvoja programske opreme se v začetku vsakega snovanja srečamo z zahtevami oziroma pričakovanji naročnika. Razpoložljive razvojne metodologije programske opreme bi težko posebej klasificirali z vidika predlaganih tehnik in metod za opredeljevanje zahtev sistema. Različne metodologije v kontekstu svojega specifičnega koncepta predlagajo različne pristope k zbiranju zahtev sistema. Ponekod se bolj posvečajo pogojem za delovanje sistema, drugje manj. Zbiranje zahtev sistema se praviloma pri vseh metodah na takšen ali drugačen način začenja skozi proces zbiranja zahtev uporabnikov. Oglejmo si nekaj splošnih značilnosti tehnik in pristopov zbiranja zahtev kot jih predpisujejo različne metodologije.

Med splošne tehnike, ki so praviloma uporabne v vsaki metodologiji ali njeni fazi, štejemo pristope, kot so, na primer, viharjenje možganov (brainstorming), kontekstne poizvedbe, intervjuji, opazovanje pri delu ali izdelava scenarijev. Pri teh tehnikah zbiramo zahteve uporabnikov, zahteve in pogoji delovanja končnega sistema so manj izraženi.

Med metodologije, ki vsebujejo sistematične tehnike tudi za zbiranje zahtev sistema, kot jih predvideva faza razvoja sistemskih zahtev, uvrščamo metode, kot so CORE-Controlled Requirements Expression, JAD-Joint Application Development, Protocol Analysis, Scenario-based Requirement Engineering, SSADM-Structured System Analysis and Design Technology, SSM-Soft System Methodology, VORD-Viewpoint-Oriented Requirements Definition (Parviainen et al. 2003, str. 52).

Nekaj metod pri zbiranju zahtev sistema posebej izpostavlja zbiranje specifikacij in zahtev delovanja programske opreme. Gre za metode, s katerimi razvijamo popoln opis delovanja sistema, ki ga razvijamo. Lahko bi rekli, da gre za tipične predstavnike kaskadnih metod. Med tovrstne lahko uvrščamo metode, kot so Boochova metodologija, Jacobsonova metodologija, OMT-Object Modeling Technique, metodo UP-Unified Process, metodo RUP-Rational Unified process, Shlaer-Mellor Object-Oriented analysis Method, SCR-Software Cost Reduction requirements method, SADT-Structured Analysis and design Technique, SASS-Structured Analysis and System Specification, metoda UML-Unified Modeling Language, pristop win-win (Parviainen et al. 2003).

Nekatere metode predvidevajo izvajanje trajnih aktivnosti pri zajemanju zahtev sistema, kot so dokumentiranje zahtev, validacija, verifikacija in upravljanje sprememb zahtev. Takšnemu pristopu so prilagojene B-metoda, metoda Petrijevih mrež, VDM-Vienna Development model in uporaba jezika-Z.

Omenjeno je bilo, da je za agilne metode značilna manj poudarjena formalna stran zbiranja, dokumentiranja in upravljanja zahtev sistema. Značilnost teh metod je tesno povezovanje informatikov razvijalcev z uporabniki, naročniki oziroma udeleženci v sistemu. Tehnike in postopki zbiranja zahtev so usmerjeni na pogosto komunikacijo z naročniki, vzpostavljanje besednjaka iz domene, pisanje scenarijev izvedbe, pisanje zgodb, učenje od uporabnikov, ugotavljanje primerov dobre prakse in distribucijo prototipov delujočih delov programa. Med predstavnike teh metod se uvrščajo XP-ekstremno programiranje, metoda 'Scrum', metoda 'Crystal clear', ASD-Adaptive Software development, DSDM-Dynamic Systems Development Method, AUP-Agile Unified process, FDD-Feature Driven Development [URL: http://en.wikipedia.org/wiki/Agile_software_development].

Eno od vprašanj, na katerega želim podati odgovor v nalogi, se osredotoča na iskanje mehanizmov povezovanja tehnologov, razvijalcev, uprave in uporabnikov pri razvoju računalniških rešitev. Eden od v literaturi predlaganih mehanizmov je obvezna uporaba in poznavanje metodologije razvoja za vse udeležence v projektu. Vsi sodelujoči v razvoju bi morali poznati metode razvoja informacijskih sistemov (Gradišar, Resinovič, 2001, str. 422).

Ob podrobnejšem razmisleku ugotovimo, da je takšen pristop smiselno pričakovati v dobro organiziranih okoljih, katerih temeljno delovanje je usmerjeno v obsežne projekte razvoja programske opreme. V slabše organiziranih ali kadrovsko podhranjenih okoljih je poznavanje kompleksnih razvojnih metodologij težko pričakovati že od nekaj, kaj šele od vseh udeležencev razvoja. Manjši razvojni projekti, prehitre tehnološke spremembe, (pre)hitro povečevanje znanja, kadrovska podhranjenost ter drugi dejavniki lahko v izbranem okolju predstavljajo oviro za celovito izvajanje zahtevnih razvojnih metodologij. V takšnih pogojih je bolj smiselno projekte razvoja računalniške rešitve izvajati na prilagojen način, v obliki prilagojenih podprocesov, tudi s selektivnim izborom najprimernejših metod iz različnih metodologij.

Ob tem pa ne smemo spregledati dejstva, da je zbiranje zahtev uporabnikov ena od faz razvojnega procesa računalniške rešitve, ki je skupna vsem metodologijam in njihovim izpeljankam. To dejstvo je pametno izkoristiti kot priložnost, da prav v tej fazi vzpostavimo čim bolj večplasten in univerzalen komunikacijski mehanizem, ki bo razumljiv in uporaben za vse začetne (zbiranje zahtev) in poznejše (načrtovanje in razvoj) udeležence v razvoju računalniške rešitve.

#### 4.3.7.1 Zbiranje zahtev po metodologiji poenotenega razvoja (UP-Unified Process)

Organizirano zbiranje zahtev uporabnika je ena osnovnih veščin za načrtno izdelavo programske opreme. Razumevanje pomena te faze razvojnega procesa sodi med osnovna znanja različnih udeležencev, ki sodelujejo v razvoju programske opreme. Na osnovi podajanja zahtev naročnika oziroma uporabnika in v povratnem sporočanju razvijalca o razumevanju prejetih zahtev je sklenjen temeljni dogovor o vsebini in funkcionalnosti prihodnjega sistema. Ta faza je pomembna, ker se tu običajno zgodi pretvorba zahtev naročnika, izraženih iz ene oblike v drugo obliko, ki jo običajno izdelajo načrtovalci oziroma razvijalci končne rešitve. Po opravljeni analizi zahtev namreč naročnik slednje znova potrdi. Če jih naročniku lahko predstavimo že kot del načrta računalniške rešitve, lahko obravnavamo sporočilo o zahtevah med naročnikom in izvajalcem kot transparentno. Vzajemno enako razumevanje zahtev med naročnikom in izvajalcem lahko obravnavamo kot enega kritičnih dejavnikov končnega uspeha ali neuspeha projekta razvoja računalniške rešitve.

V nadaljevanju je predstavljenih nekaj temeljnih značilnosti postopkov in aktivnosti zbiranja zahtev uporabnika, kot jih izvorno predpisuje metodologija poenotenega razvoja programske opreme (Jacobsen, Booch, Rumbaugh,1999c).

Metodologija poenotenega razvoja sodi med iterativne in inkrementalne. Iz nje so bile izpeljane mnoge druge metodologije, med katerimi veljajo za bolj poznane naslednje ([URL: //en.wikipedia.org/wiki/Unified_process], 2008):

- BUP (angleško: Basic Unified Process) - lahka variacija osnovne metodologije, ki jo so jo razvili pri IBM,
- EUP (angleško: Enterprise Unified Process) - dodatek k metodologiji RUP,
- EssUP (angleško: Essential Unified Process) - lahka variacija, ki jo je razvil Ivar Jacobson,
- OpenUP (angleško: Open Unified Process) - metodologija razvoja programske opreme kot jo uporabljajo v odprtokodnem razvojnem projektu Eclipse,
- AUP (angleško: Agile Unified Process) - lahka variacija osnovne metodologije, ki jo je razvil Scott Ambler,
- RUP (angleško: Rational Unified Process) - razvojna metodologija, ki jo uporablja IBM,
- RUP-SE (angleško: Rational Unified Process-System Engineering) - verzija RUP, ki jo uporablja Rational Software za sistemski inženiring.

Zaradi svojega relativno velikega števila izpeljank in pogostega citiranja si izvorna metodologija UP zasluži malo večjo pozornost. Glede na ugotovitev, da je zbiranje zahtev skupni imenovalec praktično vseh metodologij razvoja programske opreme, sodi podrobnejše poznavanje aktivnosti iz domene tega metodološkega koncepta v temeljno znanje, s katerim bi moral biti seznanjen vsak udeleženec v razvoju programske opreme.

#### 4.3.7.2 Namen in pregled aktivnosti zbiranja zahtev

Uspešno prenašanje znanja med naročnikom in izvajalcem je odločilnega pomena za uspešen razvoj novih računalniških rešitev. Tu se srečujemo z oviro, da razvijalci programske opreme običajno niso hkrati tudi njeni uporabniki. Razvijalci običajno ne poznajo zahtev in omejitev sistema, v katerem deluje uporabnik, zato morajo najprej ugotoviti, kaj je uporabniku potrebno. To ugotavljanje sloni na zbiranju zahtev uporabnika**.** Vsak sistem naj bi zagotavljal neko

vrednost za uporabnika. Osnovni namen zbiranja zahtev uporabnika je usmerjanje razvoja programske podpore k pravemu sistemu**.** Zbrati moramo opis pogojev, okoliščin in funkcionalnosti, ki jih mora imeti sistem. Z njimi naj bi se strinjala tako naročnik oziroma uporabnik kot tudi razvijalci sistema. Poglavitni izziv zbiranja zahtev tiči v predpostavki vzajemnega branja in enotnega razumevanja rezultatov.

Vsak projekt izdelave programske opreme je unikaten. Zbiralec zahtev oziroma analitik bi moral uporabiti pristop zbiranja zahtev, ki predstavlja najmanjše tveganje. V večini situacij lahko aktivnosti predstavimo z naslednjimi koraki (Jacobsen, Booch, Rumbaugh, 1999c, str. 114-117):

- priprava seznama zahtev kandidatov: predstavlja spisek zbranih idej, ki so jih prispevali različni udeleženci v življenjskem ciklu razvoja in jih razumemo kot možne zahteve sistema ali jih ocenimo, da lahko predstavljajo del prihodnjega sistema;
- razumevanje konteksta, vsebine sistema: za ugotavljanje pravih zahtev in za izgradnjo pravih sistemov so ključni razvijalci, arhitekti in analitiki. Potrebujejo jasno razumevanje vsebine, v katero je vpet sistem. Obstajata vsaj dva pristopa, ki omogočata predstavitev vsebine sistema na način, ki je uporaben za razvijalce programske opreme. To sta modeliranje domene in poslovno modeliranje;
- zajem funkcijskih zahtev: uporabnik želi, da sistem zanj izvede neko funkcionalnost. Sistem dela za uporabnika primere uporabe. Vsak primer uporabe predstavlja en način uporabe sistema. Analitik bi moral ob ugotavljanju primerov uporabe opredeliti, kako bo izgledal vmesnik do uporabnika, ko bo primer uporabe izveden. Najboljši način je pripraviti osnutek vmesnika do uporabnika in skicirati več verzij, ki pokažejo informacijo po tistem, ko bo preoblikovana. O skicah je treba razpravljati z uporabnikom in narediti končno vizualizacijo oziroma, še bolje, prototip, ki ga uporabnik lahko preizkusi;
- zajem nefunkcijskih zahtev: nefunkcijske zahteve predstavljajo lastnosti sistema, kot so omejitve okolja in izvedbe, odvisnost od računalniškega okolja, vzdrževanje, razširljivost in zanesljivost. V sistemu lahko nastopijo tudi nefunkcionalne zahteve, ki so bolj splošne narave in jih ne moremo povezati z izbranim primerom uporabe ali z razredom iz realnega sveta. Takšne zahteve obravnavamo posebej, v seznamu dopolnilnih zahtev.

#### 4.3.7.3 Pregled pripomočkov v zbiranju zahtev

Učinkovito zbiranje zahtev terja od zbiralca oziroma analitika vrsto tehnik in lastnih kreacij v obliki dodatnih pripomočkov, ki mu omogočajo dober pregled in predstavitev sistema. Tradicionalno specifikacijo zahtev v obliki monolitnega dokumenta lahko nadomestimo z nizom drugih komunikacijskih pripomočkov. Glavni namen zajemanja zahtev uporabnika je razvoj funkcijskega modela sistema, ki ga gradimo. Uporaba primerov uporabe je primeren način, ki omogoča izdelavo takšnega modela. Funkcijske zahteve so spontano razvidne iz primerov uporabe.

Delovni proces zbiranja zahtev temelji na (Jacobsen, Booch, Rumbaugh, 1999c, str. 132):

- pripomočkih, ki jih uporabljamo v procesu zbiranja zahtev: izvajalci (akterji), primeri uporabe, opis arhitekture, slovar, prototip uporabniškega vmesnika, model primerov uporabe;
- delavcih, ki sodelujejo pri zbiranju zahtev: analitik sistema, specialist za primere uporabe, oblikovalec uporabniškega vmesnika,
- delovnem toku aktivnosti procesa zajemanja zahtev.

Med osnovne pripomočke za predstavitev zahtev sistema prištevamo (Jacobsen, Booch, Rumbaugh, 1999c, str. 133-140):

- Model primerov uporabe - omogoča razvijalcem in naročnikom vzajemno sodelovanje pri vzpostavljanju zahtev in pogojev, pod katerimi bo sistem deloval. Z modelom vzpostavljamo dogovor med naročnikom in izvajalcem o tem, kaj naj bi sistem delal.
- Izvajalec (akter) - vsak tip uporabnika sistema je predstavljen kot eden ali več izvajalcev oziroma akterjev. Vsak zunanji sistem, s katerim sodeluje naš sistem, je prav tako predstavljen kot eden ali več akterjev. Akterji pogosto sovpadajo z delavci ali s poslovnimi strankami. Akter igra eno vlogo v vsakem primeru uporabe, s katerim sodeluje.
- Primer uporabe (angleško: use case) - vsak način ali obliko, s katero akter uporablja sistem, predstavimo s primerom uporabe. Vsak primer uporabe predstavlja del funkcionalnosti, ki jo sistem zagotavlja in vrača akterju neko vrednost. S primerom uporabe opredelimo zaporedje aktivnosti, vključno z alternativami, ki jih sistem izvede med interakcijo akterja s sistemom. Na ta način je primer uporabe specifikacije podroben opis. Opisuje dinamiko dogajanja.
- Opis arhitekture (pogled na model primerov uporabe) - arhitekturni pogled na model primerov uporabe naj bi vseboval primere uporabe, ki opisujejo najpomembnejše in ključne funkcionalnosti ali tiste, ki morajo biti prednostno obravnavane v življenjskem ciklu razvoja računalniške rešitve.
- Slovar - izdelamo definicijo pomembnejših in splošnih izrazov, ki jih bodo uporabljali analitiki in ostali sodelujoči pri opisovanju sistema. Slovar je koristen za doseganje soglasja med razvijalci ob opredeljevanju, definiranju različnih konceptov in idej, z namenom po zmanjševanju tveganja napačnega razumevanja. Slovar lahko pogosto izpeljemo iz poslovnega modela ali modela domene. Ker je manj formalen (ne vsebuje razredov in neposrednih relacij), ga je lažje vzpostaviti in vzdrževati z zunanjimi sodelavci, kot so uporabniki in poslovne stranke.
- Prototip uporabniškega vmesnika - nam je v pomoč med zbiranjem zahtev za boljše razumevanje in specifikacijo povezav med človeškimi akterji in sistemom. Korist je dvojna: razvoj boljšega vmesnika, boljše razumevanje primera uporabe. Lahko pa uporabljamo tudi druge izdelke, kot so model uporabniškega vmesnika ali zaslonske skice.

Dinamika dogajanja v procesu zbiranja zahtev sloni na aktivnostih iskanja akterjev in primerov uporabe, izločanja prednostnih primerov uporabe, podrobno opredelitvijo primerov uporabe, izdelavi prototipov uporabniških vmesnikov in, končno, izdelavi modela primerov uporabe. Nekatere novejše analize so pokazale na velik praktičen in ontološki pomen modela primerov uporabe. Kot prednost je izpostavljena enostavnost uporabljene slovnice, ki ima pomembno vlogo v razumevanju problemske domene in zahtevah sistema ter omogoča lažjo komunikacijo med analitiki in uporabniki. Po drugi strani pa so prepoznane tudi nekatere slabosti kot je npr. ontološko nepopolna slovnica za področje zgradbe (dekompozicije) sistema in odsotnost gradnikov za predstavitev sistema na različnih ravneh podrobnosti (Irwin, Turk, 2005).

#### 4.3.7.4 Povzetek delovne faze zbiranja zahtev uporabnika

Zbiranje zahtev uporabnika lahko izvedemo s pomočjo pripomočkov, ki nam omogočajo predstaviti interakcijo uporabnika s sistemom in želeno funkcionalnost sistema. Osnovni pripomoček so primeri uporabe. Delovna faza zbiranja zahtev poteka kot proces različnih aktivnosti, ki jih izvajamo na način, s katerim postopoma dosežemo boljše razumevanje in

predstavitev sistema. Med značilne aktivnosti zbiranja zahtev štejemo (Jacobsen, Booch, Rumbaugh, 1999c, str. 171):

- Vzpostavitev poslovnega modela oziroma modela domene, pomoč za razumevanje vsebine sistema.
- Vzpostavitev modela primerov uporabe, pomoč za razumevanje funkcionalnih in nefunkcionalnih zahtev, specifičnih za predstavljene primere uporabe. Model je dodatno opisan s pomočjo niza diagramov in podrobnih opisov vsakega posameznega primera uporabe.
- Vzpostavitev niza skic ali prototipov uporabniških vmesnikov za vsakega akterja, pripomoček za načrtovanje uporabniških vmesnikov.
- Vzpostavitev opisa dodatnih zahtev, ki so splošni in neznačilni za posamezne primere uporabe.

Izdelani pripomočki so vhodna gradiva naslednjih delovnih faz razvoja: analize, načrtovanja, izvedbe in testiranja. Primeri uporabe nas vodijo skozi preostale delovne faze korak za korakom in ponovitev za ponovitvijo. V naslednji delovni fazi, fazi analize, uredimo primere uporabe kot objekte, da bi lahko bolje razumeli zahteve in jih pripravili za načrtovanje in končno izvedbo sistema. Za vsak primer uporabe v celotnem modelu vzpostavimo ustrezno realizacijo primera uporabe. Vzpostavimo tudi testne primere v fazi testiranja. Na ta način lahko tekoče povežemo niz delovnih faz za izvedbo projekta. Aktivnosti zbiranja zahtev lahko medsebojno povežemo. Tako lahko primere uporabe izdelamo tudi na osnovi procesnega modela [URL: http://www.cems.uwe.ac.uk/~sjgreen/MohammedOdehEtAlFiguresAndtext.pdf ]

#### 4.3.7.5 Alternativni pristopi k zbiranju in analizi zahtev sistema

Nekajkrat je že bilo omenjeno, da v analizi zahtev sistema nekatere metode bolj kot druge izpostavljajo formalizem zbiranja specifikacij zahtev za delovanje programske opreme. Takšen pristop izhaja iz spoznanj, da v načrtovanju in izvedbi sistemov povezujemo funkcionalnost iz domene zahtev uporabnika in funkcionalnost notranjega delovanja programske opreme, ki ima svoje specifične zahteve.

Praktične izkušnje razvijalcev programske opreme so privedle do ugotovitev, da se znotraj sistemov, ki jih razvijajo, nekatere skupine zahtevane funkcionalnosti računalniških rešitev znova in znova ponavljajo neodvisno od tega, komu je sistem namenjen (Withall, 2007). Ta ugotovitev je pripeljala do oblikovanja vzorcev zahtev, ki so ponujeni kot pripomoček analitikom iz poslovne domene, arhitektom in informatikom pri načrtovanju programske opreme ter preizkuševalcem delovanja slednje. Vzorci zahtev programske opreme sodijo med načrtovalske pripomočke s področja agilnih metod razvoja programske opreme.

Vzorci v splošnem predstavljajo šablone z vgrajenim in strukturiranim znanjem za reševanje različnih nalog. Vzorci zahtev programske opreme so namenjeni predstavitvi in reševanju ponavljajočih in podobnih tipov zahtev (Withall, 2007, str. 19), na katere pogosto naletimo pri ugotavljanju zahtev sistema. Vzorci zahtev se lahko pojavijo v izbranem kontekstu ali pa imajo splošni značaj za sistem. Lahko se navezujejo na predstavitev ali dostop do podatkov, na zmogljivost sistema ali na specifične funkcije, ki jih zahteva uporabnik.

V preglednici (tabela 3.) je podan opis vzorcev iz kataloga vzorcev zahtev (Withall, 2007, str. 49), združenih v skupine glede na njihovo domeno oziroma njihov namen oblikovanja in možne uporabe.

Tabela 3: Predstavitev namena posameznih vzorcev iz kataloga vzorcev zahtev

| Domena vzorcev (*) | Vzorec(*) | Namen |
|---|---|---|
| Temeljne zahteve (angleško: fundamental requirements) | Tehnologija (angleško: technology) | Vzorec za definiranje (ali izločanje) uporabe ali združevanja s specifično tehnologijo. |
| | Skladnost s standardom (angleško: comply-with-standard) | Specificiranje sistema, katerega delovanje se podreja zakonskim določilom ali izbranim standardom. |
| | Referenca na zunanjo zahtevo (angleško: refer-to-requirements) | Specificiranje zunanjih zahtev, ki morajo biti izpolnjene kot da so del notranjih zahtev sistema. |
| | Dokumentacija (angleško: documentation) | Specificiranje posebnih vrst dokumentacije, ki mora biti izdelana. |
| | Vmesnik med sistemi (angleško: inter-system-interface) | Definiranje vmesnikov do drugih (pod)sistemov ali zunanjih sistemov. |
| | Sodelovanje med sistemi (angleško: inter – system - interaction) | Specifikacija posebnih vrst vzajemnega delovanja med vmesniki podsistemov. |
| Zahteve povezane s poizvedbami podatkov (angleško: information requirements) | Podatkovni tip (angleško: data type) | Definiranje predstavitve elementarnih gradnikov nosilcev podatkov |
| | Podatkovna zgradba (angleško: data structure) | Definiranje predstavitve elementarnih gradnikov nosilcev podatkov. |
| | Identifikacija (angleško: identify) | Identifikacija podatkovnih entitet. |
| | Izračun (angleško: calculation formula) | Način preračunavanja vrednosti. |
| | Podatkovna trdoživost (angleško: data longevity) | Vzpostavljanje trajanja in ohranjanja informacije. |
| | Podatkovno arhiviraje (angleško: data archiving) | Hranjenje in kopiranje podatkov na izbrane lokacije. |
| Zahteve povezane s podatkovnimi entitetami (angleško: data entity requirements) | Trajanje entitete (angleško: living entity) | Definiranje tipov informacij z življenskim ciklom. |
| | Transakcija (angleško: transaction) | Definiranje tipov dogodkov za entitete z življenskim ciklom. |
| | Konfiguracija (angleško: configuration) | Določanje parametrov za nadzor obnašanja sistema. |
| | Poročilo (angleško: chronicle) | Določanje tipov dogodkov, ki jih beležimo v življenjskem ciklu. |
| | Skladiščenje (angleško: information storage) | Ohranjanje podatkov o entitetah v času ko sistem ne deluje. |

| Domena vzorcev (*) | Vzorec(*) | Namen |
|---|---|---|
| Zahteve povezane uporabniškimi funkcijami (angleško: user function requirements) | Poizvedba (angleško: inquiry) | Definiranje ekranskih prikazov za predstavitev informacij uporabniku. |
| | Poročilo (angleško: report) | Predstavljanje informacij v obliki poročil, ki nadomeščajo papirne izpise. |
| | Dostopnost (angleško: accessibility) | Zagotavljanje dostopnosti do sistema za uporabnike s specifičnimi potrebami ali zahtevami. |
| | Uporabniški vmesnik (angleško: user interface) | Vzpostavljanje infrastrukture komponent, s pomočjo katerih uporabnik vzajemno deluje s sistemom. |
| | Vrsta poročila (angleško: reporting infrastructure) | Vzpostavljanje različnih tipov poročil. |
| Zahteve povezane z zmogljivostjo sistema (angleško: performance requirements) | Odzivni čas (angleško: response time) | Specifikacija odzivnih časov na zahteve. |
| | Pretočnost (angleško: throughput) | Specifikacija deležev, ki jih mora sistem ali njegov vmesnik izvesti pri vhodno-izhodnem procesiranju. |
| | Zmogljivost procesiranja (angleško: dynamic capacity) | Opredelitev količine entitet, ki jih mora sistem hkrati procesirati |
| | Zmogljivost hranjenja (angleško: static capacity) | Opredelitev količine entitet, ki jih je sistem sposoben naenkrat trajno hraniti. |
| | Razpoložljivost (angleško: avalilability) | Opredelitev razpoložljivosti sistema. |
| Zahteve povezane s prilagoditvijo sistema (angleško: flexibility requirements) | Nadgradljivost (angleško: scalability) | Opredelitev možnosti 'neboleče' razširitve sistema, običajno zaradi povečanega obsega poslovanja. |
| | Razširljivost (angleško: extendability) | Opredelitev delov sistema z možnostjo dopolnitve z drugo programsko opremo. |
| | Prenosljivost (angleško: unparochialness) | Opredelitev splošne uporabnosti, ki ni omejena na en poslovni sistem. |
| | Sočasnost (angleško: multiness) | Opredelitev hkratnosti. |
| | Večjezičnost (angleško: multi-lingual) | Opredelitev možnosti sistema, da prikazuje svoj vmesnik v različnih jezikih. |
| | Namestitev (angleško: installability) | Opredelitev preprostosti namestitve ali nadgradnje sistema. |
| Zahteve povezane z nadzorom dostopanja (angleško: access control requirements) | Prijava uporabnika (angleško: user registration) | Opredelitev načina prijave novih uporabnikov. |
| | Overjanje uporabnika (angleško: user authentication) | Opredelitev načina overjanja (identifikacije) uporabnika pred uporabo sistema. |
| | Pooblaščanje (angleško: user authorization) | Opredelitev privilegijev (pravic) uporabnika za izvajanje posameznih funkcij sistema. |
| | Izjemno pooblaščanje (angleško: specific authorization) | Opredelitev uporabnikov in posebnih pooblastil (ali prepovedi) za izvajanje posameznih funkcij sistema (kdo in kaj). |

| Domena vzorcev (*) | Vzorec(*) | Namen |
|---|---|---|
| Zahteve povezane z nadzorom dostopanja (angleško: access control requirements) | Prilagodljivo pooblaščanje (angleško: configurable authorization) | Opredelitev nastavljive avtorizacije. Definicije funkcionalnosti sistema za posamezne uporabnike so nastavljive, dinamične. |
| | Odobritev (angleško: approval) | Vzpostavitev skupine ali posamičnih aktivnosti, ki morajo biti pred uporabo odobrene s strani druge osebe. |
| Zahteve povezane s poslovanjem (angleško: commercial requirements) | Organizacijska zgradba (angleško: multi-organization unit) | Specifikacija tipa organizacijske enote, ki je udeležena v delovanju sistema. |
| | Obračun storitve (angleško: fee/tax) | Opredelitev dajatev ali prispevkov, ki naj jih sistem zaračunava, pobira ali o njih poroča, na osnovi svojega izvajanja. |

(*) Zaradi referenčne prepoznavnosti so ohranjena imena vzorcev tudi v njihovi izvorni obliki.

Vir: Withall, 2007

Uporaba vzorcev zahtev je lahko zelo uporaben vodič tako za programerje začetnike kot za razvijalce prototipov končnih rešitev. Vzorci zahtev so uporabni tudi za tiste razvijalce, ki prisegajo na agilne metode razvoja. Zanesljivo pa je uporaba vzorcev zahtev odličen pripomoček za vse, ki poznajo tudi načrtovalske vzorce ter njihov pomen v načrtovanju računalniških rešitev.

Glede na svoje poreklo nastanka so vzorci zahtev bolj razumljivi tistim razvijalcem, ki so specializirani za programiranje. Lahko rečemo, da bo uporaba vzorcev zahtev s stališča izvajanja procesa zbiranja zahtev verjetno bolje sprejeta v okolju, ki uporablja metodologije agilnega razvoja kot v okolju, ki uporablja tradicionalne metodologije razvojnega cikla. Vsekakor pa vzorcev zahtev tudi tam ne bi smeli spregledati.

# 4.3 INFORMACIJSKA ARHITEKTURA

Informacijska arhitektura nima natančne definicije in predstavlja širok pojem, pod katerim razumemo dejavnosti na področju organizacije ter predstavitve koncepta podatkov. Področje informacijske arhitekture se ukvarja z modeliranjem distribucije podatkov na različnih področjih, ki se ukvarjajo z njihovo predstavitvijo od knjižnic, spletnih strani, razvoja podatkovnih baz, razvoja programske opreme in drugih oblik snovanja porazdeljenih informacijskih virov (McNay, 2003, str. 104). Izraz informacijska arhitektura najpogosteje povezujemo z zasnovo in distribucijo podatkov (informacij) na spletnih straneh, vendar ga lahko uporabljamo v kontekstu poljubne informacijske zasnove ali računalniškega sistema (Barker, 2005). Nekateri avtorji opredeljujejo informacijsko arhitekturo kot eno od komponent širšega koncepta arhitekture informacijskih sistemov, ki vključuje tako arhitekturo strojne opreme in komunikacij kot model izvedbe poslovanja (Teng, Kettinger, 1995, str. 33).

Ob upoštevanju definicije informacije, ki pravi, da so informacije rezultat procesa interpretacije podatkov (Vidmar, 2002, str. 25), bi lahko rekli, da se informacijska arhitektura ukvarja z organizacijo porazdelitve in predstavitve podatkov v razpršenem računalniškem okolju. Razpršeno okolje pri tem predstavlja različne fizične lokacije podatkovnih virov. Skladno s to razlago lahko pojem dopolnimo in rečemo, da se informacijska arhitektura ukvarja z logičnimi povezavami informacijskih virov ter zasnovo računalniških programov, ki za končnega uporabnika izvajajo informacijske storitve s pomočjo podatkov in programskih komponent, razpršenih med decentralizirano topologijo sistemskih virov.

Značilnost sodobnih informacijsko komunikacijskih sistemov je lokacijska razpršenost strojne opreme, podatkov in kontrolnih funkcij, kar daje tem sistemom značilno strukturo oziroma topologijo. Vzajemno delovanje sistemskih virov v razpršenem sistemu temelji na izvajanju različnih storitev, ki so porazdeljene med specializirane podsisteme (plasti storitev). Delovanje informacijsko komunikacijskega sistema običajno predstavljamo s pomočjo arhitekture plasti storitev. V tej arhitekturi informacijske storitve uvrščamo v plast informacijskega sistema, ki je namenjena obdelavi ter predstavitvi podatkov (Vidmar, 2002, str. 91). Ta plast v arhitekturi sistema povezuje interakcijo uporabnika in komunicira z drugimi plastmi sistema (Vidmar, 2002, str. 436). Plast transportnega sistema pretvarja podatke v sporočila (in obratno) ter usmerja sporočila med računalniki, medtem ko plast prenosnega sistema zagotavlja fizično povezanost med računalniki (Vidmar, 2002, str. 90-92).

Osnovni namen informacijskih sistemov je zagotavljanje informacijskih storitev za končnega uporabnika. S čedalje večjo omrežno povezanostjo računalnikov tudi informacijski sistemi postajajo čedalje bolj porazdeljeni. Izvedba informacijskih storitev v porazdeljenih informacijskih sistemih sloni na procesiranju aplikacijskih objektov, med katere prištevamo uporabniške datoteke, semantično opredeljene bloke podatkov in programske module (Vidmar, 2002, str. 436). Celoten nabor informacijskih storitev lahko razdelimo tudi po vsebini s stališča arhitekture delovanja informacijsko komunikacijskega sistema. Zunaj njega izdelujemo naše računalniške rešitve, ki za sistem predstavljajo nestandardne storitve. Znotraj sistema pa se poslužujemo standardnih uporabniških storitev v aplikacijski plasti ter podpornih storitev, ki jih zagotavljata predstavitvena plast in plast seje (Vidmar, 2002, str. 437).

Če želimo, da naše računalniške rešitve izrabljajo vire v porazdeljenem sistemu, se morajo slednje posluževati uporabe standardnih in podpornih storitev takšnega sistema. Čedalje večja razpršenost sistemskih virov pa povzroča tudi večje probleme pri zasnovi računalniških rešitev in zagotavljanju homogenega uporabniškega okolja.

S stališča arhitekture računalniških rešitev, kjer se rešujejo ti problemi, je treba predvideti njihovo obravnavo v okviru aplikacijske, predstavitvene in sejne plasti po referenčnem modelu odprtih sistemov (Vidmar, 2002, str. 220-223), s katerim predstavljamo delovanje informacijsko komunikacijskih sistemov.

Z vidika zgornjih navedb bomo v nadaljevanju informacijsko arhitekturo obravnavali kot koncept organizacije podatkov ter delovanja programskih komponent, s katerimi zagotavljamo informacijske storitve v porazdeljenem informacijsko komunikacijskem okolju. Pri konkretni zasnovi in arhitekturnem opisu končne računalniške rešitve pa bomo upoštevali, da se informacijska arhitektura navezuje tako na obliko hranjenja podatkov (centralna, porazdeljena baza) kot tudi na medsebojno povezanost in način vzajemnega delovanja naših (ali drugih) programskih komponent, ter na poznavanje različnih referenčnih modelov delovanja sistemov in poznavanje nekaterih povezanih tehnoloških specifikacij.

### 4.3.1 Pomen referenčnih modelov za izvedbo arhitekture sistemov

Referenčni modeli predstavljajo splošen abstraktni okvir za razumevanje odnosov med entitetami v nekem okolju oziroma sistemu. Modelno predstavitev neke arhitekture predstavimo z minimalnim naborom konceptov, načel in odnosov znotraj izbrane problemske domene. Referenčni modeli so neodvisni od specifičnih standardov tehnologij, končne izvedbe ali drugih izvedbenih podrobnosti. Običajno služijo referenčni modeli kot abstraktni okvir za konkretno arhitekturo nekega sistema, v kateri uporabimo bodisi konkretne izvedbene rešitve bodisi slednje podrobno specificiramo. Za usmerjanje načrtovanja nekega sistema lahko na abstraktnem nivoju obstaja več kot en sam referenčni model [URL: http://www.oasis-open.org/committees/download.php/19679/soa-rmccs.pdf] .

Za področje povezovanja naprav v telekomunikacijskem sistemu poznamo referenčni model OSI (angleško: open systems interconnection). Ta referenčni model je mednarodna agencija za standardizacijo (ISO) utemeljila kot standard, zato ga pogosto navajamo tudi kot ISO/OSI referenčni model. Predstavlja osnovni arhitekturni model za izvedbo telekomunikacijskih storitev. Sem sodi tudi komunikacija med računalniki (napravami) v mreži. Določa, kako se informacija iz aplikacije na eni napravi preko omrežja prenese na aplikacijo na drugi napravi. Model sloni na izvajanju storitev, ki se izvajajo v sedmih plasteh, od katerih vsaka skrbi za svojo funkcionalnost, med katere sodijo (Vidmar, 2002, str. 143-44):

1. fizična plast skrbi za prenos bitov preko prenosnega medija,
2. povezavna plast prenaša podatkovne okvire med dvema točkama, ki ju povezuje prenosni medij,
3. omrežna plast skrbi za usmerjanje paketov skozi vozlišča omrežja,
4. transportna plast skrbi za storitve, ki izvajajo transport podatkov med dvema končnima računalniškima aplikacijama,
5. plast seje se uporablja za storitve, ki logično povezujejo oddaljene procese med seboj,
6. predstavitvena plast skrbi za pretvorbo podatkov iz ene v drugo obliko,
7. aplikacijska plast služi kot vmesnik med aplikacijami zunaj OSI modela in OSI modelom. Tu nastaja določena podatkovna struktura, ki jo oddajnik predaja oddaljeni aplikaciji.

Za vzajemno delovanje informacijsko komunikacijskih sistemov v svetovnem spletu se čedalje bolj uveljavlja referenčni model TCP/IP. Sestavljen je iz štirih plasti, ki pokrivajo funkcionalnost modela OSI (Vidmar 2002, str. 149).

Poznani pa so še mnogi drugi referenčni modeli, katerih poznavanje nam lahko koristi pri vzpostavljanju informacijske arhitekture. Za področje medsebojno povezljivih programskih rešitev pri vzpostavljanju in distribuciji prostorskih podatkov je na voljo OGC (angleško: Open Geospatial Consortium) referenčni model. Slednji predstavlja nabor specifikacij za vzajemno delovanje in povezovanje aplikacij ter storitev, vezanih na uporabo geokodiranih podatkov [URL: http: //portal.opengeospatial.org/files/?artifact_id=3836].

Na področju organizacije izvajanja poslovnih procesov in povezanih storitev, ki jih lahko opravljajo različni izvajalci v razpršenem računalniškem okolju, lahko uporabimo referenčni model storitvene arhitekture (angleško: Service Oriented Architecture, SOA) [URL: http: //www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf]. Na področju razvoja porazdeljenega delovanja aplikacij ali sistemov, ki je neodvisno od platforme, lahko vzamemo kot referenčni model Model Driven Architecture (OMG MDA). Slednji zasnovo sistema utemeljuje na osnovi delov in konektorjev ter s pravili vzajemnega povezovanja pri uporabi konektorjev [URL: http://www.omg.org/docs/omg/03-06-01.pdf], [URL: http://www.cwmforum.org/Model-Driven+Architecture.pdf ].

Referenčni modeli so lahko definirani na različnih nivojih abstrakcije. Zelo abstrakten model bo morda predstavljal zgolj različne kose opreme s specifično funkcionalnostjo v računalniški mreži. Podroben referenčni model pa bo lahko predstavljal vzajemno delovanje med zelo specifičnimi funkcijami (metodami) nekega računalniškega programa.

Pri vzpostavljanju konkretne arhitekture računalniške rešitve nam je poznavanje referenčnih modelov lahko v veliko pomoč. Z njihovo pomočjo in ob pomoči poznavanja različnih, že uporabljenih arhitekturnih vzorcev načrtovanja računalniških rešitev lahko pričakujemo lažje snovanje konkretnih rešitev. Uporaba in poznavanje referenčnih modelov vsekakor sodita med pomembne pripomočke oziroma veščine udeležencev v razvoju in izdelavi računalniških rešitev.

### 4.3.2 Arhitektura programskih rešitev (angleško: software architecture)

Arhitektura programskih rešitev je ožji pojem informacijske arhitekture, ki se osredotoča na zgradbo sistemov, ki vsebujejo komponente programske opreme, navzven vidne lastnosti teh komponent in relacije med njimi. Pojem se nanaša tudi na dokumentiranje, ki omogoča lažjo komunikacijo udeležencev v razvoju programske opreme ter na različne vzorce oblikovanja programskih modulov in načine njihovega povezovanja. Področje arhitekture programskih rešitev se osredotoča na zmanjševanje kompleksnosti s pomočjo posploševanja in ločevanja pomenov (Mei, 2000).

Ena od značilnosti razvoja programske opreme in orodij je postopna rast nivoja abstrakcije, kar vpliva na vpeljavo novih konceptov med načrtovalske gradnike programske opreme. To vpliva tudi na arhitekturo programskih rešitev. Uvedba abstraktnih podatkovnih tipov je omogočila realizacijo podatkov skupaj z operacijami in preseganje nivoja uporabe posameznih stavkov programskega jezika ter ločenih algoritmov. Abstraktni podatkovni tipi so prinesli možnosti, da posamezne dele sistema razvijamo s pomočjo besednjaka podatkovnih tipov, namesto s pomočjo konstruktov tedanjih programskih jezikov (Garlan, Shaw, 1994, str. 4).

Zgradba sistema postane na ta način zbirka računalniških komponent in opisov njihovih povezav. S konfiguracijo sistema lahko opišemo topologijo povezav med komponentami in njihovimi konektorji.

Arhitekturo oziroma medsebojno povezanost komponent lahko izvajamo na osnovi različnih modelov. Posledično se to odrazi na arhitekturnem slogu ali končnem arhitekturnem vzorcu. Med splošne arhitekturne modele in vzorce štejemo (Garlan, Shaw, 1994, str. 7-16):

- zasnova cevi in filtrov (angleško: pipes and filters),
- objektna zasnova sistema (angleško: object-oriented organization),
- zasnova implicitnih pozivov (angleško: implicit invocation),
- zasnova hierarhičnih plasti (angleško: layered systems),
- zasnova skladišč (angleško: repositories),
- zasnova interpreterja (angleško: table driven interpreters),
- porazdeljeni procesi (angleško: distributed processes),
- zasnova arhitekture na domeni (angleško: domain specific),
- zasnova na osnovi nadzora procesov (angleško: process control systems),
- mešana zasnova zgoraj navedenih vzorcev.

Predstavitev arhitekture končne programske rešitve lahko sloni na predstavitvi pogledov različnih opazovalcev, ki opazujejo model rešitve, ta pa vsebuje komponente, vezne člene in druge elemente. Za takšen način predstavitve arhitekture programskih rešitev so izdana tudi priporočila v obliki standarda (ANSI/IEEE 1471-2000). Slednji vsebuje priporočila za izdelavo arhitekturnega opisa, s katerim bo arhitektura sistema ustrezno dokumentirana. Priporočilo utemeljuje pomen udeležencev (angleško: stakeholders) v izgradnji oziroma uporabi sistema in njihov pogled oziroma interes (angleško: concern) nad sistemom. Skladno z navedenim sloni arhitekturni opis na različnih gledišči (angleško: viewpoints) in pogledih (angleško: views) udeležencev. Na osnovi izbranega gledišča opredelimo za gledišče tipične elemente in relacije, ki jih konkretno predstavimo s predstavitvijo posameznega pogleda. Vsak pogled je lahko sestavljen iz enega ali več arhitekturnih modelov. Arhitekturni model razvijemo s pomočjo metod, ki smo jih vzpostavili v izbranem gledišču. Nekaj možnih gledišč na sistem, kot jih navaja priporočilo (ANSI/IEEE 1471-2000), je naslednjih:

- Vidik sestavnih delov (angleško: structural viewpoint). Na osnovi tega gledišča predstavimo strojne izvedbene komponente sistema in njihovo organizacijo ter programske komponente, ki tvorijo sistem. Predstavimo vmesnike in način medsebojnega povezovanja.
- Vidik obnašanja sistema (angleško: behavioral viewpoint). Z njim opredelimo dinamične aktivnosti sistema. Prikažemo, kakšne vrste aktivnosti povzroča sistem in v katerih aktivnostih sodeluje. Predstavimo zaporedje in sinhroniziranost medsebojno povezanih aktivnosti. S tega vidika predstavimo tudi posledice akcije uporabnika in povratne storitve.
- Vidik fizične povezanosti (angleško: physical interconnect viewpoint). Z njim predstavimo razslojitev na sistemske komponente in njihove vzajemne fizične povezave.
- Vidik pričakovanih napak v izmenjavi sporočil (angleško: link bit error rate viewpoint). Z njim predstavimo zmogljivosti uporabljenega komunikacijskega sistema.

Standard IEEE: 1471-2000 je bil prevzet tudi kot ISO standard in je objavljen kot standard ISO/IEC 42010: 2007. Na osnovi standarda je vzpostavljena konvencija o arhitekturnih opisih, se pravi o pripomočkih, s katerimi lahko opišemo sistem z različnih vidikov. Osnovni namen različnih pogledov je predstavitev konteksta (vsebine) sistema različnim udeležencem, ki so

kakor koli povezani z njim, bodisi ga izvajajo, gradijo, načrtujejo, vzdržujejo ali prenavljajo. Upoštevanje standarda ne zahteva posebnega procesa. Opis arhitekture programske rešitve je skladen s standardom, če se nanaša na (EEE Std 1471 Frequently Asked Questions (FAQ) [URL: http: //www.iso-architecture.org/ieee-1471/ieee-1471-faq.html]:

- prepoznavanje udeležencev v sistemu in njihovih vlog,
- vzpostavitev in definicijo pogledov na sistem s stališča teh vlog,
- dokumentiranje opredeljenih pogledov na zgradbo sistema,
- beleženje neskladja med pogledi,
- zagotovitev osnovnih principov izvedbe za izgradnjo arhitekturnega opisa.

Različni pogledi na arhitekturo sistema postavljajo slednjo v osrednji komunikacijski kanal za različne udeležence v procesu razumevanja delovanja in razvoja podpornih programskih rešitev. Arhitektura sistema omogoča tudi vpogled v distribucijo zunanjih virov podatkov, ki so potrebni za delovanje sistema, prav tako omogoča vzdrževanje ali ponovno uporabo komponent sistema.

### 4.3.3 Arhitekturni vzorci načrtovanja računalniških rešitev (angleško: architectural patterns)

Arhitekturni vzorci predstavljajo koncepte že uporabljenih in uspešnih zasnov arhitekture računalniških rešitev. S stališča predstavitve so arhitekturni vzorci sorodni konceptu gledišč, kot ga pozna priporočilo ANSI/IEEE 1471-2000. Na osnovi koncepta arhitekturnih vzorcev prav tako definiramo poglede na elemente in njihova razmerja v izvedbeni zasnovi prihodnje računalniške rešitve. S pomočjo arhitekturnih vzorcev predstavimo vzajemno delovanje gradnikov, ki si delijo odgovornost za izvedbo končnih računalniških rešitev (Avgeriou, Zdun, 2005, str. 4.). Arhitekturni vzorci predstavljajo temeljni pregled nad organizacijo računalniške rešitve in prikazujejo nabor predvidenih podsistemov, njihovo povezanost in odgovornost. Vsebujejo tudi smernice za organiziranje relacij med njimi (Schmidt, Buschmann, 2003, str. 698).

Uporaba preizkušenega arhitekturnega vzorca lahko znatno olajša načrtovanje in razvoj računalniških rešitev, ki slonijo na komponentah. Omogoča prevzemanje preizkušenih modelov zasnove delovanja sistema, ponovno uporabo načrtov in preizkušenih komponent.

Poznamo precej splošno uveljavljenih in preizkušenih vzorcev snovanja arhitekture (zgradbe) računalniških rešitev. Z vidika različnih perspektiv (gledišč) lahko arhitekturne vzorce uvrstimo v več skupin (Avgeriou, Zdun, 2005, str. 6-7):

- pogled v obliki slojev (angleško: layered view). Značilnost te skupine je podrobna členitev sistema na posamezne vertikalne komponente, ki medsebojno sodelujejo s pomočjo vmesnikov. Slednji vsebujejo ustrezna stanja in protokole sodelovanja. Običajno vsebujejo tudi nadzorni mehanizem, ki usklajuje delovanje komponent. V to skupino uvrščamo:
  - vzorec slojev (angleško: layers pattern),

- vzorec posrednega sloja (angleško: indirection layer pattern);

- pogled v obliki pretoka podatkov (angleško: dataflow view). Za to skupino so značilne pretvorbe toka vhodnih podatkov. Komponente sistema izvajajo pretvorbo podatkov na osnovi povezovalnih elementov, ki so medsebojno neodvisni. Sem sodijo:
  - paketno sekvenčni vzorec (angleško: batch sequential pattern),
  - vzorec filtrov in cevi (angleško: pipes and filters pattern);
- pogled na centralno zbrane podatke (angleško: data-centered view). Značilnost tega pogleda na sistem je centralno skladišče podatkov, do katerega dostopajo in ga uporabljajo številne komponente sistema. Skladišče je od komponent neodvisno, prav tako so komponente običajno neodvisne druga od druge. Sistem ima lahko tudi več skladišč podatkov. Elementi, ki skrbijo za transfer zapisovanja in branja podatkov, so vmesniki, vezani tako na podatkovno skladišče kot na komponente dostopanja. V to skupino uvrščamo:
  - vzorec pasivnega centralnega skladišča (angleško: shared repository pattern),
  - vzorec aktivnega centralnega skladišča (angleško: active repository pattern),
  - vzorec šolske table (angleško: blackboard pattern);
- prilagoditveni pogled na sistem (angleško: adaptation view). V tem pogledu obravnavamo sistem v dveh delih, in sicer v obliki jedra sistema, ki je nespremenljivo, in v obliki prilagodljivega dela. Ta se spreminja skozi čas oziroma skozi nove verzije sistema. Oba dela medsebojno komunicirata skozi spojnike z dobro definiranimi vmesniki. V to kategorijo umeščamo naslednje vzorce:
  - vzorec mikrojeder (angleško: microkernel pattern),
  - vzorec odzivanja (angleško: reflection pattern),
  - vzorec prestreznikov (angleško: interceptor pattern);
- pogled na sistem z možnostjo razširitve v drugem jeziku (angleško: language extension view). V tem pogledu sistem obravnavamo kot del, ki je izvoren v strojno programskem okolju in kot del, ki lahko vsebuje dodatne (neizvorne), v drugem jeziku izdelane komponente. Ključni problemi tega pogleda se osredotočajo na vgradnjo in uspešno sobivanje dodatnih komponent z izvornimi. Izvorni in dodatni deli sistema medsebojno komunicirajo s pomočjo posredovalne (prevajalne) komponente. Sem uvrščamo:
  - vzorec tolmačenja (angleško: interpreter pattern),
  - vzorec navideznega stroja (angleško: virtual machine pattern),
  - vzorec, temelječ na pravilih (angleško: rule based pattern);
- pogled z vidika uporabniškega posredovanja (angleško: user interaction view). Za ta pogled na sistem je značilna delitev na del z uporabniškim vmesnikom in na del, ki vsebuje logiko izvedbe aplikacije v povezavi s tem vmesnikom. Elementi, ki predstavljajo podatke uporabniku, vsebujejo logiko izvedbe aplikacije in dostopajo do podatkov, so izvedeni v obliki komponent, ki medsebojno delujejo preko vmesnikov. Ti si izmenjujejo podatke in sporočila. V to skupino uvrščamo naslednje vzorce:
  - vzorec ločevanja predstavitve in dostopanja do podatkov s pomočjo kontrolerja (angleško: model-view-controller pattern),
  - vzorec hierarhičnega delovanja agentov (angleško: presentation-abstraction-control pattern);

- pogled z vidika vzajemnega delovanja komponent (angleško: component interaction view). V tem pogledu vidimo sistem kot množico neodvisnih komponent, ki vzajemno delujejo. Komponente so medsebojno neodvisne, saj zgolj izmenjujejo podatke, medsebojno pa se ne nadzirajo. Vzajemno delovanje komponent je lahko sinhrono ali asinhrono in lahko sloni na osnovi neposrednih klicev ali na osnovi izmenjave sporočil. Sem sodijo:
  - vzorec eksplicitnega poziva (angleško: explicit invocation pattern),
  - vzorec implicitnega poziva (angleško: implicit invocation pattern),
  - vzorec odjemalec strežnik (angleško: client server pattern),
  - vzorec povezave vsak z vsakim (angleško: peer to peer pattern),
  - vzorec objave in naročanja (angleško: publish subscribe pattern);
- pogled z vidika porazdelitve komponent (angleško: distribution view). V tem pogledu obravnavamo sistem kot niz v računalniškem okolju porazdeljenih komponent. Komponente so fizično razpršene med vozlišči mreže ali med njihove procese. Vzajemno delujejo na osnovi vmesnikov, ki si izmenjujejo podatke ali pozive. Sem uvrščamo:
  - vzorec posrednikov (angleško: broker pattern),
  - vzorec klica oddaljene procedure (angleško: remote procedure call pattern),
  - vzorec skladiščenja pozivov (angleško: message queuing).

Pri snovanju računalniških rešitev, kjer načrtujemo izvedbo aplikacije na osnovi porazdeljenih komponent, so nam lahko v veliko pomoč že vzpostavljeni arhitekturni vzorci. Zasnova arhitekture sodi med pomembnejše aktivnosti v procesu razvoja programskih rešitev. Predstavlja temeljno organizacijo komponent sistema in njihovo vzajemno delovanje (Wang, Fung, 2004). Ne glede na variacije razvojnega procesa, si z arhitekturo programske rešitve vzpostavimo ogrodje končne rešitve. Tudi poznavanje različnih arhitekturnih vzorcev računalniških rešitev lahko uvrstimo med pomembnejše pripomočke za udeležence v razvoju in izdelavi računalniških rešitev.

### 4.3.4 Protokoli in tehnološke specifikacije za porazdeljeno izvedbo programskih rešitev

Ugotovili smo že, da razpršenost virov, na osnovi katerih sistem deluje, uvrščamo med osnovne značilnosti informacijsko komunikacijskih sistemov. Kadar končni uporabnik ali računalniška rešitev dostopata do vsebine datoteke na oddaljenem računalniku, morata uporabljeni oddaljeni vir uporabljati enako kot, da je slednji na lokalnem računalniku. Ta lastnost sistema se imenuje transparentnost dostopa (Vidmar 2002, str. 416). V porazdeljenem načinu delovanja mora informacijsko komunikacijski sistem zagotavljati še vrsto drugih transparentnosti, kot so transparentnost lokacije virov, sočasnosti dostopanja do virov, deljivosti virov, zmogljivosti in skalabilnosti sistema.

Če si predstavljamo informacijsko komunikacijski sistem kot sistem odjemalcev in strežnikov (različnih računalnikov), poteka izvedba računalniških rešitev v takšnem sistemu na osnovi delovanja objektov, ki so porazdeljeni med njimi. Običajno klic odjemalca nima neposredne reference do objekta na strežniku (Vidmar 2002, str. 421). Za računalniško rešitev je treba

zagotoviti transparentnost dostopa med porazdeljenimi objekti. Transparentnost sistema je izvedena s pomočjo posrednikov (zastopnikov) objektov strežnika, ki objekt strežnika in vse njegove metode zastopajo na strani odjemalca. Računalniške rešitve odjemalca jih kličejo (uporabljajo) tako kot druge metode lokalnih objektov. Posamezen objekt strežnika ima lahko več posrednikov, po enega za vsak oddaljeni računalnik. Ob klicu metod posrednikov se vrednosti posredovanih parametrov pretvorijo v ustrezno obliko in se po transportnem sistemu posredujejo odrezku, ki na strani strežnika predstavlja vse oddaljene odjemalce. Odrezek na strani strežnika izlušči prejete parametre in pokliče ustrezno metodo objekta na strežniku. Na ta način odrezek in posrednik zagotavljata transparentnost sistema (Vidmar 2002, str. 421-423).

Klicanje in izvedba procedur (metod) na oddaljenih sistemskih virih sloni na izmenjavi in prenašanju sporočil med oddaljenimi objekti. Sam prenos sporočil poteka po transportnem sloju, komunikacija (semantična izmenjava) sporočil pa se odvija v aplikacijskem sloju po OSI referenčnem modelu. Tehnološko gledano izvedba oddaljenih procedur temelji na izvajanju protokolov, ki omogočajo izmenjavo sporočil v omrežju. Vzorec komunikacije lahko poteka na različne načine. Izvedeni so s pomočjo protokolov. Protokoli, ki omogočajo izvedbo tehnologije klicanja oddaljenih procedur, so:

- RPC (angleško: Remote Procedure Call) protokol. Pod tem imenom srečujemo tako izvedbo tehnologije oddaljenega klica kot ime protokola. Protokol (dokumentiran v RFC 1831) omogoča podprogramu ali rutini programske rešitve prenos sporočila (zahteve, odgovora) med različnimi lokacijami, ne da bi bilo potrebno eksplicitno programirati podrobnosti te oddaljene interakcije. Omogoča tako podajanje zahteve kot prejemanje odgovora. Izvaja se (prenaša podatke) s pomočjo UDP (angleško: user datagram protocol) protokola ali zanesljivejšega TCP (angleško: transmission control protocol) protokola in omogoča avtentikacijo klicatelja. Izmenjava sporočil običajno poteka v obliki binarnih dokumentov.
- SOAP (angleško: Simple Object Access Protocol) - je W3C/IETF standard iz skupine internetnih protokolov Predstavlja komunikacijski protokol za izmenjavo XML (strukturiranih) sporočil, ki se običajno izvaja s pomočjo HTTP/HTTPS protokolov (Vidmar 2002, str. 424). To dejstvo omogoča izrabo aplikacijskega sloja tudi za komunikacijo, kar posledično pomeni lažje (nefiltrirano) prehajanje skozi požarne pregrade. Uporablja se za komunikacijo med aplikacijami s pomočjo interneta. Protokol je neodvisen od uporabljene strojne opreme, operacijskega sistema ali uporabljenega jezika v izvedbi programske rešitve.

Funkcionalnost klicanja oddaljenih metod mora podpirati vrsto zahtev, kot so varnost, sinhronizacija in upravljanje pretoka podatkov. Izvedba funkcionalnosti klicanja oddaljenih metod je vgrajena v različne standardizirane rešitve, bodisi kot dodatek operacijskemu sistemu bodisi kot poseben samostojen dodatek, ali pa je na voljo v obliki funkcionalnosti, vgrajene v programski jezik. Povezovanje in izmenjava podatkov med aplikacijami ali komponentami aplikacije se pogosto izvaja s pomočjo vmesne programske opreme (angleško: middleware). Med najpogosteje uporabljene porazdeljene objektne modele za oddaljeni dostop do podatkov uvrščamo (Vidmar 2002, str. 423):

- lastno izvedbo klicev oddaljenih metod v okviru uporabe programskega jezika Java lahko izvedemo s pomočjo lokalnega objektnega modela RMI (angleško: remote metod invocation). Izvedba je integrirana v uporabo programskega jezika Java, kar je hkrati prednost in slabost, saj je ne moremo uporabiti v drugih programskih jezikih;
- izvedba klicev oddaljenih metod v okviru uporabe Microsoftovih operacijskih sistemov temelji na uporabi DCOM (angleško: distributed component object model). Predstavlja

Microsoftovo zaščiteno tehnologijo za komunikacijo med komponentami programske opreme, ki je porazdeljena v računalniški mreži. Tehnologija DCOM je nadgradnja starejše Microsoftove tehnologije COM (angleško: component object model). Ta tehnologija je bila sprva načrtovana za vzajemno sodelovanje med objekti (komponentami), ki bi jih lahko uporabili za različne računalniške rešitve tudi izven istega vira izvajanja aplikacije (Eddon, 1998 str. 11-13). Dopolnitev infrastrukture COM tehnologije z Microsoftovo tehnologijo OLE (angleško: object linking and embedding) se je odrazila v posodobljenih programskih komponentah, imenovanih Active-X. V osnovi je Active-X koncept Microsoftove tehnologije, ki omogoča združevanje objektnih komponent po specifikaciji vmesnika COM (Microsoft, 1997, str. 191);

- CORBA (angleško: common object request broker architecture) ali arhitektura posrednikov zahtev skupnih objektov: Predstavlja standard OMG (angleško: object management group) za vzajemno delovanje objektov (komponent) programske opreme, ki so lahko napisane v različnih programskih jezikih ali se izvajajo na različnih računalniških platformah v mreži. Za opis vmesnika posamezne komponente se uporablja jezik za definicijo vmesnika (angleško: interface definition language), s katerim opredelimo vmesnik, s katerim se objekt predstavlja okolici. Arhitektura posrednikov je zasnovana za vzajemno delovanje porazdeljenih komponent, neodvisno od uporabljenega operacijskega sistema (Vinoski, 1997).

Različne ovire pri izvedbi porazdeljenih aplikacij s pomočjo klicanja oddaljenih procedur na osnovi protokolov, ki prenašajo podatke v binarni obliki, čedalje bolj favorizirajo protokole (SOAP) z lažjim prehodom skozi požarne pregrade interneta. Tudi uporaba tega protokola se odraža v različnih ponujenih tehnoloških rešitvah:

- Spletna povezovalna tehnologija (angleško: Web services) je po definiciji W3C tehnologija, namenjena podpori vzajemnega delovanja aplikacij v računalniški mreži. Tehnologija spletnih storitev je namenjena komunikaciji med odjemalci in strežniki v mreži, ki komunicirajo s pomočjo XML sporočil v skladu s SOAP standardi. Uporaba tehnologije spletnih storitev se navezuje tudi na uporabo opisnega jezika spletnih storitev (WSDL - Web Services Description Language) in uporabo protokola UDDI (Universal Description Discovery and Integration). Slednji je namenjen ugotavljanju in objavljanju metapodatkov o izbrani spletni storitvi, tako da jo aplikacija lahko najde. Za večino specifikacij s področja spletnih storitev skrbi konzorcij W3C. Spletni servisi so zamišljeni kot nadgradnja SOAP protokola, na osnovi katere izrabimo spletno povezanost poslovnih subjektov za povezljivost aplikacij in za izvajanje standardiziranih, od programskega jezika in računalniškega okolja neodvisnih storitev, [URL: http://www.serviam.se/serviam2/ServiamDocuments/LIT-03%20Service%20Design.pdf?POSTNUKESID=ff37f1f7e55f6c0e94a38e93a70422bf];
- WCF (angleško: Windows Communication Foundation) predstavlja Microsoftovo tehnologijo vzajemnega komuniciranja aplikacij v omrežju povezanih računalnikov in je naslednica DCOM tehnologije za Microsoftovo .Net delovno okolje. Tehnologija predstavlja ogrodje za izvedbo programskih rešitev, temelječih na komunikaciji s pomočjo protokola SOAP. Poleg tega je WCF tehnologija zasnovana za podporo porazdeljenega izvajanja (angleško: distributed computing) aplikacij in za izvedbo aplikacij, ki sloni na izvedbi arhitekture storitev (angleško: Service oriented architecture, SOA), kjer odjemalci lahko uporabljajo več storitev in strežniki zagotavljajo več storitev. Izvedba WCF storitev poteka na osnovi izvedbe IIS (angleško: internet information services) storitev strežnikov, ki uporabljajo Windows operacijski sistem;
- Apache Axis predstavlja odprtokodno tehnologijo vzajemnega komuniciranja aplikacij v omrežju povezanih računalnikov. Tehnologija sloni na delovanju SOAP protokola,

opisnega jezika WSDL, izmenjavi XML sporočil in dodatnih programskih vmesnikih za izdelavo in uporabo medsebojno povezanih aplikacij. Uporaba odprtokodnega Axis ogrodja in konkretna izvedba spletnega servisa lahko sloni na uporabi Java ali na uporabi C++ programskega jezika. Axis2 je posodobljeno odprtokodno Axis ogrodje za izdelavo spletnih storitev, ki sloni na SOAP protokolu.

V izvedbi porazdeljenih računalniških rešitev ne smemo pozabiti na poznavanje protokolov in metod za dostopanje do porazdeljenih podatkov, ki bo čim bolj neodvisno od računalniškega sistema, uporabljene programske opreme in vrste podatkovne baze. Na tem področju obstajajo različne specifikacije dostopanja do podatkov:

- ODBC (angleško: open database connectivity), standard podatkovne povezljivosti, izveden z vrsto gonilnikov za posamezne podatkovne baze,
- iODBC (angleško: independent open database connectivity), odprtokodni standard povezovanja podatkov, neodvisen od računalniškega sistema,
- OLE-DB (angleško: Object linking and embedding for data base), novejši Microsoftov standard podatkovne povezljivosti,
- JDBC (angleško: Java database connectivity), javanski standard podatkovne povezljivosti.

Pri konkretni izvedbi porazdeljenega delovanja programskih rešitev se lahko naslonimo na uporabo različnih protokolov in predmetnih modelov. Funkcionalnost klicanja oddaljenih metod je izvedena na različne načine, čemur se v praksi prilagajamo s prirejanjem različnih rešitev (Vidmar, 2002, str. 423). V tej povezavi je koristno poznavanje protokolov, tehtanje transparentnosti in varnosti ponujenih rešitev. V izvedbi lastnih rešitev lahko hitro naletimo na zahtevo po izbiri specifične tehnologije (Java, .Net). Čeprav si različni proizvajalci programske opreme prizadevajo zagotavljati neodvisnost končnih rešitev od uporabljene programske opreme in računalniškega okolja, je koristno poznati ozadje njihovega delovanja in razvojne trende. Lahko rečemo, da je na področju porazdeljenega delovanja računalniških rešitev trenutni zmagovalec spletna povezovalna tehnologija (angleško: web services) in z njo povezan razvoj standardov.

### 4.3.5 Načrtovalski vzorci programskih rešitev (angleško: design patterns)

Po opravljeni fazi zbiranja in analize zahtev uporabnikov se običajno podamo na področje načrtovanja programskih rešitev, kjer pogosto naletimo na izraz načrtovalski vzorec (angleško: design pattern). Uporablja se tudi izraz projektni obrazec (Milićev, 2001, str. 66). Pod tem izrazom razumemo opis vzorca ali predlogo za rešitev problema, ki se ponavlja. Za promotorja uporabe vzorcev v načrtovanju velja ameriški arhitekt avstrijskega rodu Christopher Alexander. Ta je v sedemdesetih letih prejšnjega stoletja širil idejo o uporabi vzorcev pri načrtovanju zgradb in uporabo v urbanizmu. S pomočjo preizkušenih vzorcev načrtovanja naj bi v oblikovanje bivanjskega okolja vnašali rešitve, ki so že bile dobro sprejete med posamezniki in različnimi skupnostmi (Gamma et al. 1995, str. 356).

Ta ideja je porodila tudi dokumentiranje in objavo nekaterih vzorcev za načrtovanje objektno zasnovanih računalniških rešitev. Na osnovi zbranih izkušenj avtorji promovirajo uporabo načrtovalskih vzorcev pri načrtovanju objektnih rešitev. To naj bi bilo še zlasti dobrodošlo za začetnike in neizkušene načrtovalce. Po mnenju avtorjev vzorci preusmerjajo ustaljen način razmišljanja v nove principe reševanja problemov.

Tudi izkušeni načrtovalci računalniških rešitev včasih ne najdejo rešitve za probleme, na katere naletijo, zato pogosto ponovno uporabijo preizkušene vzorce, ki so se v preteklosti že izkazali za uporabne. Izraba načrtovalskih vzorcev naj bi bila zlasti dobrodošla pri načrtovanju objektnih sistemov, kjer želimo doseči ponovno uporabnost računalniških rešitev oziroma komponent.

Avtorji (Gamma et al. 1995), ki so dokumentirano predstavili začetni nabor načrtovalskih vzorcev, slednje deklarirajo kot predloge ali modele, s katerimi sistematično poimenujemo, razlagamo in vrednotimo pomembnejše in ponavljajoče načrtovalske rešitve pri načrtovanju objektno orientiranih sistemov. Vsak vzorec naj bi vseboval štiri temeljne elemente:

- ime vzorca: poimenovanje omogoča oblikovanje besednjaka in načrtovanje na višji stopnji abstrakcije;
- opis problema: opis razloži vsebino problema. Z opisom nakažemo, kdaj je primerno uporabiti vzorec, kako predstaviti algoritem v obliki objekta, kako uporabiti značilne objektne povezave. V opisu lahko navajamo tudi pogoje, ki morajo biti izpolnjeni, preden uporabimo vzorec;
- rešitev: opisuje seznam elementov (objektov in razredov), ki tvorijo vzorec rešitve, njihove relacije, odgovornosti in sodelovanje. Rešitev ni podana eksplicitno za konkreten primer, pač pa nakazuje možnost uporabe v različnih situacijah;
- posledice: predstavljeni so rezultati in morebitne posledice, na katere moramo računati, če bomo uporabili vzorec. So ključne pri vrednotenju alternativnih rešitev in pri razumevanju koristi oziroma posledic morebitne uporabe vzorca.

Načrtovalski vzorci so mišljeni kot pripomoček za načrtovanje. Obravnavamo jih kot abstraktne kalupe delnih rešitev oziroma sestavin elementov za reševanje splošnih problemov objektnega načrtovanja, neodvisnih od področja, za katerega načrtujemo rešitev.

Seznam vzorcev načrtovanja, kot gradnikov za načrtovanje objektnih računalniških rešitev, naj bi se neprestano dopolnjeval. Izvorno je bilo predlaganih 23 osnovnih vzorcev, ki so jih avtorji razdelili v več skupin. Uporaba predlaganih načrtovalskih vzorcev v praktični izvedbi naj bi slonela na zaporedju izvajanja naslednjih korakov (Gamma et al. 1995, str. 29):

- ponovna seznanitev z vzorcem, preučitev posledic, ki jih navaja izbrani vzorec,
- preučitev zgradbe vzorca, udeležencev in sodelovanja med njimi,
- ogled vzorčne kode, ki ilustrira uporabljen vzorec tudi v obliki kode,
- izbira imen za udeležence v vzorcu, ki imajo smisel iz konteksta konkretne rešitve,
- opredelitev razredov,
- opredelitev specifičnih imen za operacije (funkcije) konkretne aplikacije,
- izvedba operacij za izvršitev sodelovanja in odgovornosti udeležencev v vzorcu.

Obsežen začetni nabor vzorcev načrtovanja je tudi avtorje spodbudil k njihovi preglednejši organiziranosti. Vzpostavljen je bil model klasifikacije, ki jih združuje v sorodne skupine in ponuja prostor tudi za nove. Vzpostavljena je delitev po načelu namena vzorcev:

- kreativni: namenjeni so procesom izdelave objektov,
- strukturni: namenjeni so medsebojnemu sestavljanju objektov ali razredov,
- vedenjski: odražajo vzajemnost obnašanja in delitev odgovornosti objektov in razredov.

Drugi kriterij delitve, imenovan obseg, pa opredeljuje, ali se vzorec izvorno nanaša na objekt ali na razred.

Slika 8: Vrste načrtovalskih vzorcev po namenu in obsegu

| | | **N A M E N  V Z O R C E V** | | |
|---|---|---|---|---|
| | | **USTVARJALNI** *(angleško: creational)* | **STRUKTURNI** *angleško: structural)* | **VEDENJSKI** *angleško: behavioral)* |
| **O B S E G** | **RAZRED** | **- vzorec tovarniška metoda** *(angleško: factory method)* | **- vzorec adapter** *(angleško: adapter/class)* | **- interpreter** *(angleško: interpreter)*<br>**- šablonska metoda** *(angleško: template method)* |
| | **OBJEKT** | **- vzorec abstraktna tovarna** *(angleško: abstract factory)*<br>**- vzorec graditelj** *(angleško: builder)*<br>**- vzorec prototip** *angleško: prototype)*<br>**- vzorec edinec** *(angleško: singleton)* | **- vzorec adapter** *(angleško: adapter/object)*<br>**- vzorec most** *angleško: bridge)*<br>**- vzorec kompozicija** *(angleško: composite)*<br>**- vzorec dekorator** *(angleško: decorator)*<br>**- vzorec zrno** *(angleško: flyweight)*<br>**- vzorec fasada** *(angleško: facade)*<br>**- vzorec namestnik** *(angleško: proxy)* | **- veriga odgovornosti** *(angleško: chain of responsibility)*<br>**- ukaz** *(angleško: command)*<br>**- iterator** *(angleško: iterator)*<br>**- posredovalec** *(angleško: mediator)*<br>**- opomnik** *(angleško: memento)*<br>**- opazovalec** *(angleško: observer)*<br>**- stanje** *(angleško: state)*<br>**- strategija** *(angleško: strategy)*<br>**- obiskovalec** *(angleško: visitor)* |

Vir: povzeto po Gamma et al. 1995

Načrtovalske vzorce, ki jih je objavila 'banda štirih' (gang-of-four, tudi GoF), kot njihove avtorje tudi poimenujejo v računalniški srenji, štejemo med pripomočke v fazi načrtovanja sistema. Glede na izjemno zanimanje strokovne javnosti (37 ponatisov prve objave do aprila 2007) lahko sklepamo, da se objavljeni načrtovalski vzorci med uporabniki prištevajo med koristne pripomočke načrtovanja objektnih računalniških rešitev.

S pomočjo vzorcev vzpostavljamo enotni jezik za komunikacijo pri načrtovanju programskih rešitev. Vzorci imajo velik pomen za komunikacijo med programerji. V osnovi se pri uporabi vzorcev osredotočamo na področje dokumentiranja načrtovanih rešitev in na prenos znanja. Načrtovalski vzorci lahko pospešijo razvojni proces, saj zagotavljajo ponovno uporabo preizkušenih receptov. Omogočajo tudi splošno, od končne izvedbe programa neodvisno načrtovanje in dokumentiranje predmetno zasnovanih računalniških rešitev. Usmerjajo način našega razmišljanja v koncepte objektne tehnologije in kot takšni predstavljajo delovni pripomoček tako za začetnike kot za izkušene načrtovalce.

### 4.3.6 Vmesniki

Vmesniki določajo mejno področje, kjer se pri komunikaciji srečujeta dve entiteti. Na področju razvoja programskih rešitev med te entitete štejemo bodisi del strojne opreme bodisi del programske opreme ali osebo, ki uporablja sistem. Vmesniki omogočajo ločevanje metod komunikacije od notranjega delovanja entitete. Na ta način zagotavljamo možnost spreminjanja notranjega delovanja entitet, ne da bi to spreminjanje vplivalo na komunikacijo entitete z zunanjim okoljem. Vmesnik, ki predstavlja stik med računalnikom in osebo, ki ga uporablja, imenujemo uporabniški vmesnik. Poznavanje njegovih značilnosti in priporočil za izvedbo je pomembno pri razvoju programskih rešitev, ki želijo približati uporabo strojne opreme na uporabniku prijazen način.

Sicer pa so vmesniki eden od temeljev modularnega programiranja, izvedbe programskih rešitev, zasnovanih na arhitekturi porazdeljenega delovanja komponent in vseh zasnov programskih rešitev, ki sloni na delovanju posameznih delov, povezanih v mreži. Vmesniki so lahko proceduralno zasnovani, kar je značilno za uporabo tehnologije klicev za izvedbo oddaljenih postopkov, lahko pa so zasnovani na dokumentih, kar je značilno pri uporabi tehnologije spletnih storitev (Pugh, 2006, str. 89).

Izvedba vmesnika mora slediti osnovnim načelom:

- delati mora samo tisto, kar mu velevajo njegove metode (izvedba mora spoštovati dogovor),
- izvedba ne sme povzročati škode,
- če izvedba ne more izpolniti svoje naloge, mora o tem na smiseln način obvestiti klicatelja.

Razlikujemo podatkovne vmesnike in storitvene vmesnike. Med podatkovne vmesnike štejemo tiste, katerih metode (funkcionalnost) so izdelane skladno z razredi, ki vsebujejo v glavnem podatke. Med storitvene vmesnike štejemo tiste, katerih metode delujejo v glavnem na osnovi prejetih parametrov (Pugh, 2006, str. 33). Glede na to, da vmesniki predstavljajo neke vrste podatkovne izmenjevalce jih lahko kategoriziramo tudi na osnovi njihovega dostopanja do podatkov. Ločimo vmesnike tipa povleci (angleško: pull) in tipa potisni (angleško: push). Vmesnik tipa povleci preskrbi informacijo na zahtevo (na primer spletni brskalnik preskrbi informacijo z navedbo URL), vmesnik tipa potisni pa prenaša podatke oziroma informacije na znani naslov (na primer program za elektronsko pošto pošilja na znani naslov).

Izvedba vmesnika lahko vsebuje stanje (angleško: statefull) ali pa ne (angleško: stateless). Razlika med njima je v tem, da je delovanje vmesnikov, neodvisnih od stanja, vedno enako, vsak poziv metode se izvede enako. Pri vmesnikih, ki ločijo stanja, je izvedba klicanih metod odvisna od trenutnega stanja, ki pa se spreminja odvisno od zaporedja klicanih metod.

Vmesniki so namenjeni izvedbi izbranih storitev znotraj splošnega koncepta. Metode, vgrajene v vmesnik, naj bi predstavljale šopek funkcionalnosti iz koncepta, ki pogosto nastopa skupaj ali se medsebojno navezuje. V tej zvezi pogosto zasledimo napotek, naj bodo vgrajene metode kohezivne. Po drugi strani želimo, da je delovanje vmesnika čim manj odvisno od notranje izvedbe funkcionalnosti. To naj bi pomenilo, da konkretno izvedbo funkcionalnosti vmesnika lahko spreminjamo, ne da bi bilo potrebno spreminjati vsebino modulov, ki uporabljajo vmesnik. Takšnemu načelu pravimo ohlapna povezanost.

Dedovanje in polimorfizem sodita med osnovne koncepte v razvoju objektno orientiranih programskih rešitev, ki ju izkoriščamo tudi pri snovanju in uporabi vmesnikov. Dedovanje je relacija, ki definira eno izbrano entiteto na osnovi druge (Gamma et al. 1995, str. 360). Polimorfizem je sposobnost objekta, da nadomesti svoj vmesnik z vmesnikom drugega med izvedbo programa (Gamma et al. 1995, str. 361). Z dedovanjem vmesnika definiramo nov vmesnik na osnovi obstoječega (starševskega). Dedovanje sestavlja dedovanje vmesnika in dedovanje izvedbe. S pomočjo dedovanja izpeljan razred ali objekt pridobi lastnosti izvornega. Splošna oblika polimorfizma je niz razredov, ki vsi izvajajo isti nabor metod. Vsak izpeljani razred lahko poleg podedovanih vsebuje svojsko izvedbo drugih metod. S pomočjo vmesnikov pa zagotovimo takšno delovanje programske rešitve, da lahko več različnih razredov izvaja vse metode, ki jih vmesnik vsebuje.

Prednost uporabe vmesnikov:

- zahteva po oblikovanju hierarhije je izražena pozneje,
- možnost navzkrižne hierarhije,
- lahko zajame splošno razširjeno uporabo (funkcionalnost),
- večja prilagodljivost vlogam v navzkrižni hierarhiji,
- jasnejši pregled nad metodami, ki morajo biti implementirane,
- razred iz druge dedne hierarhije lahko preskrbi storitve vmesnika,
- lahko končamo z množico pomožnih razredov.

Prednost uporabe dedovanja:

- manj pooblaščanja splošnih operacij,
- zajem splošnih atributov,
- lahko zajame splošno razširjeno obnašanje,
- osnovni razredi lahko zagotavljajo splošno izvedbo,
- preprosto dedovanje izvedbe.

Slabost uporabe vmesnikov:

- brez pomožnih razredov lahko pride do podvajanja programske kode.

Slabost uporabe dedovanja:

- težje prilagodljiva na nove razmere,
- lahko nastopijo težave pri prilagajanju spremenjenim vlogam.

Že v fazi načrtovanja se je treba opredeliti bodisi na uporabo podrazredov in dedovanja bodisi na uporabo vmesnikov (Pugh, 2006, str. 85). Dedovanje nam omogoča dodajanje odgovornosti in variiranje uporabljenega algoritma (Gamma et al. 1995, str. 330). Vmesnik nam omogoča prenos odgovornosti na izbrani objekt, ne pa na celoten razred (Gamma et al. 1995, str. 175).

### 4.3.7 Uporaba vmesnikov z vidika uporabe razvojne metodologije

Načrtovanje vmesnikov se navezuje na analizo zahtev sistema. Kot uporaben pripomoček za načrtovanje vmesnikov se pogosto navajajo primeri uporabe (Pugh, 2006). Z njimi lahko hitro predstavimo, kaj sistem dela za uporabnika, in razmejimo, kaj je znotraj in kaj zunaj njega.

Omogočajo tudi predstavitev toka informacij med različnimi akterji sistema in prepoznavanje vlog akterjev. V sistemu kot akter lahko nastopa bodisi razred uporabnikov bodisi vloga uporabnika bodisi drugi sistem. Za vsak primer uporabe lahko naredimo seznam vzajemnega sodelovanja med uporabnikom in sistemom. S tem opredelimo funkcionalne zahteve uporabniškega vmesnika.

Pri načrtovanju vmesnikov se srečujemo s pojmi in koncepti, ki so skupni za množico razvojnih metodologij. Med koncepte, ki imajo veliko skupnega z vmesniki, sodijo odgovornost, vloga in modul. Vmesnik predstavlja nabor odgovornosti za izvedbo vgrajene funkcionalnosti, to je storitev, ki jo vmesnik obljublja. Z moduli izvajamo implementacijo vmesnika. Modul je lahko razred, komponenta ali oddaljena storitev. Z modulom lahko udejanjimo enega ali več vmesnikov, ki lahko odigrajo prepoznano vlogo v sistemu.

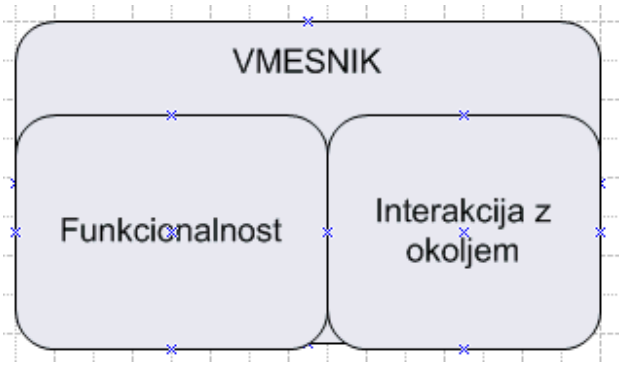
Za pomoč pri načrtovanju vmesnikov nam lahko služi preučevanje stereotipov funkcionalnosti objektov, ki v sistemu nastopajo. Kot stereotip funkcionalnosti opredelimo splošno namensko rabo objekta. Ta ima neposredno zvezo z načrtovanjem vmesnikov. Kot stereotip funkcionalnosti lahko opredelimo:

- zagotavljanje podatkov,
- izvedbo storitve,
- hranjenje trajnih objektov,
- predstavitev poslovnih pravil,
- izdelavo dokumentov,
- izdelavo pogledov, grafični vmesnik.

V fazi analize zahtev si pri načrtovanju vmesnikov lahko pomagamo z notacijo kartic vzajemnega delovanja (slika 9.), poznanih pod imenom CRC (angleško: Class-Responsibility-Collaboration).

Notacija predpostavlja izdelavo indeksnih kartic vmesnikov, kjer na eno stran kartice vpisujemo zadolžitve (funkcionalnost) vmesnika, na drugo pa spisek interakcij z okoljem.

Slika 9: Vzorec indeksne kartice vmesnika




Vir: povzeto po Pugh, 2006

Če želimo, da bo naš načrt vmesnikov bolj razumljiv čim širšemu krogu udeležencev v razvoju končne rešitve, se je med njihovim načrtovanjem priporočljivo poslužiti enega od vzorcev iz skupine kreativnih ali strukturnih načrtovalskih vzorcev (Pugh, 2006), ki se navezujejo na vmesnike.

# 5 JAVNA GOZDARSKA SLUŽBA IN INFORMACIJSKI SISTEM ZGS

## 5.1 Dejavnost službe, izvajalci javne gozdarske službe

V srednje-evropskih državah oziroma deželah je skrb za gozdove preko gozdarske službe v okviru državnih institucij, prevladujoča. Financiranje gozdarskih služb se zagotavlja pretežno iz javnih sredstev bodisi preko državnega proračuna, bodisi preko prihodkov od gospodarjenja z javnimi gozdovi ali pa kombinirano (Ferlin, 2004, str. 65).

Gozdovi sodijo med naravna bogastva, za katere Ustava Republike Slovenije (70. člen) zahteva, da se z zakonom določijo pogoji, pod katerimi se smejo izkoriščati. Kot ekosistem predstavljajo gozdovi nedeljivo celoto, ki je ni mogoče obravnavati s stališča lastniških razmer. Za zagotavljanje sonaravnega in večnamenskega gospodarjenja v skladu z načeli varstva okolja in optimalnega delovanja gozdov kot ekosistema je bila zasnovana javna gozdarska služba. Njene dejavnosti so po Zakonu o gozdovih (ZOG, 1993, 2007):

- zagotavljanje javnega interesa nad gospodarjenjem in rabo vseh gozdov in gozdnega prostora, ne glede na lastništvo,
- usklajevanje tega interesa z lastnikovimi interesi v okvirih, ki jih dovoljuje gozd kot ekosistem,
- svetovanje lastnikom pri gospodarjenju z gozdovi.

Dejavnosti javne gozdarske službe opravljajo:

- Zavod za gozdove Slovenije, ZGS,
- posamezne, z zakonom določene naloge Gozdarski inštitut Slovenije,
- posamezne, z zakonom določene naloge lahko izvajajo tudi koncesionarji.

## 5.2 Zavod za gozdove Slovenije

Uradni začetek delovanja ZGS je 1. 5. 1994. Za opravljanje svojih nalog je organiziran hierarhično na območju vse države (Medved, 2002):

- centralna enota (1),
- območna enota (14),
- krajevna enota (93),
- revir (429).

Vsebinsko gledano je izvajanje dejavnosti ZGS razdeljeno med strokovne oddelke in odseke. Gre za nosilce strokovne pristojnosti nad skupinami aktivnosti, ki so specializirane za posamezna področja dejavnosti javne gozdarske službe. Poleg strokovnih oddelkov so na ZGS osnovani tudi oddelki za skupne zadeve, in sicer za:

- informatiko,
- finančne zadeve,
- kadrovsko pravne zadeve.

ZOG v svojem 56. členu predpisuje vrsto aktivnosti, ki jih opravlja ZGS v vseh gozdovih v zvezi z javno gozdarsko službo:

- zbira podatke o stanju in razvoju gozdov kot ekosistema,
- vodi baze podatkov za svoje delo in za statistično posploševanje,
- spremlja biološko ravnotežje v gozdovih,
- spremlja razvrednotenje in poškodovanost gozdov,
- izdela program varstva gozdov,
- zagotavlja izvedbo ukrepov za varstvo gozdov,
- izdeluje načrte požarnega varstva za gozdove in nudi strokovno pomoč pri gašenju požarov,
- opravlja naloge poročevalske, prognostično-diagnostične službe,
- izdeluje strokovne podlage za program razvoja gozdov Slovenije,
- pripravi program vlaganj v gozdove,
- izdeluje gozdnogospodarske načrte, lovskogojitvene načrte območij ter druge strokovne podlage za gospodarjenje z divjadjo v skladu z zakonom,
- pripravlja načrte za premeno in sanacijo gozdov,
- usmerja in spremlja sanacijo hudourniških območij,
- sodeluje pri prostorskem načrtovanju,
- sodeluje pri usmerjanju, usklajevanju in opravljanju raziskovalne dejavnosti v gozdarstvu in lovstvu,
- pripravlja strokovne podlage za odpiranje gozdov z gozdnimi prometnicami in vodi evidenco o gozdnih cestah,
- načrtuje vzdrževanje gozdnih cest,
- spremlja in nadzira izvajanje gradnje, rekonstrukcij in vzdrževanja gozdnih cest ter prevzema opravljena dela,
- pripravlja metodologijo za zbiranje podatkov o stanju in razvoju gozdov,
- obdeluje podatke in pripravlja informacije o stanju in razvoju gozdov,
- skrbi za popularizacijo gozdov in osveščanje javnosti o pomenu gozdov,
- izvaja svetovanje, izobraževanje in usposabljanje lastnikov gozdov,
- zagotavlja seme in sadike gozdnih drevesnih in grmovnih vrst,
- prevzema opravljena dela v gozdovih in hudourniških območjih, če so financirana ali sofinancirana iz proračuna,
- vodi in odloča o upravnih stvareh iz prvega in šestega odstavka 17. člena, tretjega odstavka 17.a člena, prvega, drugega in četrtega odstavka 21. člena, 21.a člena, prvega odstavka 23. člena, drugega odstavka 24. člena, drugega odstavka 26. člena, prvega odstavka 29. člena, tretjega odstavka 30. člena, tretjega odstavka 31. člena, prvega in tretjega odstavka 42. člena, drugega odstavka 92. člena in drugega odstavka 95. člena. Upravne zadeve iz prvega odstavka 23. člena in prvega odstavka 29. člena Zavod vodi in o njih odloča v skrajšanem postopku. Zavod lahko odločbe o izbiri drevja za možni posek vroča tudi po postopku navadne vročitve, pri čemer se šteje, da je vročitev opravljena deseti dan od dneva odprave odločbe,
- opravlja v gozdovih in gozdnem prostoru naloge nadzora v skladu z določbami tega zakona,
- opravlja v gozdu in gozdnem prostoru naloge neposrednega nadzora v naravi v skladu s predpisi o ohranjanju narave in v tej zvezi naloge prekrškovnega organa,
- o nedovoljenih posegih in aktivnostih v gozdu in gozdnem prostoru pisno obvešča pristojne inšpekcijske službe.

V kontekstu poslovnega modeliranja aktivnosti, ki jih navaja ZOG, lahko obravnavamo kot sestavine poslovnih procesov v izvedbi ZGS. Na neki stopnji posploševanja lahko ugotovimo, da

je ZGS izvajalec procesov bodisi preučevanja gozdov bodisi načrtovanja, katerih končni namen je skrb za optimalno delovanje gozdov kot ekosistema. V okviru izvedbe teh procesov je treba zadovoljiti interes javnosti, interes lastnikov gozdov, interes koncesionarjev upravljanja gozdov in interese drugih organiziranih skupin ali služb, katerih delovanje je vezano na gozdni prostor. Vsi ti so potencialni porabniki oziroma potrošniki storitev ZGS.

Z vidika strateškega načrtovanja informacijske podpore za izvedbo poslovnih procesov bi lahko rekli, da so osnovni procesi znani. To, kar lahko pomembneje vpliva na nadaljnje usmerjanje razvoja informacijske podpore, pa je razlikovanje procesov, v katerih poteka notranja izdelava strokovnih gradiv, od procesov, s katerimi izvajamo storitve, namenjene zainteresirani javnosti. Možnosti, ki jih ponuja sodobna informacijsko komunikacijska tehnologija, omogočajo njihovo porazdeljeno izvajanje na osnovi bodisi centralnih bodisi porazdeljenih podatkovnih virov.

## 5.3 Informacijski sistem ZGS, dosežena stopnja informatizacije

Stopnjo informatizacije lahko merimo z različnimi merili, med katera lahko uvrstimo obseg razpoložljive informacijsko komunikacijske strojne in programske opreme, računalniško pismenost uporabnikov, kakovost kadra, ki skrbi za informatiko, in obseg sredstev, namenjenih za informatizacijo.

Delujoči informacijski sistem v ZGS ločeno obravnava zbiranje, preučevanje in izkazovanje podatkov gozdarskih vsebin od ostalih podatkov, pomembnih za delovanje organizacije (računovodsko-kadrovski sistem). Gozdarski del informacijskega sistema sloni na relacijski podatkovni bazi, nad katero se izvaja nekaj bolj ali manj povezanih programskih rešitev.

Doseženo stopnjo informatizacije v ZGS lahko predstavimo s pomočjo nekaterih notranjih razvojnih dokumentov in analiz ter na osnovi delov poročil mednarodnega projekta EU Twinning-Light Sl2004-IB-AG-05, ki je preučeval stopnjo dosežene informacijsko komunikacijske infrastrukture v naši javni gozdarski službi (Schikorr, 2007).

Izsledki notranje ankete in analize, ki je bila izvedena med uporabniki informacijske tehnologije v ZGS (Mikulič, 2005), so se odrazili v naslednjih ugotovitvah:

- več kot 50 % računalniške opreme je tehnološko zastarane,
- večina krajevnih enot nima dostopa do omrežja ZGS,
- lastni programi, ki so izdelani s Clipper in MS FoxPro programirnimi jeziki, zahtevajo DOS okolje, ki ga sodobnejši operacijski sistemi v prihodnje ne bodo več podpirali,
- ker smo nove računalniške rešitve razvijali s sodobnejšimi programirnimi jeziki, imamo sedaj preveliko število računalniških rešitev znotraj enega podsistema (primer podsistem gozdarske evidence),
- informatiki na centralni enoti in območnih enotah porabijo preveč delovnega časa za prenos podatkov iz krajevnih enot na centralno enoto in za odpravo napak,
- sistem posredovanja podatkov iz krajevnih enot na območno enoto in nato na centralno enoto težko zagotovi hitro izmenjavo, konsistentnost in varnost podatkov,
- zaradi premajhnih razpoložljivih sredstev za izobraževanje informatikov in uporabnikov informacijske tehnologije v ZGS, je v ZGS premalo znanja za hitrejše uvajanje sodobnejših načinov obravnave podatkov.

Prepoznani problemi so se odrazili v postavitvi naslednjih ciljev:

- posodobiti in dopolniti strojno računalniško opremo in vzpostaviti računalniško povezavo med vsemi organizacijskimi enotami ZGS,
- omogočiti dostop do zunanjih baz podatkov, ki so pomembne za vodenje upravnih postopkov, za vse krajevne enote (podatki GURS, Zemljiška knjiga),
- omogočiti hitrejšo izmenjavo podatkov s pogodbenimi strankami (SKZG),
- izboljšati dokumentiranost računalniško podprtih postopkov, računalniških programov in podatkov v IS,
- zagotoviti večjo varnost dostopa do podatkov in sledljivost spreminjanja podatkov,
- povečati ažurnost, zanesljivost in razpoložljivost najpomembnejših podatkov na vseh organizacijskih ravneh v ZGS,
- narediti prijaznejše uporabniške vmesnike,
- povečati uspešnost in učinkovitost pri razvoju računalniških rešitev,
- zmanjšati porabo časa informatikov za delo s podatki in nameščanjem novih verzij programov,
- povečati znanje s področja uporabe IT,
- v obravnavo podatkov vključiti sodobnejšo opremo (dlančniki, GPS) in računalniške rešitve (spletne rešitve, direktorski IS).

Z zastavljenimi cilji povezana prizadevanja pri izravnavi in obračanju razvojne krivulje v sledenje sodobnejšim trendom na področju informacijsko komunikacijske tehnologije so se v ZGS v zadnjih dveh letih (2006-2007) odrazile v izvedbi naslednjih aktivnosti:

- s prehodom na uporabo večuporabniškega podatkovnega sistema MySQL,
- s povezavo vseh organizacijskih enot v enotno mrežo (HKOM),
- z izdelavo nekaterih novih lastnih aplikacij, ki delujejo v porazdeljenem okolju,
- s pretvarjanjem obstoječih podatkov iz datotečnega sistema (dbf) v podatkovno bazo MySQL,
- s sklenitvijo dogovora o gostovanju ZGS na centralnem strežniku MKGP in koriščenju njihove podatkovne baze ORACLE,
- z organizacijo izobraževanja nekaterih kadrov za informatiko za uporabo orodij objektnega programiranja (Java),
- z nabavo od gozdarskih vsebin ločenega informacijskega sistema za podporo računovodstva in kadrovskih evidenc, ki sloni na podatkovni bazi ORACLE.

Ugotovitev stanja in dosežena stopnja informatizacije ZGS za podporo procesov s področja dejavnosti javne gozdarske službe sta bila predmet obravnave tudi v mednarodnem projektu EU Twinning-Light Sl2004-IB-AG-05. Del izvedbe projekta znotraj tretje aktivnosti (Activity 3: Analysis of soft- and hardware: content and information) je bil posvečen vrednotenju programske opreme za ravnanje s podatki, administriranju podatkov in zagotavljanju informacij v ZGS. Izsledki analize, objavljeni maja 2007, povzemajo naslednje ugotovljeno stanje (Schikorr, 2007):

- Trenutno ni na voljo programske opreme, ki bi omogočala nadzor centralne baze podatkov, upravljanje podatkov in administracijo podatkov.
- Informacijski sistem ZGS ni popoln, saj upravljanje (zbiranje, procesiranje, hranjenje, manipulacija, analiza in zagotavljanje) podatkov temelji na zbirki različnih gozdarskih programov, ki ločeno podpirajo posamezne procese s področja javne gozdarske službe. Nekateri podatki so kljub uvedbi enotne podatkovne baze še vedno hranjeni v različnih oblikah in na različnih mestih. Na osnovi podatkovne zgradbe in načina hranjenja trenutni gozdarski informacijski sistem ne more biti neposredno preveden v podatkovno bazo (DBMS).

- Vzpostavljen je sistem štirinajstih podatkovnih baz na območnih enotah, od kjer se podatki stekajo na enega nameščenega na centralni enoti ZGS. Zasnovan in deloma izdelan je nov nabor programov, ki omogoča prevedbo lokalnih podatkov (dbf) in neposredno zbiranje in procesiranje podatkov iz lokalnega okolja v centralni podatkovni sistem (MySQL).
- Vzpostavljen je model predstavitve gozdarskih podatkov v pristojnosti ZGS, ki je združljiv s podatki Ministrstva za kmetijstvo, gozdarstvo in prehrano (MKGP). Ažurni gozdarski podatki so na voljo skupaj z drugimi podatki iz kmetijskega resorja. Podatki so vzpostavljeni na centralnem podatkovnem strežniku MKGP v okolju podatkovne baze Oracle.
- Podatkovni model geografskega informacijskega sistema (GIS) sloni na aplikacijah, ki se izvajajo na posameznih delovnih postajah. Podatki so hranjeni na datotečnih strežnikih in so deljeni na posamezne teritorialne enote, ki predstavljajo tudi okvir za časovno dinamiko posodabljanja podatkov. Prava GIS funkcionalnost na osnovi centralne podatkovne baze, ki bi omogočala poizvedbe in analizo nad celotnim prostorom delovanja javne gozdarske službe, še ni vzpostavljena.
- Organizacijske enote ZGS (14 območnih enot in 93 krajevnih enot) je bilo v letu 2007 medsebojno povezanih v zaprtem omrežju HKOM. Možna je izmenjava podatkov med vsemi organizacijskimi enotami.
- Zaznano je ozko grlo na področju razpoložljivih kadrovskih virov pri nadaljnjem uvajanju hranjenja podatkov na centralnem podatkovnem strežniku in povezovanju lokalnih podatkovnih baz (DBMS).

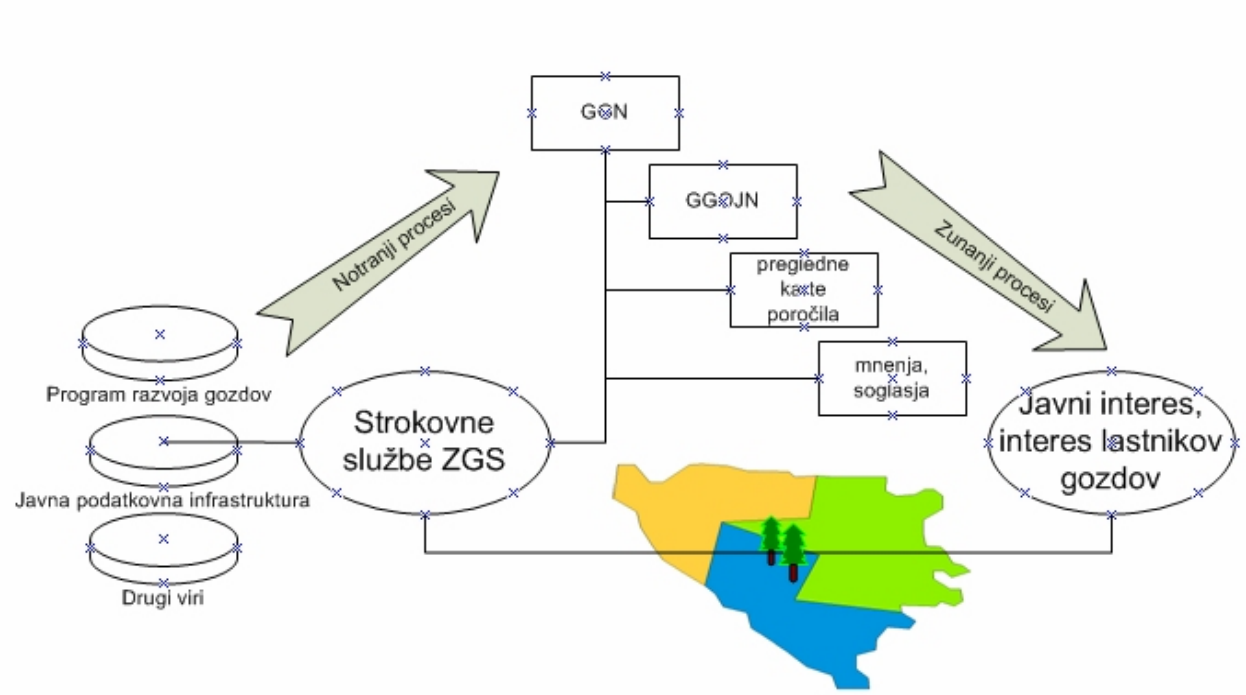
## 5.4 Organiziranost in vloga prostorskih podatkov v informacijskem sistemu ZGS

Zgodovinsko gledano izhaja delovanje ZGS iz načina delovanja nekdanjih (14) gozdnih gospodarstev, ki pri upravljanju s prostorskimi podatki pred reorganizacijo gozdarske službe niso ubirala enakih poti. Informacijska podpora procesom javne gozdarske službe v izvedbi ZGS je bila v začetku svojega delovanja usmerjena predvsem v poenotenje načina zbiranja in upravljanja podatkov. Obsežnejše aktivnosti so bile usmerjene v vzpostavitev digitalnih slojev gozdarskih prostorskih entitet ter njihovo povezovanje z atributnimi podatki, ki so predstavljali dotedanje ogrodje informacijskega sistema. Veliko začetnih aktivnosti je bilo osredotočenih na vzpostavitev dobre prakse digitalizacije entitet iz razpoložljivih kartnih virov, uvajanje namiznih programov za kartografijo in izobraževanje uporabnikov za njihovo uporabo.

Ena od posebnosti upravljanja podatkov v okviru delovnih procesov ZGS je desetletni ciklus osveževanja podatkov. Razlog za takšen način posodabljanja podatkov izhaja iz organiziranosti in načina dela, ki ugotavlja trende v razvoju gozdov s cikličnim primerjanjem podatkov po posameznih relativno trajnih območjih gozdov (gozdnogospodarskih območjih in gozdnogospodarskih enotah). Rezultati analiz ciklično zbranih podatkov so podlaga za strokovno usmerjanje ravnanja z gozdovi. Analize podatkov zagotavljajo informacije o trendih razvoja gozdov, ki jih s pomočjo smernic gozdnogospodarskih načrtov (GGN) lahko bodisi spreminjamo bodisi pospešujemo. Posledično imajo smernice 10-letni ciklus veljavnosti. V tem obdobju s podrobnimi izvedbenimi načrti gozdnogojitvenega načrtovanja (GGOJN) poteka uresničevanje dolgoročnejših smernic in svetovanje lastnikom gozdov. Ta posplošena predstavitev pretoka informacij, ki izhaja iz zbiranja in analize podatkov o gozdovih, seveda ne odraža vseh storitev in spremljevalnih aktivnosti ZGS v izvajanju javne gozdarske službe.

Namen te grobe predstavitve zbiranja in uporabe podatkov služi za ilustracijo cikla prenove podatkov na izbranem prostoru ter za ilustracijo možne delitve delovnih procesov glede na skupine njihovih potrošnikov. Procese v izvedbi ZGS lahko delimo na tiste (notranje), ki producirajo rezultate in izdelke, namenjene usmeritvam gospodarjenju in razvoju gozdov. Na drugi strani se srečujemo z drugimi procesi (zunanjimi), kjer rezultate in izdelke notranjih procesov soočamo z interesom lastnikov in javnosti. V teh procesih se običajno izvaja podrobnejša strokovna interpretacija dolgoročnih smernic razvoja gozdov, podrobno načrtovanje, izdajanje odločb, vrednotenje gozdov, podajanje mnenj, soglasij in izvedba različnih drugih storitev.

Slika 10: Podatkovni viri, procesi, izdelki in storitve ZGS




V izvajanju notranjih in zunanjih procesov ZGS prihaja do prepletanja uporabe lastnih in podatkov drugih skrbnikov prostorskih ter drugih podatkov. Obstaja tudi trajen interes za posredovanje izbranih gozdarskih podatkov drugim uporabnikom.

Narava dela javne gozdarske službe je tradicionalno osnovana na uporabi prostorskih podatkov iz različnih podatkovnih virov. Analizo in upravljanje podatkov s prostorsko lokacijo podpirajo prilagojeni geografski informacijski sistemi. Namenjeni so zajemu, hranjenju, analizi in upravljanju podatkov s prostorsko referenco (Šumrada, 2005, str. 81). Delujejo na dveh principih predstavitve podatkov, vektorski in rastrski. Vektorska predstavitev je namenjena predstavitvi diskretnih entitet (razmejenih), rastrska pa zveznih. Obstaja vrsta značilnosti, ki lahko vpliva na prednostno izbiro ene ali druge predstavitve, od katerih predvsem rastrska za ZGS vsebuje še neodkriti oziroma preslabo izkoriščen potencial. Trenutni način vzdrževanja gozdarskih prostorskih podatkov sloni na razpršeni in nepovezani uporabi programov namizne kartografije, ki podpirajo tako rastrski kot vektorski način prikazovanja podatkov. Vzdrževanje podatkov sloni na datotečnem hranjenju in prenašanju podatkov na centralni datotečni strežnik.

V zadnjem obdobju, ko je ZGS uspelo vzpostaviti prostorske podatke (v vektorski obliki predstavitve) za celoten prostor, za katerega usmerja gospodarjenje, postaja čedalje bolj aktualno hranjenje vseh geokodiranih podatkov v centralni bazi.

Izbira ustrezne podatkovne baze, ki bi omogočala hranjenje geokodiranih podatkov na centralnem mestu, je aktualna iz več razlogov. Eden od njih je vzpostavitev enotne točke dostopanja do trenutno aktualnih podatkov ZGS. Drugi se navezuje na zasnovo in izrabo geopodatkovne infrastrukture, na kateri bi načrtovali prihodnje računalniške rešitve. Vzpostavitev centralne podatkovne baze je aktualna tudi zaradi vrste storitev, ki jih ZGS izvaja na področju odkazovanja drevja za posek ter vlaganj v gozdove in spremljanja načrtovanja in izvajanja del lastnikov v gozdovih. Lahko bi rekli, da so zahteve časa presegle kakovost storitev, ki jih lahko ZGS izvaja brez centralne podatkovne baze.

Če povzamemo zgornje navedbe, lahko rečemo, da je v zasnovi informacijskega sistema ZGS treba upoštevati naslednje dejavnike:

- prostorsko razpršeno organiziranost delovanja,
- izvajanje skupine procesov, ki sloni pretežno na vzajemnem delovanju strokovnih oddelkov,
- izvajanje skupine procesov, ki vključuje sodelovanje lastnikov in zainteresirane javnosti,
- uporaba geokodiranih podatkov,
- zagotavljanje dostopa do podatkov javne gozdarske službe drugim uporabnikom,
- uporaba javne prostorske podatkovne infrastrukture.

Na osnovi dosedanjega preučevanja zasnove in razvoja informacijskih sistemov lahko ugotovimo, da sodobna informacijsko komunikacijska tehnologija omogoča izvedbo takšnih računalniških rešitev, kjer je v okviru enega sistema možno upoštevati vse zgoraj navedene dejavnike.

## 5.5 Postopnost uvajanja novih rešitev, vključevanje obstoječih

V snovanju informacijske podpore se zgodi, da ni možno zavreči obstoječih vzorcev računalniških rešitev in vzpostaviti povsem novih. V določenem okolju se lahko srečamo z zahtevo po ohranitvi ustaljenega načina pridobivanja ali izkazovanja podatkov. Nekatera delovna okolja so s temi omejitvam manj, druga bolj obremenjena.

V gozdarstvu je pomen podatkov, zbranih v preteklosti, še zlasti pomemben za dolgoročno spremljanje vzorcev sprememb v odnosu na pretekle ukrepe in načrtovanje prihodnjih ukrepov. Vsi, skozi zgodovino pridobivanja zbrani podatki, so potencial za poglobljene analize in časovne primerjave v dinamiki razvoja (degradacije) gozdov kot ekosistema. Med strateškimi smernicami prenove delovanja informacijskega sistema v gozdarstvu bomo zagotovo naleteli na zahtevo po uporabi v preteklosti zbranih podatkov. V tej navezi je potrebno upoštevati vidik vsebinske, časovne in prostorske primerljivosti nekdanjih in morebitnih novih podatkov, s katerimi spremljamo in izkazujemo lastnosti gozdnega prostora. Prehod na predmetno modeliranje oziroma snovanje računalniških rešitev na osnovi predmetnega modela iz domene gozdarstva bi moral to omejitev upoštevati. To pa lahko pomeni tudi zavestno ohranjanje nekaterih dosedanjih računalniških rešitev in izdelavo novih na osnovi integracije obstoječih.

Pri prenovi računalniških rešitev v ZGS lahko v tej zvezi praviloma pričakujemo zahteve po ohranjanju obstoječih delujočih rešitev, ki jih ni mogoče kar zavreči. Nekatere so že deloma

posodobljene na uporabo skupne podatkovne baze, druge imajo pomen za specifične analize, prilagojene uporabi podatkov iz določenega preteklega časovnega obdobja.

Upoštevanje teh zahtev navaja na zahteve po sobivanju preteklih in novih računalniških rešitev. Pri prenovi oziroma posodobitvi informacijskega sistema lahko pod takšnimi pogoji pričakujemo zahtevo po podrobnejši zasnovi informacijske arhitekture in komponentni zasnovi računalniških rešitev.

Zagotovo bodo nekatere nove predlagane računalniške rešitve v ZGS morale upoštevati možnost uporabe podatkov, zbranih v različnih časovnih obdobjih. V modelu delovanja entitet informacijskega sistema je torej že v zasnovi potrebno upoštevati aktivnosti, ki se navezujejo na uporabo podatkov v različnih časovnih obdobjih. Na to se navezuje tudi zahteva po načrtnem vzdrževanju kataloga podatkov in medsebojno primerljivih podatkovnih modelov, ki so bili aktualni v preteklih obdobjih.

## 5.6 Motivi za posodobitev informacijskih storitev

Če opredelimo informacijsko storitev kot servisiranje podatkov med podatkovnim izvorom in ponorom, potem je zagotavljanje informacijskih storitev povezano z vsemi aktivnostmi in vsemi udeleženci, ki se pojavljajo v okviru dejavnosti, tako profitnih kot neprofitnih organizacij.

Sodobna informacijsko komunikacijska infrastruktura omogoča povezovanje razpršenih podatkovnih virov, računalniških rešitev in končnih uporabnikov. Nadomeščanje zastarelih arhitektur računalniških rešitev s sodobnejšimi in približevanje storitev do vsakega zainteresiranega uporabnika je značilnost prenove informacijskih sistemov v profitnem okolju, ki motivira prenovo slednjih tudi v neprofitnem delovnem okolju. Boljša izraba možnosti, ki jih omogoča spletna tehnologija v povpraševanju in ponudbi storitev, je danes eden glavnih motivov posodabljanja informacijske podpore.

Na področju delovanja neprofitnih organizacij se motivi posodabljanja odražajo tudi v trendu uvajanja storitev v obliki novih E-storitev in prizadevanjih za boljše informiranje javnosti o področju delovanja organizacije. V procesih evropske integracije lahko pričakujemo zbliževanje nekaterih storitev in izmenjavo podatkov iz domene delovanja služb ali servisov, kjer pomen njihovega dela lahko presega lokalne okvire.

Kot primer lahko uporabimo poenotenje modela delovanja organizacij, ki skrbijo za usmerjanje razvoja naravnega okolja. Skrb za naravno okolje ne pozna administrativnih meja. Pričakujemo lahko zbliževanje modelov načrtovanja, analize, zbiranja podatkov, zagotavljanja njihove medsebojne primerljivosti in skladne rabe znotraj posameznih sektorjev javnih služb. Tudi izvajanje storitev na njihovi osnovi bo verjetno zaradi smernic poenotene politike kmetijstva in gozdarstva bolj poenoteno. O tem se lahko prepričamo že ob hitrem pregledu različnih projektov, ki potekajo v okviru nekaterih evropskih direktiv (na primer Directive 2003/4/EC - Environmental Information Directive).

Del navedenih ugotovitev lahko prenesemo tudi na področje delovanja ZGS. Pooblastilo za izvajanje javne gozdarske službe predstavlja obsežen prostor za izvajanje procesov tako preučevanja naravnega razvoja gozdov kot tudi usmerjanja racionalne izrabe gozdov po načelih trajnosti in ekološke sprejemljivosti. Izvajanje informacijskih storitev ZGS je večstopenjsko in prednostno namenjeno podpori specifičnih nalog strokovnih oddelkov v izvedbi notranjih procesov. Navezuje se na strokovne analize, ugotavljanje trendov v razvoju gozdov in

predpisovanje smernic delovanja za krepitev izbranih funkcij gozdov. Rezultate teh informacijskih storitev kot vhode uporabljajo procesi podrobnejšega usmerjanja rabe in spremljanja gospodarjenja gozdov. Tu naletimo na drug obsežen paket informacijskih storitev, povezan z izdajanjem odločb, evidentiranjem odkazanega drevja za posek, spremljanjem količin poškodovanega drevja, spremljanjem nege, varstva in ostalih vlaganj v gozdove.

Splošna načela o uveljavljanju javnega interesa v vseh gozdovih, ne glede na lastništvo, vpogled v ekološke, socialne in proizvodne funkcije gozda, pregled nad omejitvami ali dostopnostjo v gozdove, pregled omrežja gozdnih cest, možnostjo rekreacije v gozdovih, območja zavarovanih rastlinskih in živalskih vrst in predstavitev drugih gozdarskih vsebin v interesu javnosti, sodijo na področje informacijskih storitev, ki so namenjene širši javnosti.

Na osnovi predlagane delitve procesov, ki jih izvaja ZGS, na notranje in zunanje, si lahko pomagamo pri spreminjanju značilnosti arhitekture programskih rešitev v ZGS. S pojavom namiznih računalnikov v gozdarstvu se je informacijska podpora izvajala ob podpori enoslojne arhitekture računalniških rešitev, kjer se dostop do podatkov in procesiranje poslovnih pravil izvajata na posameznih, sprva nepovezanih računalnikih. Ob ustanovitvi je ZGS prevzel enoslojno informacijsko arhitekturo. Z razvojem, sprva lokalnih mrež, se je začela počasna preobrazba te arhitekture v dvoslojno, kjer so datotečni strežniki sprva predstavljali samo skladišča za veljavne podatke in stare aplikacije niso bile predelane na ločevanje delovanja programa in dostopanja do podatkov. Šele v današnjem obdobju se z novimi aplikacijami izvaja prava dvoslojna arhitektura, nekatere aplikacije pa se še vedno izvajajo po modelu enoslojne arhitekture. Obe zasnovi arhitekture programskih rešitev sta značilni za podporo procesov, ki jih v nalogi obravnavam kot notranje. Povezanost organizacijskih enot v enotno mrežo omogoča ZGS izrabo protokolov interneta tako za vzpostavitev zasebne računalniške mreže (intranet) kot tudi navezavo na svetovni splet in na povezane storitve. S povezanostjo organizacijskih enot ZGS v skupno računalniško mrežo se odpirajo drugačne možnosti računalniške podpore tako notranjim kot zunanjim procesom. Če bi želeli v ZGS vzpostaviti prožnejši model informacijske arhitekture, bi bilo koristno izvesti revizijo trenutne arhitekture programskih rešitev tudi s stališča nekaterih značilnosti procesov, ki jih izvaja.

Snovanje novih računalniških rešitev, ki bi omogočale podporo tako notranjim kot zunanjim procesom ZGS, terja razmislek o preobrazbi sedanje v večslojno arhitekturo programskih rešitev. Računalniške mreže, strežniki (podatkovni in aplikativni) ter odjemalci so temeljni nosilci večslojne arhitekture programskih rešitev. V pogojih tako razpršene uporabe sistemskih virov se je potrebno bolj kot do sedaj posvečati snovanju novih računalniških rešitev v povezavi z arhitekturo sistemskih virov.

Lahko zaključimo, da zagotavljanje vzajemnega delovanja različnih aplikacij, namenjenih informatizaciji notranjih procesov, in hkratna podpora zunanjim procesom s pomočjo spletne tehnologije in porazdeljenih sistemskih virov sodijo med osnovne motive za posodobljenje informacijske podpore v ZGS.

## 5.7 Načrtovanje posodobitve

Za načrtovanje posodobitve informacijske podpore v neprofitnem delovnem okolju veljajo enake zakonitosti in razvojne predpostavke kot za profitno okolje. V splošnem izvajamo načrtovanje na osnovi strateškega načrta delovanja organizacije, organiziranosti, načina izvajanja poslovnih procesov, organizacije podatkovnih virov in dosežene ravni informacijsko komunikacijske infrastrukture. Odgovornost za celovit razvoj oziroma posodobitev informatike neprofitnih

organizacij sega do nivoja njihovih ustanoviteljev. Ena od ovir, ki lahko nastopajo v neprofitnem sektorju na področju strateškega načrtovanja, je ta, da je pogosto v tem okolju moč strateškega odločanja v večji meri na strani ustanovitelja oziroma donatorja kot na strani izvajalca. To dejstvo je potrebno upoštevati tudi v načrtovanju posodabljanja informacijske podpore.

V načrtovanju posodobitve se v vsakem delovnem okolju lahko srečujemo z zahtevami ohranjanja podedovanih razmer ali z omejitvami, ki vplivajo na možnosti našega delovanja. Kadar nimamo prostih rok za strateško preoblikovanje načina delovanja, bo razvojna vizija slonela na tistih osnovah, ki jih prepoznavamo kot zdrave temelje, primerne za načrtovanje nadaljnjega razvoja ter na tistih dejavnikih, na katere lahko vplivamo. Običajno je sočasno z načrtovanjem posodobitve informacijske podpore potrebno ocenjevati usposobljenost kadrovskih virov, nosilcev prenove, organiziranost delovanja ter vpliv zunanjih dejavnikov in obseg dodatne zunanje pomoči, ki bo morda potrebna. Glede na izbrano metodo strateškega načrtovanja naj bi razvojne smernice vsebovale seznam dejavnikov, ki smo jih upoštevali kot pomembne za izvedbo razvojnega načrta, pa tudi oceno pričakovanih učinkov.

S stališča delovanja službe za informatiko so v nadaljevanju predstavljeni nekateri najpomembnejši dejavniki, ki bi jih bilo potrebno upoštevati ob prenovi informacijske podpore in ob uvajanju predmetne informacijske tehnologije v ZGS. Predstavljeni dejavniki izhajajo iz različnih, v nalogi preučevanih primerov razvoja predmetnih računalniških rešitev in so podrobneje obravnavani z vidika zaznanega stanja informatike v ZGS. Predstavljajo predvsem tisti del množice dejavnikov, ki jim pripisujemo pomembnejšo vlogo za uspešnost projektov lastne izdelave predmetnih računalniških rešitev.

V splošnem predstavljajo pomembnejše dejavnike, ki jih je koristno upoštevati ob uvajanju predmetne informacijske tehnologije v vsako delovno okolje. Glede na ugotovitve, da se možnosti delovanja, oblike vodenja, razpoložljivi kadrovski viri, obseg informacijskega znanja v neprofitnem okolju pogosto močno razlikuje od profitnega, so dejavniki predstavljeni tudi v obliki povzetkov oziroma delnih zaključkov, ki predstavijo pomen dejavnika v neprofitnem delovnem okolju na splošno.

### 5.7.1 Načrtovanje računalniških rešitev

Načrtovati računalniško rešitev ali ne, to je retorično vprašanje. Morda si je v neprofitnem delovnem okolju bolj umestno zastaviti drugi dve vprašanji, in sicer: kako načrtovati projekt izdelave lastne računalniške rešitve ter na kakšen način in kako podrobno predstaviti načrt prihodnje predmetne računalniške rešitve?

Standardi razvoja programske opreme in procesni vzorci razvoja računalniških rešitev učijo, da poteka izvedba računalniške rešitve v življenjskem ciklu, za katerega je značilno izvajanje različnih procesov (standard IEEE 1220-2005, Ambler, 1998). Procesi ali delni procesi, kot so zbiranje zahtev uporabnikov, načrtovanje in izdelava rešitve, testiranje, uvajanje, merjenje doseženih ciljev, nadzor kakovosti komponent ali dokumentacije, zasnova in spremljava izvedbe projekta na osnovi predvidenih nalog ter vrsta drugih podpornih procesov, se medsebojno prepletajo v obsežno in navidezno neobvladljivo gmoto procesov. Temeljni pristop, ki ga ilustrira večina v nalogi obravnavanih primerov razvoja računalniških rešitev, je delitev kompleksnih nalog in povezanih procesov na večje ali manjše sestavne dele. To dopušča parcialno izvedbo nalog na osnovi znanja in dela udeležencev razvoja, ki so jim dodeljene različne vloge.

Univerzalni model razgradnje sistema ne ločuje med profitno ali neprofitno naravnanimi izvajalci.

V grobem bi lahko rekli, da je že vsaka razgradnja sistema na sestavine lahko podlaga za načrtovanje in izvedbo računalniške rešitve. V odvisnosti od pogleda razgradnje sistem običajno predstavimo z abstraktnimi modeli, ki vsebujejo sestavine, značilne za izbrani pogled. Za izdelavo računalniške rešitve predstavlja temeljni pogled na sistem predstavitev modela elementov in njihovega vzajemnega delovanja. Z uvedbo različnih pogledov delamo sistem bolj berljiv (razumljiv) posameznim vlogam udeležencev v razvojni skupini za izdelavo računalniške rešitve. Vloge udeležencev pripisujemo na eni strani modeliranju sistema in načrtovanju (predstavitvi rešitve problema) ter na drugi strani reševanju (izvedbi) predstavljenega problema v obliki računalniške rešitve. Ta je povezana s produkcijo, običajno z izdelavo sestavin, ki jih predstavimo z delnimi načrti. Delne načrte uporabimo kot pripomočke pri izdelavi predstavljene sestavine, na njihovi osnovi pa lahko načrtujemo tudi projekt izvedbe v obliki seznama opravil, potrebnega časa, vlog izvajalcev in drugih virov, potrebnih za izdelavo dela računalniške rešitve.

Predstavitev delovanja sistema v obliki modela sestavin izpostavlja modeliranje kot temeljno komunikacijsko sredstvo procesa razvoja računalniške rešitve. Njegova uporabna vrednost sega od sodelovanja z uporabniki (naročniki) preko procesa izdelave do procesa vzdrževanja ali procesa integracije z drugimi rešitvami. Nobena od preučevanih razvojnih metodologij ne pušča dvoma o izjemni pomembnosti modeliranja delovanja sistema in njegovega razumevanja za vse udeležence razvoja.

Poleg nesporne zahteve po predstavitvi delovanja sistema, je drugi najpomembnejši vidik izdelave predmetne računalniške rešitve v lastni režiji vzpostavitev organizacije projekta za izvedbo rešitve. Z njim opredelimo vloge in odgovornosti nosilcev vlog za izvedbo končne rešitve. Glede na to, da je slednja skupek različnih sestavin, s projektom opredelimo tudi izdelke, ki jih je potrebno izdelati (module, komponente, alternativne rešitve, podrobne načrte in dokumentacijo rešitve). Skladno z medsebojno povezanostjo izdelkov opredelimo tudi zaporedje in urnik njihove izdelave. V organizacijo izvedbe projekta lahko vključimo tudi vrsto dodatnih procesov, kot je kontrola skladnosti in kakovosti izdelanih sestavin, in druge podporne procese, s katerimi skrbimo za optimizacijo vloženega dela.

S stališča načrtovanja izvedbe računalniške rešitve smo opravili temeljno delo, če uspemo predstaviti prihodnjo računalniško rešitev v obliki modela vzajemno delujočih sestavin, ki bo razvojno ekipo usmerjala v njihovo konkretno izdelavo in če lahko taisti model uporabimo tudi kot izhodišče za organizacijo izvedbenega projekta.

Podrobnemu načrtovanju računalniške rešitve se lahko izognemo z razvojnim pristopom, ki sloni na izdelavi prototipov delujočih računalniških rešitev. Vsakokratna izdelava izboljšane verzije delujoče rešitve se tesno navezuje na neprestano in tesno sodelovanje z uporabnikom (naročnikom). Dialog z njim v tem primeru praviloma bolj sloni na predstavitvah funkcionalnosti vsakokratne računalniške rešitve, manj pa bo transparenten pregled sestavin na rešitve in njihovo vzajemno delovanje. Tudi projekt izdelave računalniške rešitve bo v tem primeru potekal drugače, saj vsaka ponovna izdelava izboljšanega prototipa praviloma pomeni izvedbo lastnega projekta. V neprofitnem okolju, kjer ni vzpostavljene odgovornosti uporabnika (naročnika) ali, kjer slednji nima jasne predstave o želeni končni rešitvi, se takšen agilni pristop lahko izkaže za manj ugodnega od pristopa, ki sloni na obsežnejšem načrtovanju.

Ne glede na dileme o tehtnosti zahtev bolj ali manj podrobnega načrtovanja za razvoj računalniških rešitev, lahko zaključimo, da je načrtovanje končne rešitve zagotovo potrebno.

Navezuje se tudi na projekt izvedbe, v katerem poteka komunikacija med izvajalci in uporabniki (naročniki). Na osnovi teh spoznanj in zaznanega stanja lahko vzpostavim trditev, da predstavlja za ZGS modeliranje sistemov, za katere se izvaja lasten razvoj računalniških rešitev ter uporaba standardnega jezika za modeliranje, izrazito podcenjen razvojni potencial.

Tudi organizacija in izvedba projektov razvoja računalniških rešitev, ki jim preučevani vzorci razvoja pripisujejo velik pomen, sodita v kategorijo spregledanih in v ZGS preslabo upoštevanih načinov dela. Uvedbo modeliranja pri načrtovanju rešitev in uvedbo projektnega dela uvrščam med temeljne dejavnike uspešne izdelave računalniških rešitev v ZGS.

### 5.7.2 Zasnova informacijske arhitekture

Za vse, ki se v profitnem ali neprofitnem delovnem okolju soočijo z zahtevo po zasnovi (prenovi) informacijske arhitekture, se med temeljne pripomočke uvršča preučevanje uveljavljenih arhitekturnih vzorcev načrtovanja računalniških rešitev. Preučevanje razlik med modeli in predstavitev lastne arhitekturne zasnove, ki naj bo predstavljena tako z vidika sestavnih delov (komponent) kot pogledov vseh sodelujočih razvijalcev, sodi med dejavnike, pomembne za uspešno dokončanje razvojnega projekta.

V splošnem se na nivoju delovanja organizacij, ki jih uvrščamo v neprofitni sektor, vedno pogosteje srečujemo z zahtevami po boljši dostopnosti do podatkov ali njihovih storitev. Svetovni splet je idealno okolje za distribucijo takšne ponudbe. To je potrebno upoštevati tudi pri zasnovi arhitekture programskih rešitev. Informacijsko komunikacijska tehnologija omogoča zasnovo različnih arhitektur in njihove kombinacije. Predmetna arhitektura izvorno sloni na zasnovi predmetov, razredov, dedovanju in ograjevanju. Kot takšna sloni na podrobnem (drobnozrnatem) modelu medsebojnih interakcij objektov. Brez uporabe sistematičnih delovnih napotkov ene od razvojnih metodologij v okviru razvojnega projekta, je pri takšni arhitekturi uspešnost končne računalniške rešitve vprašljiva.

V pomanjkanju razvojnega kadra se zdi bolj racionalen razmislek o snovanju arhitekture na višji abstraktni ravni kot je čista objektna (drobnozrnata). V okolju s skromnimi razvojnimi resursi je bolj priporočljivo snovati arhitekturo na uporabi komponent z zaključeno (in porazdeljeno) funkcionalnostjo. Seveda se je v takšnem primeru potrebno bolj posvetiti preučevanju vzajemnega delovanja in možnostim medsebojnega povezovanja pod okriljem enotnih tehnoloških standardov. Danes so različne komponente na voljo tako v obliki plačljivih kot odprtokodnih rešitev, ki jih lahko tudi brez obsežnega razvojnega vložka uporabimo v lastnih rešitvah. V tistih okoljih, kjer se dejavnost neprofitne organizacije odraža v zagotavljanju različnih podatkov ali povezanih storitvah, ki bi jih lahko uvrstili kar v kategorijo javne infrastrukture, je vsekakor priporočljiv resen razmislek o uvajanju storitev, ki temeljijo na enem od uveljavljenih arhitekturnih vzorcev za podporo storitvene arhitekture.

Med osnovne korake v načrtovanju posodobitve informacijske podpore sodi vzpostavitev oziroma revizija obstoječe informacijske arhitekture. V ZGS je del temeljev informacijske arhitekture začrtan z organiziranostjo in s povezanostjo vseh organizacijskih enot v enotno računalniško mrežo. Vzpostavljena informacijsko komunikacijska infrastruktura omogoča vse sodobne arhitekture porazdeljenega delovanja računalniških rešitev, značilnih za današnje „spletno“ (porazdeljeno) informacijsko okolje. Informacijsko arhitekturo lahko izvajamo na čisto objektni zasnovi ali pa na višji stopnji abstrakcije s pomočjo komponentne in storitvene arhitekture.

Z vidika dosežene stopnje razvoja nekaterih lastnih računalniških rešitev in v okviru danih pogojev izvedbe informacijske podpore, je za ZGS najbolj primerno snovati komponentno in storitveno arhitekturo programskih rešitev. Na tej osnovi se lahko izognemo zahtevam po izdelavi nekaterih sestavin, ki so že na voljo v splošni obliki oziroma funkcionalnosti. Za ZGS imajo, gledano s stališča zahtev informacijske podpore nekaterim procesom, na zasnovo informacijske arhitekture največji vpliv model centralno hranjenih podatkov, zahteva po delovanju nekaterih aplikacij na osnovi specifičnega uporabniškega posredovanja ter zahteva po uporabi in izmenjavi nekaterih podatkov z drugimi organizacijami. Dolgoročno razvojno gledano bi bilo koristno v ZGS informacijsko arhitekturo snovati predvsem na konceptu vzajemno delujočih komponent s specifičnimi (kompaktnimi) nabori funkcionalnosti. V ta koncept sodi izvedba enega od naslednjih posamičnih ali, še bolje, kombiniranih arhitekturnih vzorcev:

- trislojna arhitektura odjemalec-strežnik,
- vzorec klica oddaljene procedure,
- vzorec posrednikov,
- vzorec skladiščenja pozivov.

Komponentna arhitektura omogoča vzajemno integracijo delovanja tako delujočih rešitev kot tudi prihodnjih programskih komponent, namenjenih informacijski podpori zaključenih postopkov ali storitev, ki jih izvaja ZGS (arhitektura odjemalec-strežnik in vzorec klica oddaljene procedure). Glede na smernice razvoja programskih rešitev, ki delujejo v porazdeljenem računalniškem okolju, je za ZGS primerna tudi storitvena informacijska arhitektura (vzorec posrednikov). Dostopanje do storitev ZGS preko objavljenih vmesnikov namreč omogoča zagotavljanje različnih storitev, med katere lahko prištevamo tako tiste, ki so namenjene notranjemu delovanju kot tiste, ki so namenjene zainteresirani javnosti. Storitve omogočajo odzivanje na zahteve iz različnih virov. Funkcionalnost storitve pa lahko spreminjamo neodvisno od njihovih naročnikov, kar lahko, poleg razširljivosti in prožnih možnosti vzdrževanja, štejemo med prednosti storitvene arhitekture.

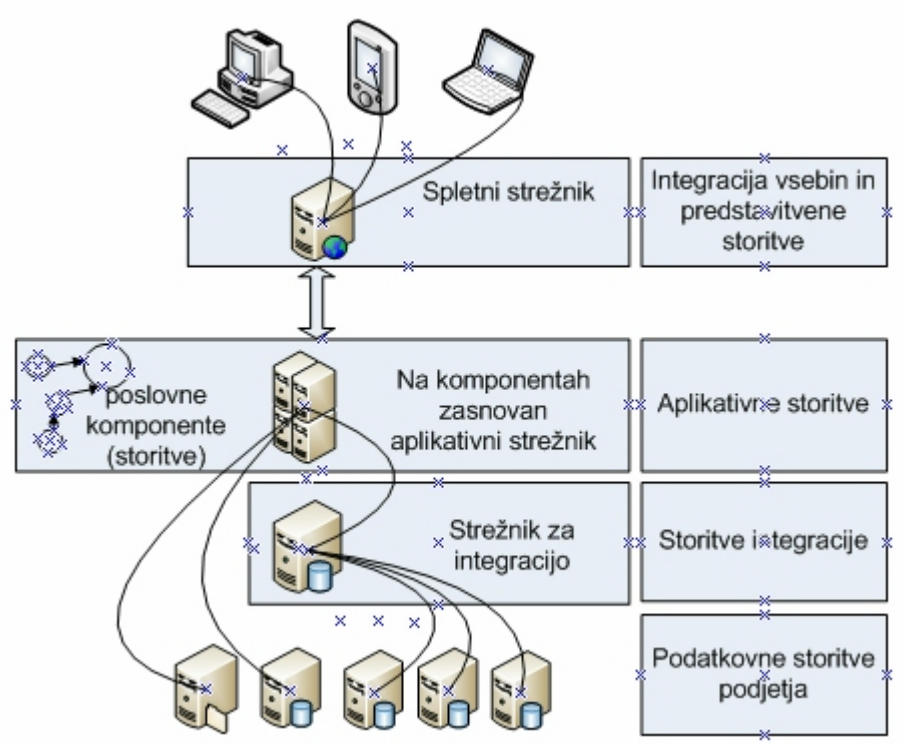
Strateško gledano je za ZGS najprimernejše kombiniranje komponentno storitvene arhitekture. Kombinacija obeh omogoča zadovoljevanje potreb v trenutnih pogojih delovanja in ima velik razvojni potencial. Ena od trenutno aktualnih storitev iz koncepta storitvene arhitekture je, na primer, spletna kartna storitev (angleško: web map service). Omogoča poizvedbe po predstavitvi različnih tematskih kart, ki jih izdeluje ZGS, pa jih zainteresiranim trenutno še ne ponuja v obliki spletne storitve. Z uvajanjem komponentne in storitvene arhitekture lahko večinoma dvoslojno zasnovo delovanja aplikacij v ZGS nadgradimo s trislojnimi programskimi rešitvami (delujočih na predstavitvenem, aplikacijskem in podatkovnem sloju). Trislojna informacijska arhitektura deluje na vmesnih programskih rešitvah (angleško: middelware), ki zagotavljajo aplikaciji ali komponentam aplikacije povezovanje in izmenjavo podatkov z drugimi aplikacijami (Zhang, Jacobsen, 2003, str. 1058-1060). Na zasnovo informacijske arhitekture se navezuje uporaba izvedbene tehnologije in povezanih standardov. V tej zvezi se bo na izvedbenem nivoju (podpora izobraževanju) potrebno opredeliti do platforme vmesnih programskih rešitev (spletni servisi, J2EE, CORBA, .NET), ki jo bo ZGS v prihodnje prednostno favoriziral pri izdelavi porazdeljenih računalniških rešitev.

Prednosti večslojnih programskih rešitev se kažejo predvsem v možnostih lažjega vzdrževanja in nadomeščanja posameznih komponent v slojih, neodvisno od novih zahtev ter tudi od sprememb

tehnologije. Trislojna arhitektura omogoča tudi nadaljnjo optimizacijo arhitekture z nadomeščanjem aplikacijskega sloja z več sloji, kar v končnem rezultira v n-slojni arhitekturi programskih rešitev, ki so še bolj prilagojene na delovanje aplikacij v okolju, ki ga zagotavlja informacijsko komunikacijska tehnologija.

ZGS je vezan na uporabo prostorskih podatkov. Referenčni model za njihovo zbiranje in uporabo predstavljajo vektorski in rastrski podatki, s katerimi predstavljamo izbrane lastnosti prostora. Centralno hranjenje in vzdrževanje teh podatkov, dostop vseh organizacijskih enot ZGS do teh podatkov ter skrb za združljivost s podatki drugih organizacij, ki vzdržujejo prostorske podatke, ostaja med temeljnimi predpostavkami v informacijski arhitekturi ZGS. Ena od zahtev za delovanje javne gozdarske službe v izvedbi ZGS je souporaba prostorskih podatkov drugih lastnikov. V tem pogledu spodaj predlagana informacijska arhitektura omogoča tudi tovrstno izkoriščanje spletnih storitev drugih ponudnikov.

Slika 11: Nosilci in odjemalci komponentno zasnovane informacijske arhitekture v porazdeljenem delovanju računalniških rešitev




Ne nazadnje je potrebno upoštevati, da na ravni procesiranja prostorskih operacij sodobne geopodatkovne baze vsebujejo čedalje večjo vgrajeno funkcionalnost, ki jo lahko izrabimo že na samih podatkovnih strežnikih. Centralno hranjeni podatki in v geopodatkovno bazo vgrajena funkcionalnost operacij nad entitetami zmanjšuje zahteve po lokalno zmogljivih aplikacijah ali težkih odjemalcih (angleško: fat client). Slednji čedalje bolj postajajo pregledovalniki podatkov na nivoju predstavitve rezultatov analiz in poizvedb, predhodno izvedenih na podatkovnih in aplikacijskih strežnikih.

### 5.7.3 Prehod na predmetno zasnovane računalniške rešitve

Koncept predstavitve delovanja nekega sistema na osnovi bolj ali manj podrobno opredeljenih vzajemno delujočih entitet (z lastnostmi in funkcionalnostjo) je ena največjih odlik predmetno zasnovane paradigme razvoja računalniških rešitev. Uporaba predmetnih programirnih jezikov za izvedbo tako zasnovanih rešitev je nadgradnja tega koncepta. Vendar pa zgolj uporaba slednjih ne daje jamstva kakovostnih predmetnih računalniških rešitev.

V neprofitnem delovnem okolju lahko uvajanje predmetne informacijske tehnologije, tako kot v profitnem, obravnavamo bodisi kot priložnost za temeljno vzpostavitev informacijske podpore, bodisi kot priložnost za njeno nadgradnjo. V obeh primerih za razvoj predmetno zasnovanih računalniških rešitev potrebujemo predstavitev predmetnega modela entitet in povezav, ki nastopajo v domeni delovanja organizacije. Na tej osnovi lahko združujemo dele funkcionalnosti v izbrane module in komponente. Če ugotovimo, da vzajemno delovanje osnovnih predmetov iz domene delovanja lahko predstavimo na višji abstraktni ravni delovanja, za katero že obstaja modul ali programska komponenta podobne funkcionalnosti, lahko takšne delne rešitve s pridom uporabimo v projektu svojih končnih računalniških rešitev.

Temeljna značilnost razvoja predmetnih računalniških rešitev je njihova zasnova in izvedba na ravni elementarnih gradnikov predmetne informacijske arhitekture: predmetov, razredov in njihovih razmerij. Ob razvoju s pomočjo osnovnih gradnikov je potrebno oceniti pogoje izvedljivosti končnih računalniških rešitev. Dosledno upoštevanje paradigme objektnega razvoja računalniških rešitev, kjer izvedba sloni na implementaciji elementarnega objektnega modela iz domene delovanja organizacije, predstavlja težjo razvojno pot. Sloni na izvedbi razvojnih projektov in delovanju razvojnih skupin kot ga predpisujejo metodologije razvoja predmetnih računalniških rešitev (Wang, Fung, 2004). Zahteva načrtovanje projekta, izkušene kadrovske vire s številnih delovnih področij, dobro poznavanje ene od metodologij razvoja računalniških rešitev in dobre mehanizme izvajanja skupinskega dela. Takšen pristop je bolj kot ne pisan na kožo organizacijam, ki imajo dovolj kadrov za izvajanje številnih vlog, udeleženih v projektu razvoja računalniške rešitve, ali specializiranim podjetjem za razvoj računalniških rešitev.

Neprofitna delovna okolja so običajno bolj ali manj podhranjena s strokovnimi kadri informatike. Ta primanjkljaj v veliki meri ovira kompleksen razvoj računalniške rešitve, kjer izvedba temelji na osnovnih gradnikih objektnega modela v domeni delovanja. To pa ne pomeni, da slednjega ni potrebno modelirati. Potrebno se je le zavedati dejstva, da drobnozrnata zasnova objektov, razredov in razmerij zahteva tudi njihovo izvedbo. Slednja pa zahteva kadre, čas, testiranje, vračanje v predhodne razvojne faze, popravljanje in restrukturiranje. Več kot je objektov in povezav med njimi, več virov potrebujemo. V takšnih pogojih se zdi bolj racionalna odločitev iskanje razvojnih možnosti na drugi ravni uporabe predmetne tehnologije. V fazi izvedbe ne tvorimo več vseh elementarnih razredov predmetov, pač pa se razgledamo po možnosti uporabe že narejenih komponent ali programskih modulov, ki vsebujejo dele želene funkcionalnosti. Na ta način lahko izvedemo prilagojen prehod na izdelavo predmetnih računalniških rešitev. Izdelamo sicer predmetni model iz domene delovanja, vendar njegova izvedba ne sloni na lastni izdelavi vseh sestavnih delov, pač pa na njihovem delnem nadomeščanju z že izdelanimi komponentami ali moduli.

Tudi za ZGS zgolj nadomeščanje uporabe sedanjih programskih jezikov z enim od popularnih predmetnih jezikov ne jamči uspešnega prehoda na predmetno zasnovane računalniške rešitve. Izhodišče za izdelavo slednjih bi moralo sloneti na razvoju predmetnega modela iz domene delovanja ZGS oziroma delovanja javne gozdarske službe. Na osnovi takšnega predmetnega

modela lahko izvajamo abstrakcijo v obliki posameznih komponent, hkrati pa skrbimo za integracijo nekaterih rešitev, ki jih še ni potrebno prenavljati.

Na osnovi ocene stanja in preučevanja razvojnih metodoloških pristopov bi med pomembnejše dejavnike za prehod na predmetno zasnovane računalniške rešitve v ZGS uvrstil naslednje:

- predstavitev informacijske podpore s pomočjo modela entitet in njihovih relacij, ki nastopajo v domeni delovanja ZGS (ali javne gozdarske službe),
- predstavitev informacijske podpore v obliki možnih komponent sistema, ki bi združevale posamezne skupine predmetov in omogočale pogled na celoten sistem v obliki podsistemov,
- uvajanje uporabe modelirnega jezika za predstavitev predmetnega modela, vzajemnega delovanja in komponent sistema,
- uporabo pripomočkov pri modeliranju objektov in njihovih medsebojnih razmerij v obliki primerov vzorcev zahtev ter uporabo načrtovalskih vzorcev uspešnih programskih rešitev,
- uvedbo (učenje) enega od predmetno orientiranih programskih jezikov, ki omogoča izdelavo predmetno zasnovanih računalniških rešitev ter deklarira povezovanje komponent iz različnih virov v enoten sistem na osnovi odprtih standardov.

Za ZGS imajo vložki zgolj v učenje uporabe enega od predmetnih jezikov z dobro razširjeno podporo (na primer Java, .Net) veliko možnosti, da se izkažejo kot preozek okvir za uspešnejše računalniške rešitve od dosedanjih. Med ključne dejavnike, na osnovi katerih bi lahko pospešili razvoj predmetnih računalniških rešitev v ZGS, je poleg poznavanja uporabe predmetnega programirnega jezika potrebno uvrstiti še: poznavanje in uporabo enega od modelirnih jezikov, načrtovanje rešitev na osnovi objektnega modela iz domene delovanja, poznavanje različnih vzorcev uspešnih programskih rešitev ter poznavanje standardov za komunikacijo med komponentami oziroma delnimi računalniškimi rešitvam v okviru uporabljenega programirnega jezika.

Dokaze za zgoraj navedene usmeritve lahko najdemo v množici razvojnih metodologij, ki so nastajale in se razvijale od pojava predmetne informacijske tehnologije naprej. Preučevanje razvojnih metodologij nam v največji meri pokaže na potrebna povezana znanja in pripomočke, ki so poleg poznavanja objektnega programskega jezika, potrebni posamezniku ali razvojni skupini pri razvoju predmetnih računalniških rešitev.

Prehod na predmetno zasnovane računalniške rešitve zahteva obsežnejši nabor potrebnega „tehničnega“ informacijskega znanja kot se zdi na prvi pogled. Nabor potrebnega znanja je za razvojno skupino lahko izziv, kako znanje s skupnimi močmi osvojiti, lahko pa je nepremostljiva ovira uspešnemu razvojnemu delu. Načina za pridobitev tega znanja sta vsaj dva. Prvi je opredelitev vlog in vmesnikov (pripomočkov) sodelovanja med udeleženci razvojne skupine ter usmerjanje v pridobivanje specialnega znanja, skladno z dodeljenimi vlogami. Druga možnost je usmerjanje vseh udeležencev razvoja v pridobivanje širokega razvojnega znanja (modeliranje, načrtovanje, programiranje). Takšen model pridobivanja znanja bo verjetno bolje sprejet v okolju, kjer projektno delo nima posebej uspešne tradicije in kjer je razvojno delo bolj ali manj prepuščeno posameznikom, ki svojega znanja niso pripravljeni prav velikodušno medsebojno izmenjevati. Ima pa lahko določeno prednost pred prvim načinom, saj tu ne prihaja do izrazitega primanjkljaja specialistov.

### 5.7.4 Predmetni model dejavnosti

Modeliranje vzajemnega delovanja entitet s področja delovanja poslovnega subjekta sodi med pripomočke pri načrtovanju predmetno zasnovanih računalniških rešitev. Na ravni izvedbe poslovnih procesov se nekateri procesi lahko izvajajo na standardizirani podatkovni (informacijski) osnovi. Lahko se izvajajo v obliki storitve, ki je javno objavljena. Takšne storitve omogočajo načrtovanje in povezovanje računalniških rešitev, izdelanih z različnimi tehnologijami. Podlaga za izdelavo oziroma ponudbo storitve so lahko referenčni modeli vzajemnega delovanja standardnih entitet, ki nastopajo v domeni neke dejavnosti (na primer izdelava tematskih kart). Na osnovi predmetnega modela entitet v izbrani dejavnosti lahko oblikujemo vrsto storitev, ki bodo dostopne vsakokratnim izvajalcem izbranih opravil s področja dejavnosti. Za ilustracijo lahko navedemo primer spletne kartografske storitve (WMS), s pomočjo katere vsakokratnemu odjemalcu prikažemo karto na osnovi dinamične poizvedbe nad prostorskimi podatki.

Tako za profitne kot neprofitne organizacija velja, da so objavljeni referenčni ali lastni predmetni modeli vzajemnega delovanja entitet, značilnih za dejavnost, v kateri izvajajo svoje storitve, nepogrešljiv pripomoček tako za snovanje storitev po modelu storitvene arhitekture kot tudi za izdelovanje lastnih računalniških rešitev, ki takšne storitve uporablja.

Ali je ideja o splošnem predmetnem modelu delovanja ZGS ali širše predstavitve dejavnosti javne gozdarske službe vredna razmisleka ali ne, bi se moralo pokazati v soočenju naravovarstvenega, gozdarskega, gospodarskega in še kakšnega pogleda na razvoj mehanizmov za izvajanje politike sonaravnega gospodarjenja v gozdovih. Takšen pogled lahko presega domet enega samega izvajalca v dejavnosti, pa tudi meje ene same regije ali države.

Gledano s perspektive globalnih naravnih sprememb in tudi s stališča političnega združevanja Evrope lahko ugotovimo, da uvajanje enotnih razvojnih politik na različnih področjih našega delovanja zaznamuje naš jutri. Javna gozdarska služba na regionalni ravni pri tem ne bo izvzeta. Ali je predmetni model delovanja javne gozdarske službe ene regije pri tem priložnost za stroko in za promocijo dokazano dobrih vzorcev načrtovanja in gospodarjenja v gozdu, bodo morali odgovoriti drugi. Eden od možnih načinov predstavitve je zanesljivo snovanje predmetnega modela entitet v delovanju javne gozdarske službe in njihove vzajemne povezanosti.

Četudi takšen model ne bi imel ambicije za širšo (medregijsko) uveljavitev, ga lahko obravnavamo kot koristen referenčni pripomoček za načrtovanje informacijske podpore ali njene posodobitve za vse (in vsakokratne) izvajalce javne gozdarske službe. Lahko ga obravnavamo tudi kot referenčni model, s potencialom združevanja izobraževalnih, raziskovalnih in morda celo organizacij v opravljanju nalog javne gozdarske službe.

### 5.7.5 Metodologija razvoja računalniških rešitev

Preučevanje različnih razvojnih metodologij kaže na njihovo neprestano spreminjanje, izboljševanje in preoblikovanje. Pri razvoju informacijske podpore z izdelavo lastnih predmetnih računalniških rešitev se razvijalci učijo na preteklih izkušnjah in iščejo svojim razmeram ter možnostim prilagojen način dela. Takšen razvojni pristop je potrebno upoštevati tudi v neprofitnem delovnem okolju.

Delovni postopki in pripomočki, ki jih bomo pri razvoju uporabili, naj slonijo na usmeritvah preizkušenih metodoloških pristopov. Iz mejnih pogojev, ki jih opredeljuje vsaka metodologija, lahko veliko razberemo o njeni primernosti za uporabo v specifičnih delovnih pogojih. Navodila izvajanja metodologije je potrebno presojati v luči naših izkušenj. Če vsebuje metodologija vrsto napotkov, ki so nam po svojem konceptu tuji, se rajši ozrimo po alternativni možnosti.

Izkušnje profitnih organizacij kažejo, da togo izvajanje določene metodologije razvoja računalniške rešitve ni predpogoj za uspešno delo. Grožnje pri uvajanju metodoloških podlag v razvoj računalniških rešitev so pravzaprav drugje. Ključni dejavniki uspeha so organizacija, vodenje in spremljanje razvojnega procesa. Upoštevanje strogih vseobsegajočih kaskadnih metod, ki podrobno predpisujejo množico opravil, predstavlja zahteven izvedbeni zalogaj tudi za dobro organizirana in kadrovsko usposobljena delovna okolja. Nepriljubljenost togih metodoloških okvirov, izkušnje in želje programerjev po boljši odzivnosti potrebam naročnikov oziroma uporabnikov so pripeljale do številnih modifikacij in metodoloških izboljšav. Sem sodi tudi popularizacija uporabe procesnih vzorcev razvoja in uporaba procesnih ogrodij. Vedno pogostejša sestavina razvojnih procesov postaja izdelava prototipov rešitev. Ti so postali nekakšen zaščitni znak novih agilnih razvojnih metodologij.

Na osnovi spoznanj iz preučevanja primerov metodologij se mi zdi potrebno poudariti, da bi morala odločitev o uvedbi metodologije sloneti prvič na njenem dobrem poznavanju in drugič na oceni njene izvedljivosti v konkretnem delovnem okolju. Ni nam treba biti žal časa za študijo njene izvedljivosti. Izvedljivost je potrebno v neprofitnem delovnem okolju navezovati predvsem na usposobljenost, razpoložljive kadrovske vire in možne oblike njihovega skupinskega dela. Uporaba orodij za skupinsko delo je pri tem skoraj obvezna. Če se izkaže, da razpolagamo pretežno s kadri s programerskim znanjem, potem bomo verjetno iskali smernice za razvoj računalniških rešitev pri metodologijah, ki slonijo na izdelavi prototipov. Če želimo ustvariti dokumentacijo, ki bo predstavljala predlogo, po kateri bo možno naročiti izdelavo rešitve pri zunanjih izvajalcih programiranja, potem se bo bolj koristno zgledovati po smernicah bolj „strogih“, v dokumentiranje in podrobnejše načrtovanje usmerjenih metodologij.

Pri odločanju o uporabi ene ali druge metodologije je koristno že na začetku razčistiti z dilemo o formalni komunikaciji med izvajalcem in naročnikom (uporabnikom). Naročnik končne rešitve je tisti, ki jo bo prevzel v svoje delo. Če izdelujemo rešitev za uporabnika znotraj lastne organizacije, mu zamisel končne rešitve lahko kažemo tudi v zelo konkretni obliki prototipa aplikacije. Lahko pa mu isto rešitev prikazujemo v bolj abstraktni obliki modelov entitet končne rešitve. V prvem primeru uporabnik vidi bolj kot ne samo končni vmesnik za rokovanje z aplikacijo, v drugem pa različne poglede na sestavine in mehanizem vzajemnega delovanja entitet aplikacije. Prav to pa je lahko presežek vloženega dela v modeliranje entitet sistema. Informacijski inženiring mu skozi različne metodologije pripisuje veliko uporabno vrednost od področja ponovne uporabe komponent, možnosti združevanja z drugimi podsistemi in nadgrajevanja, do pisanja uporabniških navodil. Našteto ima lahko dodatno težo za neprofitne organizacije pri zbiranju in ustvarjanju novega informacijskega in organizacijskega znanja.

Uvajanje metodologije v razvoj lastnih računalniških rešitev je zagotovo izziv za funkcijo vodenja razvojnih projektov. Povezovanje naročnika in izvajalca ter uvajanje novih metodoloških konceptov ali komunikacijskih sredstev bo sprva zagotovo naletelo na zadrege, povezane s pomanjkanjem začetnih izkušenj. Pojavile se bodo zahteve po dodatnem izobraževanju. Iz nekaterih praktičnih izkušenj uporabe metodologij izhajajo ugotovitve, da je včasih vloga razširjanja in skrb za poenotenje znanja s strani skrbnika projekta lahko bolj odločilna za uspeh projekta od zgolj nadzorne vloge vodenja (Bruegge Dutoit, 2004). To je

morda še zlasti pomembno za okolja, kjer imajo zaposleni odklonilen ali ambivalenten odnos do samoizobraževanja, ali pa slednjega občutijo kot prevzemanje nepotrebnih dodatnih obveznosti.

V ZGS bi lahko utemeljili primanjkljaj razvojno-metodološkega in tehničnega znanja ter omejeno število razvojnih kadrovskih virov za lastno izdelavo računalniških rešitev. Dosledno vpeljavo in uporabo ene od poznanih metodologij razvoja je v takšnih pogojih skoraj nemogoče zagotoviti oziroma jo je potrebno prilagoditi razvojnim možnostim. V takšnih pogojih se zdi racionalna odločitev tudi zasnova lastne metodologije.

V pogojih trenutnega delovanja in želenega prihodnjega razvoja informacijske podpore je za ZGS smiselno poiskati oziroma izdelati metodologijo razvoja, s katero bi lahko dosegli naslednje cilje:

- omogočili lasten razvoj predmetno zasnovanih aplikacij v obliki celovitih ali delnih rešitev za podporo izvedbe poverjenih nalog javne gozdarske službe,
- omogočili integracijo novih rešitev z obstoječimi,
- omogočili dokumentiranje zahtev uporabnikov in njihovo modeliranje v kontekstu celotnega informacijskega sistema,
- omogočili izdelavo dokumentacije za naročanje izvedbe delnih rešitev pri zunanjih izvajalcih,
- omogočili izdelavo dokumentacije za dolgoročno lastno vzdrževanje računalniških rešitev in podporo končnim uporabnikom.

Ob predpostavki, da ZGS potrebuje metodologijo za razvoj in izdelavo lastnih predmetno zasnovanih računalniških rešitev, predlagam izdelavo lastne metodologije. Na osnovi metamodela procesnega ogrodja OPEN za izvajanje metodologije predlagam naslednje metodološke komponente:

- podprocese razvoja, znotraj katerih izdelamo posamezne pripomočke ter izdelke kot del celotnega nabora izdelkov razvojnega procesa,
- zaključen seznam izdelkov, ki jih izdelujemo v podprocesih razvoja in predstavljajo pripomočke za doseganje zastavljenih ciljev,
- formiranje vlog izvajalcev za izvedbo metodologije in za oblikovanje programa izobraževanja, kjer se posamezen izvajalec lahko specializira za izvedbo več vlog.

Na osnovi predlaganih komponent metodologije predlagam izdelavo naslednjih sestavin komponent:

1. razvojni podprocesi:

    - okvirni opis problema (sistema), predlog rešitve, analiza izvedljivosti,
    - izločanje zahtev sistema, analiza, modeliranje domene in primerov uporabe,
    - specifikacija rešitve sistema, izdelava modelov računalniške rešitve ob uporabi pripomočkov (načrtovalski vzorci računalniških rešitev),
    - predstavitev izvedbene arhitekture za sistem, ki smo ga predhodno modelirali,
    - načrtovanje izvedbe projekta po delovnih področjih in zadolžitvah izvajalcev oziroma izdelava razpisne dokumentacije za zunanjo izvedbo sistema,
    - izdelava funkcionalnih paketov končne rešitve (deli),
    - preverjanje delovanja delov rešitve in vgradnja v obstoječi sistem (predaja v uporabniško okolje).

2. delovni proizvodi in pripomočki, ki jih izdelamo v podprocesih:

   - Izvedbeno analitični dokument z orisom vsebine celotnega projekta in oceno izvedljivosti.
   - Vsebinski model sistema v kontekstu delovanja ZGS, s katerim predstavimo elemente (entitete), za katere načrtujemo (delno) rešitev. Izdelamo (zberemo) scenarije (opise) izvedbe opravil (procesov), za katere načrtujemo rešitev. Izdelamo model primerov uporabe. S pomočjo primerov uporabe predstavimo uporabniške scenarije in funkcije sistema. Predstavimo akterje (vloge) uporabe sistema in razmerja med primeri uporabe. Izdelamo diagrame interakcije. Predstavimo nefunkcijske zahteve in morebitne omejitve sistema.
   - Na osnovi pretvorbe vsebinskega modela izdelamo predmetni model delovanja sistema. Predstavimo izvedbene podrobnosti, izdelamo diagrame stanja, dogodkov, predstavimo interakcijo objektov in sporočil, izdelamo diagrame sodelovanja, zaporedja pozivov, predstavimo vmesnike, predvidene kontrolne mehanizme.
   - Izdelamo izvedbeni model računalniške rešitve, ki predstavi sestavne dele (komponente) končne programske rešitve in njihovo nahajališče na sistemskih virih (informacijska arhitektura). Izdelamo diagram komponent in njihove razporeditve.
   - Izdelamo projektni načrt in načrtovano dinamiko lastne izdelave (programiranje) računalniške rešitve oziroma pripravimo razpisno dokumentacijo za zunanjo izvedbo rešitve.
   - V primeru lastne izvedbe rešitve skupine povezane funkcionalnosti sistema razdelimo na delovne pakete (delne rešitve), za katere predhodno izdelamo prototipe rešitve in načrt izvedbe. Z vlogami programerjev jih realiziramo na osnovi načrtov izvedbe.
   - Izdelamo poročila testiranja in preverjanja izvedenih delovnih paketov.
   - Izdelamo uporabniški vodič in izvedbeni priročnik za izvajanje delne rešitve (v celotnem sistemu).

3. vloge izvajalcev, s pomočjo katerih oblikujemo skupine za izvedbo podprocesov:

   - projektni vodja, ki vodi celoten projekt v skladu z načeli izvajanja projektov;
   - predstavnik uporabnika, ki sprejema odločitve v imenu uporabnika in skrbi za povratne informacije uporabnikov, če ti sodelujejo pri razvoju;
   - strokovnjak iz gozdarske domene, ki sprejema odločitve, povezane s strokovno neoporečnostjo izvedbe opravil, za katere izvajamo informacijsko podporo;
   - izvajalec modeliranja, ki s pomočjo enega od jezikov modeliranja izvaja predmetno modeliranje delovanja sistema;
   - skrbnik podatkov, ki skrbi za opise podatkov in skladnost šifrantov. Zagotavlja časovno primerljivost zbranih podatkov, sodeluje pri oblikovanju predmetov v domeni delovanja sistema;
   - poznavalec načrtovalskih vzorcev in tehnoloških standardov, ki nudi pomoč pri uporabi načrtovalskih vzorcev in pri uporabi izbrane tehnološke platforme;
   - poznavalec komponent, standardov njihove medsebojne integracije, izdelovalec prototipov delnih rešitev;
   - programer, ki programira del celotne rešitve, delovni paket (predstavljen s prototipom rešitve in načrtom izvedbe);
   - vodja programerjev, ki usmerja in odloča o podrobnih načrtovalskih in izvedbenih aktivnostih med iterativno-inkrementalnim razvojem delovnih paketov;
   - izvajalec testiranja, ki preizkuša delne rešitve in njihovo delovanje v celotnem sistemu ter izdeluje poročila testiranja;

- urednik dokumentacije, ki skrbi za popolnost, vsebinsko skladnost ter skladiščenje pripomočkov in nastale dokumentacije.

Izvedba procesa oziroma podprocesov na osnovi predlagane metodologije naj bi potekala po naslednjem scenariju:

- Projektni vodja s pomočjo sodelavcev izdela okvirni opis problema (sistema), predlog rešitve, predlaga seznam podprocesov, končnih izdelkov ter vlog v predvidenem projektu razvoja računalniške rešitve. Poda oceno izvedljivosti.
- Na osnovi odločitve o izvedbi projekta projektni vodja skupaj z izvajalci, ki so potencialni nosilci vloge odgovornega izvajalca podprocesov, izdela dokončen seznam izdelkov (pripomočkov), ki bodo nastali v izvedbi projekta. Projektni vodja izdela seznam vlog, potrebnih za izdelavo posameznega izdelka, in opredeli podprocese, znotraj katerih je predvidena zaključena izdelava predvidenih izdelkov. Izdelki podprocesa naj bi praviloma predstavljali zaključeno celoto predstavitve enega od pogledov, ki se nanaša na ves sistem (na primer model primerov uporabe, model delovanja objektov sistema).
- Projektni vodja predstavi celoten projekt izvajalcem (vsem potencialnim nosilcem vlog, opredeljenih s projektom). Na osnovi seznama podprocesov in predvidenih vlog za izvedbo podprocesa sodelujoči izvajalci na osnovi vlog, ki jih lahko prispevajo, imenujejo izvajalce (v obliki delovne skupine) za vsak podproces (skupino izdelkov) posebej. Odvisno od obsega dela lahko za izvedbo podprocesa predvidimo vključitev več enakih vlog. Posamezen izvajalec je lahko nosilec več vlog. Lahko je tudi imenovan (delegiran) v več delovnih skupin projekta.
- Za vsako od delovnih skupin imenujemo odgovornega izvajalca. Odgovorni izvajalec odgovarja za svoje delo neposredno vodji projekta in je zadolžen za organizacijo in izvedbo izdelave (predajo) izdelkov, predvidenih v izvedbi podprocesa. Ima pooblastila odločanja znotraj izvedbe podprocesa. Odgovorni izvajalec je v delovni skupini odgovoren za koordinacijo dela in vlog ostalih udeležencev v skupini. Ob morebitnih dilemah ali nesoglasjih v skupini posreduje vodja projekta. Imenovanje odgovornega izvajalca lahko izhaja iz želje skupine ali iz opredelitve projektnega vodje. Po možnosti naj bo odgovorni izvajalec nosilec več vlog.
- V hierarhiji nosilcev izvedbe projekta so ostali udeleženci delovne skupine, mimo odgovornega izvajalca, obravnavani kot izvajalci. S svojim delom prispevajo k izdelavi izdelkov (ali njihovih delov), predvidenih v izvedbi podprocesa. Vsak od izvajalcev izvaja delo v okviru jasne, vnaprej opredeljene (poimenovane) vloge. Izvajalec je lahko usposobljen za izvedbo več vlog. V dogovoru in ob sodelovanju z odgovornim izvajalcem načrtuje delo in sprejme v delo načrtovane zadolžitve v okviru izvedbe podprocesa. Odgovorni izvajalec spremlja izvedbo dela izvajalca in mu pri izvedbi zadolžitev po potrebi pomaga. Izvajalec izraža zahtevo po pomoči odgovornega izvajalca z vednostjo vodje projekta. Odgovornost izvajalca za opravljeno delo izhaja iz domene sprejete vloge ter sprejetih lastnih odločitev pri uporabi svojega znanja v izvajanju sprejetih zadolžitev.
- V okviru načrtovanja dela v podprocesu izvajalec in odgovorni izvajalec vsako od zadolžitev poimenujeta in jo po potrebi opišeta. Zadolžitve služijo kot merilo oziroma pripomoček za spremljanje izvedbe dela in osnova za poročanje o poteku dela vodji projekta. Praviloma naj bi se zadolžitve nanašale na izdelavo posameznega pripomočka (poročilo, opis postopka, diagram, scenarij, primer uporabe, prototip, delni načrt, del programske kode ipd.). Zadolžitve so sestavine izdelka oziroma izdelkov), ki jih izdelujemo v podprocesu.
- Kot jezik za modeliranje se uporablja poenoteni jezik modeliranja (UML). Kot sredstvo za komunikacijo se uporabljajo pripomočki in diagrami jezika UML. Uporabimo UML

razrede modelov (modeli razredov), aktivnostne diagrame (interakcijski modeli), diagrame zaporedja (interakcijski modeli), komponentne diagrame (modeliranje podsistemov v arhitekturi modelov razredov in modelov objektov).

Za predlagani način razvoja je ključna opredelitev izdelkov in vlog. Vodja projekta z izdelki opredeli inkrementalnost (korake oziroma podprocese) razvojnega projekta. To mu nalaga odgovornost za večplastno predstavitev sistema (pogledi na sistem). Odgovorni izvajalci podprocesov skupaj z izvajalci poskrbijo za iterativnost izvedbe vsakega podprocesa. To jim nalaga odgovornost za izvedbo dela po načelu sistemskega inženiringa, po katerem vsaka izdelava izdelka poteka v razvojnem ciklu analize, načrtovanja, izvedbe in preverjanju njegove končne ustreznosti. Predlagani način dela je lahko koristen za organizacijo izvedbe projekta, omogoča decentralizacijo odločanja v izvedbi projekta in porazdelitev odgovornosti med odgovorne izvajalce. Na osnovi dosedanjih izkušenj lahko omenjene dejavnike ocenimo kot šibkejši člen v organizaciji dela ZGS.

Vloge opredeli vodja projekta. Kot pripomoček lahko uporabi povzetek vlog, pridobljen na osnovi preučevanja drugih metodologij razvoja. Vsekakor je bolje, če je opredeljena kakšna vloga več kot manj. Vodilo pri opredelitvi vlog naj bodo načrtovani izdelki (pogledi) na končno računalniško rešitev. Z vlogo opredelimo znanje, izkušnje in delovne veščine, potrebne za izdelavo izdelka. Ob predpostavki, da se razvoj sistema izvaja v obliki projekta, ima atomizacija vlog, udeleženih pri izdelavi načrtovanih izdelkov, lahko več koristnih posledic. Z njeno pomočjo lahko na ravni pridobivanja dodatnih interdisciplinarnih znanj izvajamo korekcije v organiziranosti in izvajanju službe za informatiko. Drobnozrnatost vlog omogoča usmerjanje v izobraževanje in dodatno usposabljanje, nosilcem vlog dopušča prostovoljno in participativno delovanje na projektu v okviru prevzetih nalog. V okolju, kjer imajo informatiki pester nabor temeljnih izobrazb, omogoča vključevanje v izvedbo projekta na ravni vloge, ki izhaja iz osnovne izobrazbene specializacije. Možno je tudi vzpostaviti način delovanja, kjer nosilec vloge določen čas (ali trajno) dela izključno za potrebe vloge v razvojnih projektih. Za takšne primere bi bilo koristno, da bi v sistemizaciji delovnih mest v ZGS obstajalo generično delovno mesto (na primer izvajalec projekta-specialist).

Za ZGS predlagani model prinaša priložnost za spremenjeno organizacijo delovanja službe za informatiko pri razvoju lastnih računalniških rešitev. Predlagan model dopušča centralizacijo dela pod okriljem odgovornega izvajalca in izvajalcev posameznega podprocesa na centralni enoti. Prav tako dopušča organizacijo dela, ki temelji na projektnem vključevanju posameznih (specialističnih) vlog informatikov iz območnih enot. Oba načina dela sta bila v preteklosti sicer že nekajkrat predlagana, vendar sta bila sprejeta z zadržki in se niti eden niti drugi nista dokončno uveljavila.

Presežek, ki ga lahko prinese jasna in eksplicitna opredelitev vlog v okviru predlaganega načina izvedbe razvojnih projektov, dopušča kombiniranje obeh, sicer že predlaganih načinov dela, poleg tega pa ima lahko še nekatere koristne učinke. Predlog izvedbe razvojnega procesa decentralizira odločanje in reševanje problemov na raven izvedbe podprocesov, vnaša več demokratičnosti pri oblikovanju delovnih skupin in imenovanju vsakokratnega odgovornega izvajalca (vodje) podprocesa, dopušča participativno udeležbo informatikov območnih enot ZGS, omogoča preciznejše načrtovanje potrebnih znanj in dodatnega usposabljanja izvajalcev, dopušča stimulacijo na osnovi prostovoljne udeležbe na projektih.

Nekaj omenjenih učinkov in dejavnikov lahko v veliki meri povežemo s konceptom delovanja učečih organizacij. V tem pogledu predstavlja predlagani način dela prilagojeno metodologijo razvoja računalniške rešitve, ki ima sočasno vgrajene nekatere mehanizme oziroma načela

delovanja učeče organizacije. Na tej ravni razumevanja predlagane izvedbe razvoja računalniških rešitev se odpira dodaten prostor za boljše izkoriščanje potencialov zaposlenih, njihovih sposobnosti in ambicij, pa naj gre za posodabljanje načinov vodenja skupin (vodja projekta, odgovorni izvajalec) ali za učenje (pridobivanje specialnih znanj) ali delovanje (izvajalec z vlogo) udeležencev nasploh.

### 5.7.6 Zbiranje zahtev s primeri uporabe, pripomočki

Za snovanje in izdelavo računalniške rešitve je zbiranje zahtev uporabnikov temeljni podproces oziroma razvojna faza v vsakem metodološkem pristopu razvoja. Običajno največ uporabnih informacij dobimo neposredno od izvajalcev aktivnosti. Ob komunikaciji z njimi se soočimo z njihovim besednjakom in bolj ali manj podrobnim seznamom aktivnosti, ki jih izvajajo. Zbiranje zahtev in analizo lahko izvajamo na osnovi vprašalnikov, razgovorov ali s pisanjem scenarijev izvedbe dela. Pogosto si pri analizi zahtev lahko pomagamo s preučevanjem organizacijskih predpisov ali izvedbenih delovnih navodil.

V svetu obstaja vrsta organizacij, ki si prizadevajo zagotavljati enotne storitve po enotnih kvalitativnih in kvantitativnih kriterijih. Običajno izvedbo storitev prikazujejo s procesnimi modeli in opisujejo izvedbo v obliki internih predpisov. Tudi neprofitne organizacije si čedalje bolj prizadevajo za izboljševanje izvedbe in dostopnosti storitev, ki jih zagotavljajo. Lahko rečemo, da se tudi v neprofitnem okolju čedalje bolj izvaja predstavitev izvedbe storitev v obliki procesov in opisov poteka dela, po katerem se oblikuje enoten standard izvedbe storitev.

ZGS zagotavlja enotni model storitev na razpršenem prostoru v izvedbi območnih organizacijskih enot. Strokovni oddelki na centralni enoti in območni strokovni odseki nastopajo v vlogi skrbnikov za poenoteno izvedbo dela in izdelavo navodil za izvedbo dela. Pod okrilje takšnih navodil lahko štejemo strokovne smernice izvedbe, predvidene časovne okvire, seznam spremljajočih izdelkov ali dodatnih storitev, merila kakovosti za opravljeno delo, obseg potrebnih pripomočkov in drugih virov za izvedbo dela.

Nabor že izdelanih navodil in usmeritev za izvedbo dela strokovnih gozdarskih oddelkov lahko štejemo med pripomočke za analizo pri zbiranju zahtev uporabnikov. Če so izvedbeni predpisi ohlapni in omogočajo različne interpretacije izvedbe, jih je potrebno za namene izdelave računalniške rešitve poenotiti. Predlagati je potrebno enoten nabor aktivnosti, še preden jih bomo računalniško podprli. Poenotenje izvedbenih aktivnosti posameznih opravil je koristno izvajati z vključevanjem in sodelovanjem izvajalcev procesov.

Na voljo imamo različne tehnike in pripomočke. Ena od možnosti je pisanje scenarijev izvajalcev procesa. Druga možnost izhaja iz metode upravljanja delovnih procesov. Dokumentiranje postopka, ki ga izvaja uporabnik, temelji na navedbi zaporedja izvedbenih aktivnosti v postopku in udeleženimi (uporabljenimi) vhodno-izhodnimi podatki. V splošnem metoda predvideva zbiranje različnih parametrov, s pomočjo katerih ugotavljamo, koliko časa in na kakšen način poteka izvedba nekega opravila. Procesni model, ki na ta način nastaja, ima širšo uporabno vrednost v spremljanju opravljenega dela, optimizaciji in vodenju poslovnih procesov.

V ZGS bi lahko združili oba pristopa. Končni izdelek bi bila poenotena navodila za izvedbo posameznih postopkov. Za namene računalniške podpore bi predstavljala osnovni vhodni pripomoček za izdelavo modela primerov uporabe. Na osnovi seznamov izvedbenih aktivnosti v

okviru vnaprej poimenovanih strokovnih postopkov lahko zbiramo dejanske scenarije izvedbe s pomočjo različnih izvajalcev. Scenariji lahko iz različnih razlogov izvedbeno variirajo. Na osnovi analize aktivnosti v predlaganih scenarijih lahko strokovni oddelki predlagajo (vzpostavijo) končni scenarij (model) izvedbe aktivnosti in potrebnih vhodno-izhodnih količin, za katerega pozneje načrtujemo tudi računalniško podporo.

Slika 12: Centraliziramo zbiranje scenarijev aktivnosti s pomočjo intraneta


Predlagan postopek je eden od možnih načinov za povezavo strokovnih oddelkov, uporabnikov in načrtovalcev računalniške rešitve na ravni vsebinskega modela podsistema (opravila). Tega za potrebe računalniške rešitve predstavimo s pomočjo modela primerov uporabe. Načrtovanje računalniške rešitve se nadaljuje s pretvorbo vsebinskega modela v sistemski model in izdelavo drugih pripomočkov v procesu predlagane metodologije. Vzpostavljena informacijsko komunikacijska infrastruktura omogoča izdelavo dodatnih pripomočkov za povezovanje udeležencev v razvoju računalniške rešitve.

### 5.7.7 Platforme, komponente, standardi, vzorci, modeliranje

Informacijsko komunikacijska infrastruktura v svoji zasnovi predpostavlja delovanje računalniških rešitev na osnovi porazdeljene uporabe sistemskih in podatkovnih virov, odjemalcev in ponudnikov storitev. Obseg potrebnega znanja za izvedbo predmetnih računalniških rešitev za okolje porazdeljenih sistemskih virov, je neodvisen od vrste organizacije v kateri izdelujemo take rešitve. Porazdeljevanje funkcionalnosti računalniških rešitev med oddaljene izvajalce lahko označimo kot razvojni trend sodobnih računalniških rešitev, ki ga je potrebno obvladovati v obeh tipih organizacij. Predmetni model računalniške rešitve se spreminja v porazdeljen predmetni model, kjer oddaljeni predmeti zagotavljajo zaključene dele funkcionalnosti končne rešitve. Pozivanje oddaljenih metod sloni na različnih objektnih modelih in standardih. To dejstvo pogosto ovira medsebojno klicanje oddaljenih metod objektov med računalniškimi rešitvami, izdelanimi na osnovi uporabe različnih programskih jezikov. Če želimo uspešno prebroditi to razvojno past, se je potrebno bodisi dobro seznaniti z odprtim

standardom porazdeljenega objektnega sistema (CORBA), bodisi uporabiti kakšno od enotnih razvojnih platform za izdelavo porazdeljeno delujočih računalniških rešitev (.NET, Java).

V okolju, ki je v izdelavi računalniških rešitev v večji meri naslonjeno na uporabo že narejenih komponent, je tudi načrtovanje v večji meri navezano na iskanje modulov oziroma poznavanje storitev in funkcionalnosti, ki jo ponujajo. Področje načrtovanja in izvedbe računalniških rešitev lahko navežemo na obsežen nabor odprtokodnih rešitev, kjer lahko uporabimo prosto dostopne komponente ali si pomagamo s preučevanjem koncepta v njih uporabljenih rešitev. V splošnem za vsa razvojna okolja, tako profitna kot neprofitna, velja, da distribuirana izvedba predmetnih računalniških rešitev poleg poznavanja uporabe predmetnih programskih jezikov in nekaterih metodoloških principov razvoja računalniških rešitev že v začetni fazi izvedbe rešitve prinaša s seboj zahtevo po dodatnem tehničnem znanju na področju poznavanja protokolov, standardov, delovanja komponent in vmesne programske opreme.

Vse navedeno velja tudi za proces lastne izdelave predmetnih računalniških rešitev v ZGS. Končne točke zaprtega omrežja (intranet ZGS) se za izvedbo nekaterih storitev spreminjajo v lahke odjemalce (brskalnike), ki jih uporabljamo za predstavitev centralno hranjenih podatkov. Nekatere pogoste in splošne podatkovne storitve so že na voljo na strežnikih upravljalcev teh podatkov. To dejstvo je potrebno upoštevati, če bo ZGS želel v svojih aplikacijah uporabljati tudi podatke državne prostorske infrastrukture.

Ugotovljeno je bilo, da v ZGS dolgoročno obstaja tudi potreba po informacijski podpori storitev (notranjih), v obliki posebnih namenskih programskih rešitev in orodij. Z njimi poteka vzpostavitev, priprava in predstavitev podatkov, ki še potrebujejo kreativnost končnega uporabnika v obliki interpretacije in spremljajočih komentarjev (preoblikovanje prostorskih entitet, preglednice, izdelava posebnih dokumentov). Dele zahtevane funkcionalnosti je možno zagotoviti tudi z uporabo programskih komponent. S stališča pičlo odmerjenih finančnih virov je smiselno preučevati in uvajati predvsem tiste, ki so javno in brezplačno dostopne. Po potrebi in v okviru možnosti prilagoditve je smiselno tudi preoblikovanje izvornega objektnega modela iz domene delovanja. Ob takšnem pristopu dobi večji pomen tudi uporaba pripomočkov, kot so primeri vzorcev zahtev in načrtovalski vzorci programskih rešitev. Na osnovi izbire standarda vzajemnega delovanja komponent je možno utemeljiti tudi odločitev o izbiri enotnega razvojnega okolja (izbira platforme).

Uvajanje predmetne informacijske tehnologije v ZGS ima v danih razmerah svoj največji potencial v kombinaciji z uvajanjem komponentne večnivojske arhitekture programskih rešitev. Izdelava oziroma uporaba modulov (delnih rešitev sistema) na nivoju aplikativnih storitev v večnivojski informacijski arhitekturi, po načelih predmetnega razvoja, je v danem trenutku posodabljanja informacijskih storitev, glede na možnosti in glede na razvojne trende spletne tehnologije, za ZGS izvedba pravih stvari na pravi način. Predlagan pristop omogoča tako predmetni razvoj lastnih komponent kot integracijo obstoječih računalniških rešitev v okvir celovitega (sicer heterogeno sestavljenega) informacijskega sistema ZGS. Takšnemu pristopu lahko pripišemo več koristnih učinkov:

- izvajanje informacijske podpore s pomočjo različnih povezanih računalniških rešitev (podsistemov) omogoča preprostejše vzdrževanje in nadgradnje,
- ohranjanje nekaterih računalniških rešitev,
- krajši razvojni cikli pri uvajanju novih računalniških rešitev, saj ne gradimo ene same računalniške rešitve, ampak samo dele za podporo izbranim procesom,
- možnost integracije različnih tržno dostopnih plačljivih ali odprtokodnih komponent,
- možnost dopolnjevanja odprtokodnih komponent,

- naročanje komponent pri zunanjih izvajalcih in možnosti za prenos znanja,
- možnost postopnega pridobivanja znanja uporabe objektne tehnologije na manjših projektih,
- zagotavljanje spletnih računalniških rešitev in uvedba lahkih odjemalcev storitev.

Navedene koristne učinke lahko obravnavamo tudi kot priložnosti za načrtno pridobivanje novega znanja in posodobitve načina delovanja službe za informatiko v ZGS.

### 5.7.8 Delovanje oddelka za informatiko

Med dejavniki, ki jih je potrebno upoštevati pri izvajanju informacijske podpore, so tudi kadri informatike, organiziranost informatike v podjetju in način delovanja službe za informatiko. V profitnem delovnem okolju se uveljavlja zavedanje, da nove tehnološke možnosti in spremenjeni infrastrukturni pogledi na informatiko terjajo posodabljanje organiziranosti. Eden od modelov organiziranosti, primeren tudi za geografsko razpršena podjetja, predlaga funkcionalno organiziranost na osnovi delovanja skupin za razvoj, načrtovanje, pomoč in storitve ter skupin za informatiko poslovnih enot. V profitnih delovnih okoljih od kadrov informatike pričakujejo številna interdisciplinarna znanja in pripravljenost na neprestano prilagajanje na nove spremembe. Kot pomoč organizacije navedenim pričakovanjem je predlagano načrtovanje in usmerjanje pridobivanja novega znanja, spodbujanje kroženja in izmenjave znanja. Profitna okolja se pogosto navezujejo tudi na zunanje izvajalce v izvajanju nalog informatike. Kot dejavnik uspešnosti informatizacije procesov s pomočjo zunanjih izvajalcev je izpostavljena dobra komunikacija med naročnikom in izvajalcem.

Profitne organizacije so razvile različne modele organiziranosti in vzorce delovanja, ki so primerni tudi za delovanje službe za informatiko v neprofitnem delovnem okolju. Pri obsegu zahtevanega znanja informatikov v obeh okoljih praviloma ne bi smeli iskati razlik. Dejstvo pa je, da je pridobivanje specialnih znanj drago in si ga profitne organizacije praviloma lažje privoščijo od neprofitnih. V teh okoljih še vedno lahko naletimo na vzorec razmišljanja, ki omejuje obsežna vlaganja v pridobivanje znanja zaradi strahu po prebegu usposobljenega kadra v profitno delovno okolje. Za neprofitno delovno okolje je morda ena od groženj uspešnejšega delovanja službe za informatiko odsotnost mehanizmov stimulacije za neprestano prilagajanje kadrov na nove spremembe. V specifičnem okolju se kot grožnja lahko pojavlja tudi nepripravljenost posameznikov po delitvi znanja z drugimi. Ta nepripravljenost delitve znanja bodisi ščiti doseženi položaj v hierarhiji in izkazuje nepogrešljivost posameznika, bodisi izvira iz presežka lastnih vlaganj v izobraževanje, ki jih posameznik ni pripravljen podariti. Med grožnje uspešnemu delovanju službe za informatiko v neprofitnem delovnem okolju zagotovo lahko prištejemo odsotnost sodobnih oblik vodenja ter skupinskega in projektnega dela. Vsako od zaznanih groženj lahko obravnavamo tudi kot zahtevo po spremembi. Uvajanje posameznih sprememb na ravni funkcijskega oddelka je lahko počasen in za marsikoga težaven proces. Zdi se, da je v okolju, ki zazna večje število groženj oziroma potrebo po številnih spremembah, potrebno iskati rešitve na višji organizacijski ravni kot je funkcijska. Enega od modelov, ki dokazuje uspešno prilagajanje na spremembe, predstavlja učeča organizacija.

V pogojih dane organiziranosti službe za informatiko je potrebno skrbeti tudi za usmerjanje delovanja. Tudi smernice strateškega delovanja oddelka za informatiko na funkcijski ravni organizacije lahko poiščemo z uporabo metod strateškega načrtovanja. V uvodnih poglavjih so bili obravnavani nekateri pristopi in tehnike strateškega načrtovanja. Pogosto je citiran in uporabljan harvardski model politike organizacije, ki sloni na analizi prednosti, slabosti,

nevarnosti in priložnosti (SWOT), in je usmerjen v razvoj ujemanja delovanja organizacije in njenega okolja. Tovrstna analiza ima splošno uporabno vrednost v različnih situacijah, ki zahtevajo sprejemanje odločitev ali usmerjanje delovanja.

Eden od zastavljenih ciljev naloge je iskanje konkretnih strateških aktivnosti delovanja službe za informatiko v ZGS na osnovi dejavnikov, povezanih z uvajanjem predmetne informacijske tehnologije. Odločil sem se, da v iskanju možnih strateških aktivnosti uporabim SWOT analizo. Ta temelji na vrednotenju nekaterih dejavnikov razvoja ocenjenih iz ravni delovanja službe za informatiko v ZGS. Stanje oziroma vrednotenje dejavnikov sem ocenjeval s pomočjo ankete med nekaterimi potencialnimi nosilci projekta posodobitve informacijske podpore.

Metodi SWOT analize je večkrat očitano, da dopušča preveč subjektivnih dejavnikov, zato je za njeno izboljšavo predlagano izpolnjevanje večjega števila SWOT matrik. Prav tako je analiza brezpredmetna, če njenega namena ne opredelimo z dosegljivim ciljem, ki ga želimo doseči. Svojo ad hoc SWOT analizo sem deloma prilagodil tem zahtevam in jo zasnoval na naslednji način:

- izpostavil sem cilj, ki se glasi - oddelek za informatiko ZGS želi izdelati lastno računalniško rešitev, ki naj bi uporabljala podatke iz različnih virov, delovala v mreži povezanih računalnikov in bi jo sami izdelali z uporabo objektnega programskega jezika,
- na osnovi preučevanja različnih dejavnikov, predstavljenih v uvodnih poglavjih, sem izdelal seznam tistih, ki se v strokovni literaturi največkrat pojavljajo v zvezi s projekti razvoja objektno zasnovanih računalniških rešitev,
- seznam dejavnikov, povezanih z načrtovanjem in izvedbo računalniške rešitve, sem dopolnil z nekaterimi dejavniki, ki se navezujejo na organiziranost službe za informatiko v ZGS ter nekaterimi dejavniki, ki glede na moje poznavanje lahko od zunaj vplivajo na delovanje službe (povezovanje z drugimi ustanovami, možnosti outsorcinga, sodelovanje z drugimi strokovnimi oddelki),
- celoten seznam, ki v skupnem predstavlja tako notranje dejavnike za izvedbo cilja kot zunanje dejavnike, ki lahko vplivajo na doseganje cilja, sem dal v presojo štirim sodelavcem, ki imajo ključno vlogo za delovanje službe za informatiko na nivoju centralne enote ZGS (skrb za infrastrukturo, administracija podatkov, razvoj računalniških rešitev) in jih pozval, naj se opredelijo do pomembnih ter izločijo po njihovem mnenju nerelevantne dejavnike za doseganje cilja. Vsak je imel možnost dopolnitve morebiti manjkajočih dejavnikov s stališča doseganja zastavljenega cilja. Sam ankete nisem izpolnjeval, prav tako je nisem dal izpolniti vodji oddelka. Želel sem odkriti dejavnike, ki izhajajo iz razumevanja oziroma vedenja povsem operativnih izvajalcev funkcije informatike,
- v drugem koraku sem sodelavce pozval na izpolnitev druge ankete, ki je vsebovala različne trditve v povezavi z navedenimi dejavniki v prvi anketi. Na osnovi podajanja kvalitativne ocene sodelavcev (zelo dobro, dobro, slabo, zelo slabo in ne vem) sem izdelal kvalitativno opredeljeno listo dejavnikov, ki sem jih tokrat po svoji presoji glede na referenčno znanje, pridobljeno z izdelavo naloge, razporedil v SWOT matriko. Na tej osnovi sem izdelal pregled dejavnikov prednosti, slabosti, nevarnosti in priložnosti za doseganje zastavljenega cilja,
- od štirih pozvanih anketirancev so obe anketi izpolnili trije, od katerih vsak pokriva po eno od vlog informatike: skrb za infrastrukturo, administracija podatkov, razvoj računalniških rešitev. Drugi od razvijalcev računalniških rešitev, ki ankete v štirinajstih

dneh ni izpolnil, je sprva želel spremeniti nekaj anketnih vprašanj. Po nekajkratnem pozivu na izpolnitev izvorne ankete se je izgovoril, da ankete ne more izpolniti zaradi prezasedenosti in obilice prevzetega dela.

Tabela 4: SWOT analiza, teža dejavnikov in kvalitativna ocena stanja

| vrsta | rang | Dejavniki, ki po mnenju anketiranih pomembno vplivajo na doseganje cilja (rangirani so glede na doseženo stopnjo uteži, 6 – najpomembnejši, 1 – nepomembni). |
|---|---|---|
| NOTRANJI DEJAVNIKI | 6 | poznavanje metod dokumentiranja zahtev uporabnikov |
| | 5 | poznavanje metodologije objektnega načrtovanja za razvoj aplikacij |
| | 5 | razpolaganje z ažurnim prikazom arhitekture informacijskega sistema ZGS |
| | 5 | povečevanje vlaganj v znanje programiranja |
| | 5 | povečevanje vlaganj v druga znanja povezana s programiranjem |
| | 5 | kupovanje razvojne pomoči pri zunanjih razvijalcih |
| | 5 | zagotavljanje vpogleda v podatke javne gozdarske službe javnosti preko spleta |
| | 4 | vzpostavitev obveznega dokumentiranja končnih računalniških rešitev |
| | 4 | uporaba metodologije pri OO razvoju aplikacije |
| | 4 | izdelava objektnega modela entitet v domeni gozdarskega delovanja |
| | 4 | izvajanje razvoja lastnih aplikacij na podlagi podrobnih projektov |
| | 4 | uporaba enega od modelirnih jezikov med udeleženci razvoja lastnih računalniških rešitev |
| | 4 | uvedba projektnega dela v razvoj lastnih aplikacij |
| | 4 | strateško načrtovanje pogojev za uporabo objektnih programskih rešitev in izdelavo lastnih objektnih aplikacij |
| | 4 | sprememba načina vodenja razvoja računalniških rešitev |
| | 3 | izvedba računalniških rešitev na osnovi boljšega znanja načrtovanja in dokumentiranja |
| | 3 | izvedba računalniških rešitev z iskanjem uspešnejših oblik skupinskega dela |

| Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini | dobro | slabo |
|---|---|---|
| Našo stopnjo kakovosti izvajanja dokumentiranja zahteve uporabnika ocenjujem kot: | | xxx |
| Našo doseženo stopnjo poznavanja objektnega programiranja za razvoj aplikacij ocenjujem kot: | | xxx |
| Našo doseženo stopnjo poznavanja tehnik objektnega načrtovanja za razvoj aplikacije ocenjujem kot: | | xxx |
| Našo stopnjo kakovosti dokumentiranja končne računalniške rešitve ocenjujem kot: | | xxx |
| Naš obseg poznavanja metodologij za OO razvoj aplikacije ocenjujem kot: | | xxx |
| Našo kakovost dokumentiranosti arhitekture informacijskega sistema ocenjujem kot: | x | xx |
| Svojo stopnjo poznavanja objektnega modela entitet v domeni našega delovanja ocenjujem kot: | | xx (*) |

| vrsta | rang | Dejavniki, ki po mnenju anketiranih pomembno vplivajo na doseganje cilja (rangirani so glede na doseženo stopnjo uteži, 6 – najpomembnejši, 1 – nepomembni). |
|---|---|---|
| NOTRANJI DEJAVNIKI | | |

| Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini | dobro | slabo |
|---|---|---|
| Trditev, da je za uspešen razvoj objektne aplikacije potreben načrt, ki izhaja iz objektnega modela ocenjujem kot: | xx | (*) |
| Našo operativno usposobljenost za objektno programiranje ocenjujem kot: | | xxx |
| Našo operativno sposobnost za izvajanje razvoja računalniške rešitve v obliki procesa ocenjujem kot: | x | xx |
| Našo operativno sposobnost za povezovanje sedaj razdrobljenih aplikacij v povezano celoto v skladu s pričakovanji strokovnih oddelkov ocenjujem kot: | x | xx |
| Naše izkušnje in stopnjo učinkovitosti izvajanja projektnega dela ocenjujem kot: | | xxx |
| Svojo stopnjo poznavanja enega od modelirnih jezikov ocenjujem kot: | | xxx |
| Učinek našega dosedanjega kupovanja pomoči pri zunanjih razvijalcih ocenjujem kot: | x | xx |
| Našo stopnjo zagotavljanja vpogleda v podatke javne gozdarske službe javnosti preko interneta ocenjujem kot: | | xxx |
| Način izvedbe razvoja aplikacije v obliki projektne naloge ocenjujem kot: | xx | |
| Obseg našega razpoložljivega znanja za izdelavo spletne aplikacije ocenjujem kot: | | xxx |
| Prenos razvoja novih aplikacij na izvajalce izven ZGS in kasnejše lastno vzdrževanje teh aplikacij ocenjujem kot: | xx | x |
| Prenos razvoja novih aplikacij na izvajalce izven ZGS in tudi prenos vzdrževanja na zunanje ponudnike ocenjujem kot: | x | xx |
| Trditev, ki pravi skrb za razvoj aplikacij v ZGS naj sloni na programerjih posameznikih, ki znajo programirati in znajo sami poiskati zahteve uporabnikov in izdelati končno aplikacijo ocenjujem kot: | x | xx |
| Trditev, ki pravi skrb za razvoj aplikacij v ZGS naj sloni na skupinskem delu posameznikov, ki znajo programirati in si znajo sami razdeliti delo dokumentiranja, načrtovanja in izdelave aplikacije ocenjujem kot: | x | xx |
| Trditev, ki pravi skrb za razvoj aplikacij v ZGS naj sloni na podrobnem projektu, ki opredeli vloge in odgovornost za skupinsko delo posameznikov pri zbiranju zahtev, načrtovanju in izdelavi aplikacije. Projekt naj vodi koordinator, katerega dolžnost je spremljati delo in poročati o problemih, ki se rešujejo skupno ocenjujem kot: | xx | x |

| | |
|---|---|
| 6 | navezovanje na prostokodne informacijske rešitve, uporaba neplačljivih orodij |
| 3 | navezovanje na plačljivo informacijsko tehnologijo in uporaba plačljivih orodij (npr. Microsoft) |

| vrsta | rang | Dejavniki, ki po mnenju anketiranih pomembno vplivajo na doseganje cilja (rangirani so glede na doseženo stopnjo uteži, 6 – najpomembnejši, 1 – nepomembni). |
|---|---|---|
| NOTRANJI DEJAVNIKI | | |

| Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini | dobro | slabo |
|---|---|---|
| Našo doseženo stopnjo operativne uporabe (obvladovanja) neplačljivih razvojnih orodij, ki jih uporabljamo v ZGS ocenjujem kot: | | xxx |
| Našo doseženo stopnjo operativne uporabe (obvladovanja) plačljivih razvojnih in drugih orodij, ki jih uporabljamo v ZGS ocenjujem kot: | xx | x |

6 ustvarjanje pogojev za lastno administriranje baz podatkov
6 uporaba centralne relacijske podatkovne baze (MySQL)
5 znanje administriranja baze (MySQL) v lastni režiji
5 uporaba centralne geoprostorske podatkovne baze

| Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini | dobro | slabo |
|---|---|---|
| Naše poznavanje relacijske podatkovne baze – Oracle ocenjujem kot: | | xxx |
| Naše poznavanje relacijske podatkovne baze - MySQL ocenjujem kot | xxx | |
| Naše poznavanje podatkovne baze DBase ocenjujem kot: | xx | (*) |
| Prenos administriranja podatkov izven ZGS ocenjujem kot: | | xxx |
| Našo operativno usposobljenost za vzpostavitev neoporečnih podatkov (kakovost, celovitost) ocenjujem kot: | x | xx |

6 ustvarjanje pogojev za lastno zagotavljanje podpore uporabnikom informacijskih rešitev
6 vzpostavitev delovanja službe za informatiko na nivoju enega razvojnega centra brez delegiranja razvojnih nalog na informatike območnih enot
5 okrepitev službe za informatiko na CE in preoblikovanje v obliko centra informatike
5 povečevanje stimulacije nosilcev razvojnega dela

| Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini | dobro | slabo |
|---|---|---|
| Morebitno preoblikovanje organiziranosti službe za informatiko po načelu organiziranosti drugih strokovnih oddelkov ocenjujem kot: | x | xx |
| Morebitno preoblikovanje organiziranosti službe za informatiko po načelu, da razvoj aplikacij teče samo na CE ocenjujem kot: | xxx | |
| Možnost, da se informatiki OE prostovoljno odločajo o sodelovanju na projektu razvoja objektne aplikacije ocenjujem kot: | xx | x |

<table>
<tr><th>vrsta</th><th>rang</th><th>Dejavniki, ki po mnenju anketiranih pomembno vplivajo na doseganje cilja (rangirani so glede na doseženo stopnjo uteži, 6 – najpomembnejši, 1 – nepomembni).</th></tr>
<tr><td>NOTRANJI DEJAVNIKI</td><td></td><td>
<table>
<tr><th>Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini</th><th>dobro</th><th>slabo</th></tr>
<tr><td>Možnost, da se informatikom OE predpisuje obveznosti pri sodelovanju na projektu razvoja objektne aplikacije ocenjujem kot:</td><td>x</td><td>xx</td></tr>
<tr><td>Možnost razvoja lastne aplikacije deloma s kadri na CE, deloma s kadri na OE ocenjujem kot:</td><td></td><td>xxx</td></tr>
<tr><td>Trditev, ki pravi skrb za razvoj aplikacije v ZGS je izključna domena oddelka za informatiko ocenjujem kot:</td><td>x</td><td>xx</td></tr>
<tr><td>Trditev, ki pravi skrb za razvoj aplikacije v ZGS je deloma domena strokovnega oddelka deloma domena oddelka za informatiko ocenjujem kot:</td><td>xx</td><td>x</td></tr>
<tr><td>Trditev, ki pravi skrb za razvoj aplikacije v ZGS je izključna domena strokovnega oddelka, ki si po potrebi najde pomoč ali svetovalce v oddelku za informatiko ocenjujem kot:</td><td></td><td>xxx</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">ZUNANJI DEJAVNIKI</td><td>6<br>4</td><td>vzpostavitev povezanosti organizacijskih enot ZGS v enotno računalniško mrežo<br>strateško načrtovanje pogojev za lastno vzdrževanje informacijske strojne infrastrukture
<table>
<tr><th>Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini</th><th>dobro</th><th>slabo</th></tr>
<tr><td>Stopnjo in kakovost povezanosti OE in KE v enotno mrežo na nivoju SLO ocenjujem kot:</td><td>xxx</td><td></td></tr>
<tr><td>Prenos vzdrževanja infratrukture (mreže, strežniki) izven ZGS ocenjujem kot:</td><td></td><td>xxx</td></tr>
</table>
</td></tr>
<tr><td>6<br>5<br>5<br>4<br>4<br>4<br>4<br>3<br>3</td><td>vzpostavitev dogovora z drugimi strokovnimi oddelki o standardnem načinu podajanja zahtev za informacijsko podporo<br>zagotavljanje dostopanja do baz podatkov drugih ustanov<br>dokumentiranje delovnih procesov, ki jih izvaja ZGS v obliki izvedbenih navodil izvajalcev procesov<br>izdelava procesnega modela delovanja ZGS<br>izdelava procesnega modela delovanja javne gozdarske službe<br>ločevanje računalniških rešitev z gozdarskimi vsebinami od rešitev računovodskih in kadrovskih vsebin<br>izboljšanje sodelovanja s strokovnimi oddelki<br>načrtovanje združljivosti programskih rešitev z MKGP-Ministrstvo<br>iskanje možnosti zagotavljanja storitev (ne samo vpogleda v podatke) javne gozdarske službe javnosti preko spleta</td></tr>
</table>

| vrsta | rang | Dejavniki, ki po mnenju anketiranih pomembno vplivajo na doseganje cilja (rangirani so glede na doseženo stopnjo uteži, 6 – najpomembnejši, 1 – nepomembni). |
|---|---|---|
| ZUNANJI DEJAVNIKI | | |

| Kvalitativna ocena stanja ali trditve v povezavi z dejavniki v skupini | dobro | slabo |
|---|---|---|
| Našo doseženo stopnjo dostopanja do tujih (javna uprava) baz podatkov ocenjujem kot: | x | xx |
| Našo stopnjo sodelovanja na področju računalniških rešitev z MKGP-Ministrstvom ocenjujem kot: | x | xx |
| Našo stopnjo izvajanja storitev javne gozdarske službe javnosti preko interneta ocenjujem kot: | | xxx |
| Svojo stopnjo poznavanja procesnega modela delovanja ZGS ocenjujem kot: | x | xx |
| Naš obseg razpolaganja z izvedbenimi delovnimi navodili za opravila strokovnih oddelkov, ki jih računalniško podpiramo ocenjujem kot: | x | xx |
| Doseženo stopnjo sodelovanja službe za informatiko z drugimi strokovnimi oddelki (predvsem na načinu podajanja zahtev za informacijsko podporo) ocenjujem kot: | xx | x |

Opombe: (*) – brez odgovora

V preglednici (tabela 4.) so notranji in zunanji dejavniki rangirani po stopnji njihove ponderirane pomembnosti za doseganje zastavljenega cilja. Razporejeni so po vrsti, od najpomembnejših proti manj pomembnim. V modelu ponderiranja so imeli dejavniki, prepoznani kot pomembni, utež 2, pogojno ali morebiti pomembni dejavniki utež 1, nepomembni pa utež 0. Ponderiranje je bilo izvedeno kot vsota zmnožkov uteži in števila odgovorov, pripisanih posameznemu dejavniku in ocenjenemu pomenu. Zaradi preglednosti so v preglednici izpuščeni dejavniki, katerih ponder je bil manjši od 3.

Na osnovi kvalitativnih ocen lahko razberemo stanje, ki ga zaznavajo anketiranci. Iz kvalitativne ocene anketirancev lahko podamo mnenje, ali posamezen dejavnik krepi (prednost, priložnost) ali slabi (slabost, nevarnost) možnost za doseganje cilja. V analizi predstavljajo dobro ocenjeni dejavniki skupino koristnih dejavnikov, slabo ocenjeni pa škodljive dejavnike za doseganje cilja. Dejavniki z dvema ali s tremi slabimi ocenami terjajo ukrepanje in deloma nakazujejo načrtovanje ustrezne strateške aktivnosti.

Subjektivno gledano je katerega od dejavnikov mogoče tudi drugače razporediti, vendar končna razporeditev omogoča dovolj pregleden vpogled v nekaj skupin dejavnikov, ki jih lahko, razvojno strateško gledano, obravnavamo kot prednosti, slabosti, nevarnosti ali priložnosti. Če želimo delati prave stvari na pravi način, bi si morali, izhajajoč iz analize, postaviti še vprašanje, kako na celovit način strateško usmerjati razvoj oddelka za informatiko, da bi poleg zastavljenega cilja dosegli čim več vzporednih učinkov.

Skozi preučevanje primerov razvoja računalniških rešitev je razvidno, da je v zagotavljanju računalniških rešitev potrebno tehnično znanje, poleg tega pa je treba razumeti organizacijo, delovanje podjetja, poslovne procese in zahteve uporabnikov. Vzpostavljanje komunikacije med načrtovalcem in uporabnikom ter med razvijalci in izvajalci vodenja projekta in ne nazadnje vodstvom podjetja predstavlja pomemben dejavnik uspeha v procesu zagotavljanja celovite informacijske podpore. Znanje s področja domene, za katero načrtujemo računalniško rešitev, je

potrebno predstaviti v obliki, ki bo razumljiva za vse udeležence, ki sodelujejo pri načrtovanju in razvoju računalniških rešitev. Za vzpostavljanje komunikacijskega kanala in boljše razumevanje obravnavanih problematik je koristno izrabiti prednosti modeliranja. Modeli imajo trajnejši pomen v razumevanju delovanja sistemov. Lahko jih uporabimo kot povezovalni člen med teoretičnimi koncepti (domena stroke) in izvedbenimi koncepti (domena ponudnikov storitev).

V dani situaciji ZGS lahko za ključni dejavnik uspešne računalniške rešitve proglasimo informatika programerja, za katerega pa je iluzorno pričakovati, da lahko sam prevzame vse zgoraj navedene zahteve. Iz analize je razvidno, da se anketiranci dobro zavedajo pomena ključnih dejavnikov, ki vplivajo na doseganje zastavljenega cilja. Pri opredeljevanju priložnosti in groženj so mnenja večkrat deljena. Kot priložnost in prednost je izpostavljena suverenost nad vzpostavljeno računalniško povezanostjo organizacijskih enot in administriranjem podatkovne baze. Po drugi strani se kot slabost in grožnja izkazuje odsotnost in pomanjkanje vrste znanj, ki spremljajo lasten razvoj računalniških rešitev.
Delitev dela med nosilce razvoja na osnovi strateške zasnove informacijske arhitekture in načrtovanega razvoja programskih rešitev je racionalen odgovor na zastavljene izzive celovitega razvoja informatike. Pristop k razvoju računalniške rešitve bi moral imeti nekaj metodoloških korenin. Predlagana metodologija razvoja za ZGS izpostavlja ključen pomen vloge v razvoju računalniške rešitve. V pogojih, ko so zahtevana številna interdisciplinarna znanja in je število izvajalcev omejeno, je potrebno projekte zasnovati na vlogah. Na osnovi seznama ciljnih izdelkov in izvajalcem dodeljenih vlog lahko utemeljimo oblikovanje delovnih skupin. Trenutna organiziranost ZGS dopušča, da se za posamezne vloge specializirajo tudi informatiki na območnih enotah. Pri zasnovi objektnega modela imajo lahko nepogrešljivo vlogo tisti območni informatiki, ki imajo gozdarsko strokovno izobrazbo. Analiza kaže, da sta tako prostovoljna udeležba na razvoju rešitev kot centralizacija nekaterih funkcij službe za informatiko (programiranje) prepoznana kot dejavnika priložnosti. Odločitev o enotnem razvojnem modelu izdelave lastnih računalniških rešitev, ki bi omogočal participativno sodelovanje izvajalcev in ustrezna organizacijska prilagoditev delovanja skupin izvajalcev na centralni (ali drugi organizacijski) enoti, se zdi v takih pogojih modra odločitev.

V takšnih razmerah je močneje izpostavljena vloga vodje oddelka. Poleg investiranja v pridobivanje novega znanja za posamezne vloge v razvojnih projektih, mora poskrbeti za kroženje in izmenjavo znanja med udeleženci razvojnih projektov. Oblikovanje delovnih skupin, jasno opredeljeni cilji, motivacija udeležencev, spremljanje rezultatov, skrb za primerno delovno okolje in nagrajevanje so med ključnimi dejavniki uspešnega ravnanja s človeškimi viri. Aktivni vodja in učeča organizacija ponujata pravi model odzivanja v situaciji, ko je zaznan razvojni zastoj, ki ga lahko razberemo tudi pri iskanju smernic strateškega delovanja oddelka za informatiko ZGS.

## 5.8 Grožnje in priložnosti v uvajanju predmetne računalniške tehnologije

Uvajanje predmetne tehnologije je danes prevladujoč vzorec v razvoju računalniških rešitev. To, kar predmetno tehnologijo dela privlačno za njene razvijalce in uporabnike, je način razmišljanja o svetu, ki nas obdaja. Snovanje in uporaba računalniških rešitev namreč temelji na abstraktnem posploševanju in sestavljanju entitet, s katerimi se srečujemo pri vsakdanjem delu in življenju. Vzpostavljanje teh predmetov v medsebojno relacijo, razmišljanje o mehanizmih, ki predmetom v relaciji dajejo možnost delovanja, se pravi izvajanja aktivnosti s posledico, ki je za nas uporabna, poenostavlja razumevanje med udeleženci razvoja in uporabniki.

S pomočjo predmetov kartiramo fizične modele iz problemske domene in jih pretvarjamo v računalniške rešitve. V tem procesu potekajo podprocesi, ki zahtevajo predstavitev delovanja sistema, organizacijo in izvedbo projekta razvoja računalniške rešitve. S tem je povezana pogosta in nedvoumna komunikacija tako med načrtovalci in izvajalci računalniških rešitev kot tudi z uporabniki, ki jim je rešitev namenjena. Organizacija in vodenje projektov razvoja računalniških rešitev je največji izziv in v tem pogledu največja grožnja uspešni izvedbi zastavljene naloge. Drugi, s tem povezan izziv, je poznavanje uspešnih metodoloških pristopov razvoja in njihova prilagoditev na razmere, v katerih izvajamo projekt, saj se metodologija običajno navezuje na organizacijo, vodenje in izvedbo takšnih projektov. Izmenjava spoznanj o delovanju sistema med udeleženci razvoja, ki imajo različna vrednostna in strokovna izhodišča, na osnovi katerih vzpostavljajo vzajemno sodelovanje, pa je tretja grožnja uspešnega uvajanja predmetnih računalniških rešitev.

Izdelava lastne računalniške rešitve pogosto zahteva oblikovanje ad hoc delovnih skupin, uvedbo projektnega načina dela, izpostavljena je funkcija vodenja. Med grožnje uspešnega lastnega razvoja računalniških rešitev v neprofitnem delovnem okolju lahko uvrščamo tekmovanje drugih funkcijskih oddelkov za omejene vire (kadre informatike) in njihovo razporejanje na izvedbo drugih „nujnih“ strokovnih projektov, ki v določeni meri potrebujejo vlogo informatika. Netransparentnost prioritet projektov in delegiranja nalog kadrom informatike lahko povzroča konflikt vodenja, zamegljuje spremljanje dela, pa tudi ogroža organizacijo novih razvojnih projektov.

Osredotočanje na organizacijo projektov razvoja računalniških rešitev lahko spodbudi razmislek o notranji reorganizaciji ali preoblikovanju načina dela. Pri tem prihaja do zahtev po sodelovanju vodstva podjetja in preverjanja temeljnih strateških usmeritev podjetja. Osredotočanje na posamezne vloge, ki so ključne za izvedbo načrtovanih računalniških rešitev, lahko pomaga oblikovati strategijo izobraževanja in dodatnega usposabljanja. Vse to so priložnosti za spremembe dosedanje organiziranosti in ustaljenih vzorcev dela.

Ena od groženj v delovanju neprofitnih organizacijah je lahko povezana z načinom pridobivanja dodatnega znanja in usposobljenosti. Omejene možnosti kupovanja manjkajočega znanja na trgu, toge oblike organiziranosti, vzorci individualnega delovanja zaposlenih in odsotnosti menedžerskih znanj med strokovnjaki, ki uresničujejo svoje smotre v delovanju organizacije, pogosto vodijo do samoizobraževanja posameznikov. Takšen način pridobivanja znanja je sicer dobrodošel v vsakem okolju, vendar ima lahko tudi slabo stran. Neprofitno organiziranost pogosto spremlja odsotnost mehanizmov nagrajevanja, zato posameznik ni pripravljen takšnega znanja deliti z drugimi, lahko prihaja do namernega ograjevanja znanja zaradi zaščite doseženega položaja v hierarhiji. Iz teh razlogov pridobljeno znanje ostaja izolirano od znanja drugih sodelavcev in obstaja v organizaciji na ravni neizkoriščenega potenciala. Drug problem lahko predstavlja sistematičen način usposabljanja, v ozadju katerega je včasih le motiv za zbiranje certifikatov napredovanja. Program tovrstnih usposabljanj bi moral vnašati poleg novih znanj tudi jasna sporočila o njihovem pomenu za strategijo delovanja organizacije.

Proces izdelave računalniških rešitev zahteva združevanje znanja iz domene dejavnosti, ki ji je rešitev namenjena in znanja s področja predmetne računalniške tehnologije, v končno skladno celoto. V procesu vzajemne komunikacije udeležencev vzpostavimo vrsto pripomočkov, katerih uporabna vrednost lahko presega izvorni namen njihovega nastanka. Ti pripomočki se lahko izkažejo s svojo večnamensko uporabnostjo, kar je lahko presežek oziroma dodatna pridobitev (priložnost) za organizacijo. To še zlasti drži, če sledimo modelu delovanja učeče organizacije, ki na celovit način dokumentira in hrani svoje znanje. Spremljajoče pripomočke in dokumentacijo

razvoja računalniških rešitev uporabljamo za postopke vzdrževanja, nadgradnje ali ponovne uporabe, lahko pa jo koristno uporabimo tudi v modeliranju poslovnih procesov, organiziranju dela ali izdelavi vsebinskih navodil za izvedbo različnih opravil. Izdelava lastnih računalniških rešitev na takšni osnovi je priložnost tudi za vzpostavitev mehanizmov izmenjave znanja v učeči organizaciji.

Zagotovo nekatera okolja uvedbo predmetne računalniške tehnologije še vedno sprejemajo zgolj kot zamenjavo dosedanjih z novimi predmetnimi programskimi orodji. Drugje se zavedajo, da uvajanje predmetne računalniške tehnologije lahko temeljno vpliva na zasnovo drugačne informacijske arhitekture. V takšnih okoljih je uvajanje predmetne računalniške tehnologije priložnost za pridobivanje dodatnih konceptualnih znanj in uvajanje sodobnejših pristopov pri načrtovanju in izdelavi računalniških rešitev.
Ali vidimo razvoj na predmetni tehnologiji zasnovanih računalniških rešitev kot grožnjo ali razvojno priložnost, je odvisno predvsem od širše ravni poznavanja dejavnikov, ki spremljajo uvajanje predmetne informacijske tehnologije. Od znanja, s katerim razpolagamo, in sposobnosti samokritične presoje v luči ocene izvedljivosti projektov zasnove in izdelave predmetno orientirane računalniške rešitve, je odvisno, ali smo to razvojno priložnost sposobni izkoristiti ali ne.

### 5.8.1 Celovitost pristopa, razvoj kulture

Uvajanje delovanja projektnih skupin ter posodabljanje znanja iz organizacije dela in vodenja je ena od priložnosti za spreminjanje načina dela v vsaki neprofitni organizaciji. Povezovanje posamičnega znanja v skupinsko in organizacijsko znanje je ob tem izziv za aktiviranje vodij, ki bi morali postati nosilci preoblikovanja togih organizacijskih oblik delovanja. Opuščanje ravnanja dela na nivoju individualnih nalog ter uvajanje skupinskega dela omogoča skupinsko izmenjavo znanja, usmerjanje v učenje na področja, kjer nam znanja primanjkuje, in skupinsko reševanje problemov. S stališča posameznika je verjetno tudi lažje pridobiti njegovo motivacijo za pripadnost projektni skupini kot organizaciji na splošno. Vzorce za spreminjanje kulture ustaljenih načinov dela lahko najdemo v modelih delovanja učeče organizacije, ki najbolj celovito obravnava pomen vnašanja novega znanja v organizacijo in pomen njegovega pretvarjanja v drugačno organizacijsko kulturo. Širše gledano je model učeče skupnosti del evropske razvojne strategije, ki združuje interese splošnega sektorja in industrije. Model združuje zainteresirane na osnovi izmenjave znanja. Naravni podaljšek oziroma sestavna komponenta takšnega modela je lahko učeča organizacija.

Računalniške rešitve so tehnični pripomočki, namenjeni podpori izvedbe aktivnosti uporabnika. Razvoj računalniških rešitev poteka na delovnem polju, kjer se srečujejo razvijalci in uporabniki, vsak s svojim znanjem, kulturo, vrednotami in izkušnjami. Z vzajemnim sodelovanjem povezujejo informacijsko komunikacijsko tehnologijo in znanje iz izbrane domene dejavnosti. Uporabnik računalniške rešitve in razvijalec način izvedbe svojega dela ter poznavanje zakonitosti uporabe informacijske tehnologije soočata v neposrednem dialogu. Pri tem se drug drugemu bolj ali manj prilagajata in iščeta primerno komunikacijsko sredstvo, s katerim bosta lahko dosegla zastavljen cilj. Ob tem uporabljata vrsto navodil, scenarijev, prototipov in drugih pripomočkov, ki so jima na voljo. Poti za uspešno medsebojno komunikacijo je vsekakor več. Pripomočki so pri tem samo sredstvo za izmenjavo znanja. Poleg uvajanja uporabe učinkovitih pripomočkov komunikacije, je potrebno poskrbeti tudi za aktivnosti usklajevanja načinov dela med udeleženci razvoja, reševanje konfliktov, izmenjavo izkušenj s sodelavci in za pridobivanje novega znanja, skratka za vodenje in usmerjanje delovanja sodelujočih akterjev.

### 5.8.2 Novi strateški cilji v dejavnosti, prožnost organizacijskih oblik

Razvoj novih računalniških rešitev ali zahteva po njihovi prenovi lahko seže tudi na področje ponovnega preverjanja strateških usmeritev delovanja podjetja. Zahteva po drugačnem sledenju strateškim ciljem lahko vpliva na preoblikovanje dosedanjih organizacijskih oblik ter delovanja in povezovanja organizacij, ki imajo isto poslanstvo v neprofitnem sektorju. V dejavnostih, kjer dobiček ni glavno merilo uspešnega razvoja podjetja, so včasih lahko bolj primerne drugačne oblike delovanja, temelječe bolj na sinergiji skupnih prizadevanj kot na tekmovalnosti.
Oblikovanje skupnega modela delovanja različnih organizacij z istim poslanstvom lahko predstavlja del trajne infrastrukture neke dejavnosti. Na takšni osnovi lahko sloni delovanje izvajalcev, koncesionarjev in administrativnih servisov.

Iskanje sinergije in načinov boljšega delovanja posameznega izvajalca v dejavnosti z isto vizijo in poslanstvom delovanja lahko tudi preseže lokalni ali regionalni okvir delovanja. Modeli za iskanje takšnih sinergij so lahko priložnost za organizacije, ki delujejo na področju neprofitnih dejavnosti. Preseganje lokalnih okvirov razmišljanja in sprejemanje izzivov pravega združevanja Evrope na osnovi vzajemnega lokalnega delovanja, ki sloni na upoštevanju preteklih zgodovinskih in kulturnih izročil, je prav gotovo tudi eden glavnih izzivov našega prihodnjega delovanja.

# 6. ZAKLJUČEK

Uporaba tehnologije v delovnem okolju na splošno predstavlja čedalje bolj nepogrešljiv pripomoček za podporo izvajanja poslovnih procesov organizacij oziroma delovanja drugih sistemov, kjer se ljudje združujejo z namenom zavestnega in načrtnega delovanja. V delovanju poslovnih subjektov sodijo tehnološke spremembe tradicionalno med glavne spodbujevalce iskanja konkurenčnih prednosti. Utemeljitev za uvajanje novih tehnologij običajno sloni na pričakovanjih izrabe prednosti, ki jih prinaša novejša tehnologija v primerjavi s staro. Običajno tudi velja, da uvajanje novih tehnologij spremljajo nova znanja, orodja in pripomočki, s katerimi olajšamo njeno uvedbo. Med podjetniško strategijo, ki si z uporabo novejše tehnologije obeta boljši tržni položaj in med konkretno uporabo tehnologije v organizaciji stoji vrsta dejavnikov, od katerih je odvisno, ali bo uvedba nove tehnologije dosegla pričakovane učinke ali ne. Z ustrezno širino in s premišljenim strateškim načrtovanjem lahko hkrati z načrtovanjem boljšega tržnega položaja načrtujemo še vrsto vzporednih učinkov, ki se navezujejo na uvajanje in uporabo nove tehnologije. Te zakonitosti veljajo tudi za uvajanje in posodabljanje informacijske tehnologije.

Informacijska tehnologija se v svojem raziskovalnem in uporabnem področju osredotoča na hranjenje in izmenjavo podatkov, ki jih uporabljamo v različnih procesih našega življenja in delovanja. Pojem informacijska tehnologija pogosto krčimo na pomenska področja, ki za potrebe našega medsebojnega razumevanja bolje odražajo njihovo specifičnost oziroma značilnosti. S pojmom predmetna informacijska tehnologija tako označujemo način snovanja informacijskih sistemov, delovanja računalniških rešitev in modeliranja podatkov (predmetov), ki izhaja neposredno iz predstavitve entitet realnega sveta. S pojmom informacijsko komunikacijska tehnologija (IKT) pa označujemo področje razvoja in izrabe komunikacijske infrastrukture v procesih izmenjave podatkov. Obe področji informacijske tehnologije vsebujeta razvojni potencial, ki v zadnjem obdobju izrazito vpliva na iskanje novih in drugačnih oblik informacijske podpore poslovnih subjektov na globalni ravni. Na osnovi smernic razvoja in zaznanih izkušenj profitnega sektorja, kjer je dinamika iskanja novih priložnosti največja, sem v nalogi iskal dejavnike, ki lahko ob uporabi omenjenih tehnologij pomembnejše vplivajo na uspeh ali neuspeh prenove informacijske podpore tudi v neprofitnem delovnem okolju. Vsako uvajanje nove tehnologije je v veliki meri povezano z zahtevami po novem znanju, zato sta obe področji informacijske tehnologije obravnavani predvsem z vidika izpostavljenega tehnološkega in organizacijskega znanja.

Kljub nekaterim značilnim razlikam, ki izhajajo iz načina regulacije njunega delovanja, ima informacijsko komunikacijska tehnologija v neprofitnem sektorju enak pomen kot v profitnem. Razlike morda lahko najdemo le v poudarkih nekaterih priložnostih, ki jih omenjena tehnologija ponuja enemu ali drugemu sektorju. Za oba sektorja so značilne priložnosti na področju vzpostavljanja novih elektronskih storitev. Morda se pri opredeljevanju pomena informacijsko komunikacijske tehnologije za profitni sektor bolj poudarjajo priložnosti v iskanju konkurenčne prednosti na trgu in možnosti za povečevanje produktivnosti, medtem ko je v neprofitnem sektorju pogosto poudarjeno zagotavljanje celovitih storitev javne uprave, povezovanje ponudnikov in odjemalcev storitev novih interesnih skupnosti ter uvajanje novih storitev, ki jim lahko pripišemo širši družbeno socialni pomen.

V celovitem pristopu uvajanja informacijsko komunikacijske tehnologije raziskave izpostavljajo vlogo strokovnjakov z interdisciplinarnimi znanji tehničnih, organizacijskih in menedžerskih znanj. Ta spoznanja so se zbistrila iz izkušenj v profitnem sektorju, ki ga tržni način delovanja

sili v stalno iskanje novih konkurenčnih prednosti. Odraz takšnega iskanja so hitrejši ciklusi spremenjenih načinov delovanja ter povezanih aktivnosti, kot so reorganizacija, prenova poslovnih procesov in posodabljanje tehnologije. Spremljanje učinkov uvajanja informacijsko komunikacijske tehnologije je skozi študije pokazalo, da gre pri izkoriščanju njenih prednosti za zahteven proces, ki pogosto terja vrsto spremljajočih, organizacijskih, menedžerskih in inštitucionalnih sprememb.

Med profitnimi in neprofitnimi organizacijami se čedalje pogosteje omenja zbliževanje načinov delovanja. To se odraža predvsem v posnemanju oblik in načinov delovanja ter izkušenj profitnega sektorja. Prenos uspešnih vzorcev delovanja v neprofitni sektor je utemeljen s koristnimi učinki, zaznanimi v profitnem sektorju. Odraža se na prevzemanju novih oblik delovanja, povezanih z organizacijo in njenim vodenjem. Na tej osnovi je upravičeno med obema sektorjema družbeno gospodarskega delovanja vzpostaviti tudi analogijo dejavnikov, zaznanih kot pomembnih, za uvajanje informacijske tehnologije in izkoriščanje priložnosti, ki jih prinaša.

Med dejavnike, pomembne za uspešno uvajanje IKT, se uvršča vloga informacijske tehnologije v strateških razvojnih programih, delovanje oddelkov za informacijsko tehnologijo v organizaciji, interdisciplinarno znanje in usposobljenost kadrov za izvajanje informacijske podpore, poznavanje informacijsko komunikacijske tehnologije, uvajanje organizacijskih sprememb, spremembe v načinu vodenja, skrb za celovito motiviranost in komunikacijo, izobraževanje zaposlenih ter pomoč zunanjih partnerjev. Med dodatne dejavnike, pomembne za izrabo potenciala predmetne informacijske tehnologije v razvoju računalniških rešitev, pa se uvrščajo zbiranje zahtev uporabnika, predmetno modeliranje, načrtovanje rešitev in uporaba modelirnih jezikov, obravnavanje razvoja računalniških rešitev kot proces ter vodenje razvoja v obliki projekta.

S stališča poznavanja tehnologije in zagotavljanja usposobljenosti kadrov za izvajanje informacijske podpore je pomembno spoznanje, da informacijsko komunikacijska tehnologija in predmetna informacijska tehnologija postajata čedalje bolj neločljivo povezani oziroma soodvisni. Lahko rečemo, da se zbližujeta v neločljivo celoto. Če se na ravni uvajanja predmetne informacijske tehnologije bolj poglobljeno ukvarjamo z zasnovo in izdelavo računalniških rešitev na osnovi entitet (predmetov), s katerimi modeliramo delovanje (procese) v realnem svetu, potem se na ravni uporabe informacijsko komunikacijske tehnologije bolj posvečamo porazdeljevanju in vzajemnemu delovanju informacijskih virov s pomočjo telekomunikacijskih povezav in protokolov komunikacije. Temeljni potencial spajanja (poznavanja) obeh tehnologij so možnosti za zagotavljanje celovite informacijske podpore v delovanju organizacije. Stična točka obeh tehnologij postajajo skupki osnovnih objektov, komponente in moduli, ki vzajemno delujejo v porazdeljenem sistemu povezanih računalniških naprav.

V profitnem sektorju je pomanjkanje interdisciplinarnega znanja informatikov, kamor poleg tehničnega sodijo tudi nova podjetniška in komunikacijska znanja, zaznano kot ovira v uvajanju informacijsko komunikacijske tehnologije. Na enak način lahko označimo njegovo pomanjkanje med neprofitnimi organizacijami. Glede na nekatere razlike v delovanju, kamor lahko štejemo tudi prožnost zaposlovanja, nagrajevanja, možnosti dodatnega usposabljanja zaposlenih in kupovanje manjkajočega znanja na trgu, lahko zaključim, da je pomanjkanje teh mehanizmov med neprofitnimi organizacijami lahko dodatna grožnja pri vzpostavljanju pogojev za boljšo izrabo potencialov informacijske tehnologije.

Na področju pridobivanja znanja je pomembna ugotovitev, ki izhaja iz modela delovanja učeče organizacije, ki pravi, da doseže znanje posameznikov pravi učinek za organizacijo, če

pridobljeno znanje ne ostaja izolirano od znanja drugih. To velja tako za organizirano pridobivanje znanja kot tudi za samoizobraževanje. Profitne organizacije imajo prožnejše mehanizme za nagrajevanje, kar lahko spodbuja tudi kroženje znanja. V neprofitnem sektorju ta primanjkljaj lahko nadomestimo z drugačno strategijo delovanja organizacije.

Dinamika iskanja novih oblik delovanja v profitnem sektorju prinaša in preizkuša nove načine delovanja, ki izkazujejo uporabno vrednost tudi za neprofitni sektor. Med takšne sodobne predloge o drugačnem delovanju organizacije, ki spreminja nekatere ustaljene vzorce, vsekakor sodi koncept učeče organizacije. Spodbujanje ustvarjanja skupinskega znanja na ravni organizacije, ustvarjanje mehanizmov za kroženje znanja, timsko delo, spremenjeni načini vodenja, delitev odgovornosti in pooblaščanje zaposlenih sodijo med osnovne dejavnike, ki v učeči organizaciji spreminjajo možnosti in priložnosti delovanja zaposlenih. Vse naštete oblike delovanja so, izhajajoč iz zahtev posodabljanja informacijske podpore, združljive z zahtevami po pridobivanju novih tehnoloških znanj, skupinskem delu, vodenjem projektov in vzpostavljanjem odgovornosti za izvedbo razvojnih procesov oziroma podprocesov.

Med koncepti delovanja učečih organizacij je izpostavljeno tudi učenje iz primerov drugih. Ta koncept pridobi na pomenu, če so znotraj organizacije vzpostavljeni mehanizmi beleženja, hranjenja in izmenjave znanja. Uporaba orodij za skupinsko delo sodi med te mehanizme. Med primere, na osnovi katerih se neprofitne organizacije lahko učijo, vsekakor sodijo tudi vzorci aktivnosti, ki jih profitne organizacije uvajajo v razvoj predmetno zasnovanih računalniških rešitev. Vzorcev razvoja predmetno zasnovanih računalniških rešitev sicer ne moremo pripisati izključno profitnim organizacijam, saj pri njihovem nastanku sodeluje tudi neprofitni sektor s področja znanstvenih in raziskovalnih ustanov, vendar veliko izkušenj o načinu razvoja računalniških rešitev prihaja prav iz profitnega sektorja.

V njem se je izoblikovalo stališče, da je razvoj računalniških rešitev preveč kompleksen in predrag proces, da bi ga lahko prepuščali neorganiziranemu delovanju. V ta namen so predlagane metodologije razvoja računalniških rešitev kot priporočene zbirke delovnih faz, pravil, tehnik, orodij, pripomočkov ter menedžmenta v razvoju sistema. To, kar daje profitnemu sektorju posebno težo pri preučevanju »njihovih« primerov razvoja računalniških rešitev, so njihove izkušnje. Praviloma so te izkušnje generatorji za izboljšavo obstoječih in osnova novih metodoloških pristopov razvoja računalniških rešitev. Na osnovi iskanja skupnih sestavin ali dejavnikov, ki jim posamezne metodologije pripisujejo večji pomen, se lahko učimo oziroma si ustvarimo boljšo sliko o njihovi vlogi in pomenu. To je pomembno tudi za neprofitne organizacije, saj je delovanje slednjih pogosto omejeno z nekaterimi dodatnimi dejavniki, ki za profitne organizacije običajno niso kritični omejitveni faktor. Sem lahko uvrstimo uporabo različnih pripomočkov, ki jih ponuja tržišče, razpoložljive kadrovske vire, lažje dosegljiva specialna znanja, potrebna v razvoju računalniške rešitve, projektni način dela, vodenje, merjenje doseženih ciljev, sočasno izvajanje podpornih procesov, kot so presojanje kakovosti, upravljanje tveganja, upravljanje ponovne rabe ali upravljanje z infrastrukturo projekta.

Preučevanje primerov razvojnih metodologij podaja jasno sporočilo o njihovi naravi. Profitne organizacije razvoj računalniške rešitve obravnavajo kot proces, ki ga je potrebno izvajati v obliki projekta. Razgradnja razvojnega procesa na podprocese povečuje preglednost in omogoča vnos mehanizmov za boljšo kakovost dela, ki ga je potrebno opraviti. Predlagana izvedba podprocesov običajno sloni na njihovi izvedbi skozi faze življenjskega razvojnega cikla, ki jih po potrebi lahko ponavljamo. V nekaterih primerih so podprocesi dodatno križani z različnimi podpornimi procesi. Za izvedbo podprocesov je običajna tudi podrobnejša delitev dela, ki sega do ravni predpisanih opravil. Ta imajo običajno večji pomen za organizacijo projekta in njegovo operativno izvedbo.

Predvsem na osnovi poznavanja zahtev po členitvi razvojnega dela, potrebnih kadrovskih virov in povezanih dejavnikov, ki jih metodologije razvoja računalniških rešitev razlagajo in predpisujejo na sistematičen in pregleden način, lahko metodologije ocenjujemo tudi kot bolj ali manj izvedljive v neprofitnem delovnem okolju. To bi moral biti osnovni namen njihovega preučevanja, na osnovi katerega lahko prilagodimo izbrano metodologijo pogojem, pod katerimi jo bomo uporabili pri svojem delu. Pravilna izbira potrebnih in izvedljivih sestavin iz nabora predlaganih metodoloških prvin, formiranje delovnih skupin izvajalcev z opredelitvijo nalog in delnih ciljev, ki jih je potrebno izvajati skozi povezane zaključene podprocese, omogočajo opredelitev osnovnih projektnih elementov, ki jih vodja projekta usklajuje v skladno celoto. Če ima pri tem na voljo mehanizme, ki spodbujajo ustvarjalni potencial in zagon izvajalcev (udeležencev) procesa, potem lahko rečemo, da razpolaga s ključnimi dejavniki uspeha v razvoju računalniške rešitve. V tej povezavi ugotavljam, da je lahko uspeh projekta, kjer uporabimo metodologijo za razvoj računalniške rešitve, v neprofitnem razvojnem okolju v veliki meri odvisen od organiziranosti in dosežene stopnje organizacijske kulture. Na osnovi te ugotovitve lahko dejavnike, ki vplivajo na uspeh razvoja računalniške rešitve, razširimo oziroma povežemo s koncepti delovanja učeče organizacije. To trditev lahko vzpostavimo tako za profitne kot neprofitne organizacije.

Izkušnje neuspelih projektov razvoja računalniških rešitev se pogosto odrazijo v nekaterih izboljšavah, kjer je uporabljena metodologija v glavnem ohranjena nespremenjena, predlagane so le spremembe procesa. Te pogosto slonijo na vnašanju različnih podpornih procesov in drugačnih vzorcih skupin aktivnosti. To prinaša tudi dopolnitve v organizacijo in način izvedbe projekta. Pri posnemanju takšnega načina delovanja oziroma izboljševanja izbranega razvojnega procesa lahko kot koristen pripomoček obravnavamo preučevanje procesnih vzorcev. Procesni vzorci so predlogi preizkušenih načinov delovanja, ki izhajajo iz splošnih problemov, na katere so razvijalci naleteli v razvojnem procesu.

Preučevani primeri metodologij razvoja so bili razpeti med dva glavna tipa, standardne in agilne metodologije. Oba imata značilnosti inkrementalnosti in iterativnosti. Po načelu inkrementalnosti poteka razvoj s postopno izvedbo novih aktivnosti, ki slonijo na rezultatih predhodnih. Skladno z načelom iterativnosti poteka zaključen obseg dela ciklično, običajno skozi faze zbiranja zahtev, načrtovanja, izvedbe in preizkušanja rezultatov.

Preučevanje vzorcev metodologij razvoja pokaže, da je zbiranje zahtev uporabnikov oziroma zbiranje zahtev sistema, ki je malo širši pojem, njihova skupna stična točka. Ta ugotovitev ni nova. Za udeležence razvoja računalniških rešitev predstavlja zbiranje zahtev sistema začeten izziv, se pa navezuje na sprejemanje nekaterih zgodnjih odločitev. Gre predvsem za način zbiranja zahtev in komunikacijo udeležencev v razvoju. V načinu zbiranja zahtev je pogosto predlagano zbiranjem zgodb uporabnikov in beleženje zahtev s primeri uporabe. V komunikaciji udeležencev razvoja je uporaba poenotenega jezika modeliranja de facto standard.

Agilne metodologije razvoja uporabljajo kot komunikacijsko sredstvo z uporabnikom (naročnikom) scenarije in delujoče prototipe računalniške rešitve. Večina standardnih metodologij razvoja pa stavi na predstavitev delovanja prihodnjega sistema s pomočjo abstraktnega modela sestavin. Seznam zbranih zahtev in model sestavin, predstavljen z različnimi pogledi, običajno predstavljata tudi komunikacijsko sredstvo, na osnovi katerega naročnik in izvajalec skleneta sporazum o skupnem razumevanju zahtev končne računalniške rešitve. Različni modeli predstavitve sistema in enotna notacija predstavitve, ki jo razumejo vsi udeleženci razvoja računalniške rešitve, so izpostavljeni kot skupna točka njihovega nadaljnjega delovanja. Uveljavljanje modeliranja in enotne notacije za izdelavo komunikacijskih sredstev

med udeleženci razvoja računalniške rešitve sodi med pomembnejše dejavnike v izvedbi razvojnega projekta.

Izkušnje razvijalcev predmetnih računalniških rešitev so pokazale, da se v začetni fazi analize zahtev sistema določene povezave med elementi pogosto ponavljajo. Njihove izkušnje in predlagane rešitve se zbirajo na enoten sistematičen način, v obliki vzorcev zahtev sistema. Uporabo kataloga teh vzorcev lahko uvrstimo med koristne dopolnilne načrtovalske pripomočke v začetnih razvojnih fazah procesa razvoja računalniške rešitve.

Standardne razvojne metodologije bolj kot agilne prisegajo na podrobnejšo predstavitev zahtev delovanja sistema in poudarjajo pomen podrobnejšega načrtovanja končne rešitve. Agilne metodologije favorizirajo izdelavo prototipov končne računalniške rešitve. Ta se osredotoča na predstavitev ogrodja delujoče računalniške rešitve in postopno dodajanje funkcionalnosti, ki poteka skozi ponovitve, v katerih končni uporabnik v dialogu z razvijalcem prispeva k njeni končni funkcionalnosti. Pristop zahteva jasno predstavo naročnika o končni funkcionalnosti in sloni na manj podrobnem vmesnem dokumentiranju zahtev in izdelavi načrta rešitve. Takšen pristop naj bi se predvidoma odrazil v hitrejši izvedbi končne rešitve. Zaradi krčenja dokumentiranja nekaterih delovnih faz so nekatere podrobnosti končne rešitve lahko slabše dokumentirane, kar ima lahko potencialne posledice za način izvedbe razvojnega projekta ali na poznejše zahteve nadgradnje ali vzdrževanja.

Neprofitne organizacije običajno tržne storitve iščejo s pomočjo javnih razpisov. V primeru, da bi končno izdelavo rešitve poverili zunanjemu izvajalcu je potrebno opredeliti način medsebojne komunikacije. V primeru uporabe določene vrste metodologije je potrebno sprejeti odločitev ali je vsakokratni prototip delujoče aplikacije primernejše komunikacijsko sredstvo med naročnikom in izvajalcem, kot abstraktni model sistema.

V splošnem v okoljih, kjer je eden od načinov delovanja organizacije sistematično zbiranje in spodbujanje kroženja ustvarjenega znanja, lahko pričakujemo podrobneje dokumentirano izvedbo procesov, natančnejšo predstavitev akterjev in načina delovanja sistema. V takšnih okoljih so lahko nekatere tradicionalne metodologije razvoja računalniških rešitev v večji sinergiji s konceptom temeljnega delovanja organizacije kot agilne.

Lahko rečemo, da je v profitnem okolju uporaba agilnih razvojnih metodologij v razcvetu. Končno ugotovitev o večji primernosti standardnih ali agilnih razvojnih metodologij v neprofitnem okolju pa je težko podati. Osnovno vodilo o izbiri ene ali druge naj bo način delovanja organizacije, tradicija timskega in projektnega dela. Pri odločitvi o načinu lastnega razvoja računalniških rešitev je pomembno vodilo tudi usposobljenost izvajalcev za izdelavo in predstavitev prototipov računalniških rešitev. V agilnih metodologijah je pomembno znanje programiranja, centraliziranо delovanje razvojne ekipe in sposobnost komuniciranja ter prenašanja zahtev uporabnika v izboljšane računalniške rešitve.

Uvajanje predmetnih računalniških rešitev in vzpostavitev vzajemnega delovanja porazdeljenih komponent predstavlja eno glavnih razvojnih področij sodobne uporabe IKT. Komunikacijska tehnologija omogoča različne načine izkoriščanja informacijskih virov, zato sodi znanje s tega področja med najpomembnejša podporna znanja pri razvoju predmetnih računalniških rešitev. To velja tako za profitno kot neprofitno delovno okolje. Poznavanje arhitekture programskih rešitev in protokolov porazdeljenega delovanja računalniških rešitev pa je treba v vsakem od teh okolij, poleg znanja programiranja, uvrstiti med temeljna znanja kadrov informatike, ki jim je poverjeno načrtovanje in izvedba končnih predmetno zasnovanih računalniških rešitev. Zelo uporaben pripomoček pri razširjanju navedenega znanja je zgledovanje po preizkušenih arhitekturnih

vzorcih načrtovanja računalniških rešitev. Med osnovne načrtovalske pripomočke, pomembne tudi za izvedbo sistema, sodijo načrtovalski vzorci programskih rešitev. Na osnovi njihovega poznavanja oziroma uporabe lahko prihranimo veliko časa pri načrtovanju in izvedbi končne računalniške rešitve.

Različna, v nalogi predstavljena spoznanja, zbrana s preučevanjem vzorcev in posameznih metodologij razvoja računalniških rešitev, so vgrajena tudi v praktičen primer predloga posodobitve informacijske podpore v ZGS. Predstavljena je organiziranost, dosežena stopnja opremljenosti z informacijsko komunikacijsko tehnologijo. Vzorčno je podan konkreten predlog delovanja s stališča zahteve, da bi razvoj predmetnih računalniških rešitev potekal v lastni izvedbi. Podanih je tudi nekaj delnih zaključkov s stališča delovanja neprofitnih organizacij.

Tudi strateško načrtovanje s pomočjo SWOT analize je predstavljeno na praktičnem primeru, in sicer pri iskanju smernic prihodnjega delovanja službe za informatiko ZGS. V analizo so vključeni dejavniki, v preučevanju primerov razvoja opredeljeni kot pomembni za razvoj predmetnih računalniških rešitev. Na osnovi analize zaznanih razmer v delovanju službe za informatiko lahko ugotovimo, da tudi nosilci razvoja obravnavajo samo znanje programiranja kot preozek okvir za celovito izrabo IKT in predmetne informacijske tehnologije. Izpostavljenih je nekaj konkretnih strateških aktivnosti, ki jih je potrebno prednostno obravnavati. V nalogi je vzorčno predlagan lasten prilagojen proces razvoja računalniških rešitev. Na osnovi predlaganega razvojnega procesa so predstavljene sestavine za organizacijo in izvedbo razvojnega projekta.

Ocenjujem, da sem v nalogi zastavljene cilje dosegel. Pri posodabljanju informacijske podpore s pomočjo predmetno zasnovanih računalniških rešitev lahko tehnologe, razvijalce, upravo in uporabnike računalniških rešitev v njihovem medsebojnem dialogu povežemo z abstraktnimi modeli delovanja sistema, za katerega načrtujemo računalniško rešitev. Pri tem imamo na voljo vrsto pripomočkov v obliki metodoloških priporočil, standardov, referenčnih modelov in preizkušenih vzorcev delovanja drugih razvijalcev.

Sistem lahko predstavimo na različnih abstraktnih nivojih bodisi kot model vzajemno povezanih procesov, bodisi kot model vzajemno delujočih objektov in akterjev. V prepoznavanju entitet sistema si lahko pomagamo z ontologijo in razvojem modela domene sistema. Abstrakcija delovanja sistema na procese, ki zahtevajo informacijsko podporo, je eden od pogledov delovanja sistema, preko katerega se lahko navežemo na podrobnejše modeliranje entitet, ki so v procesih udeležene. Sistem entitet (predmetov, udeležencev) lahko predstavimo s toliko modeli, kot je opazovalcev sistema. V načrtovanju predmetnih računalniških rešitev je racionalni pristop utemeljen z opredelitvijo nekaj ključnih pogledov na sistem. Običajno želimo predstaviti pogled na sestavine in pogled na delovanje sistema. Tretji pomemben pogled je predstavitev razporeditve fizičnih povezav med komponentami, ki omogočajo delovanje sistema. Statični pogled na sistem lahko kaže vse elemente sistema ali pa so nekatere skupine elementov združene v okvir modulov, v katerih se izvaja funkcionalnost, prilagojena izbranim procesom. Modeliranje na več ravneh abstrakcije in uporaba enotne notacije za predstavitev modelov predstavljajo temeljno povezovalno sredstvo udeležencev v razvoju računalniških rešitev.

Predmetni razvoj računalniških rešitev ponuja možnosti tudi za integracijo dosedanjih rešitev. Osnovo predstavlja izraba IKT in modeliranje računalniških rešitev na način, ki bo dosedanje rešitve obravnaval kot delne rešitve ali kot komponente končne celovite informacijske podpore v delovanju organizacije. Področje integracije obstoječih rešitev bo morda zahtevalo predelavo vmesnikov in nekatere prilagoditve uporabljenega podatkovnega (objektnega) modela sistema. V splošnem je iskanje in uporaba primernih komponent, ki že vsebujejo potreben del

funkcionalnosti, eden od današnjih trendov snovanja predmetno zasnovanih računalniških rešitev. Na tem področju lahko zaznamo prizadevanja po poenotenju in standardizaciji. Paleto standardnih in medsebojno združljivih konceptov vzajemnega delovanja komponent lahko prevzamemo tudi z izbiro enotne razvojne platforme. Ključni pomen v porazdeljenem delovanju računalniških rešitev ima način pozivanja izvedbe oddaljenih metod in uporaba nekaterih prilagojenih protokolov. Osnove delovanja porazdeljenih aplikacij sodijo v temeljno znanje, potrebno za povezovanje obstoječih in novih porazdeljenih računalniških rešitev.

Med temeljna spoznanja naloge, pomembna predvsem za neprofitni sektor, sodi ugotovitev, da je za uspešno izvedbo razvoja predmetne računalniške rešitve nujno vzpostaviti mehanizme izmenjave znanja med udeleženci razvoja. Sredstvo za izmenjavo znanja so lahko različni abstraktni predmetni modeli, ki predstavljajo pogled na sistem ter uporaba enotnega jezika modeliranja. Mehanizem, ki omogoča zbiranje in kroženje znanja ter povezanost udeležencev v enotno računalniško mrežo, so različna orodja za skupinsko delo. Okolje, ki uvaja koncepte delovanja za pridobivanje novega znanja udeležencev razvoja in spodbuja različne načine njegove izmenjave, pa predstavlja učeča organizacija.

Na osnovi doseženih spoznanj lahko podam tudi sklepno misel naloge. V načinu organiziranega delovanja kot ga predvideva koncept učeče organizacije in v načrtovanem pristopu razvoja predmetne računalniške rešitve, ki ga obravnavamo kot proces in vodimo kot projekt, so skriti osnovni dejavniki uspešnega razvoja računalniške rešitve. V poznavanju zakonitosti predmetne informacijske tehnologije in njenem povezovanju z možnostmi, ki jih nudi informacijsko komunikacijska tehnologija, pa so skriti osnovni dejavniki za vzpostavitev celovite informacijske podpore v organizaciji.

# LITERATURA


1. Ambler Scott W.: Building Object Applications That Work. Cambridge University Press, 1998, 476 str.
2. Ambler Scott W.: The Object Primer. Cambridge University Press, 1998, 248 str.
3. Ambler Scott W.: Process Patterns, Cambridge University Press, 1998, 549. str.
4. Ambler Scott W.: Why Data Models Shouldn't drive Object Models (And vice versa), [URL: http://www.agiledata.org/essays/drivingForces.html ], januar 2008.
5. Avison David E., Fitzgerald Guy: Information Systems Development - Methodologies, Techniques and Tools, 2nd edition. McGraw Hill, USA, 1996, 505 str.
6. Avison David E., Fitzgerald Guy: Where Now for Development Methodologies. Communications of the ACM, January, 46: 1, 2003, str. 79-82.
7. Avison David, Adams Carl: Dangers inherent in the use of techniques. Identifying framing influences, Information Technology & People, februar 2003, Vol. 16 (no. 2), str. 203-34, Emerald Group Publishing Limited.
8. Avgeriou Paris, Zdun Uwe: Architectural Patterns Revisited - A pattern language. Proceedings of 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, Germany, July 2005: str. 1-39.
9. Barker Iain: What is information architecture. KM Column, [URL: http://www.steptwo.com.au/papers/kmc_whatisinfoarch/index.html], april 2005.
10. Boehm Barry W.: A spiral model of software development and enhancement. ACM Sigsoft Software engineering notes, Vol 11, Issue 4, avgust 1986, str. 14-24.
11. Bruegge Bernd, Dutoit Allen H.: Object-Oriented Software Engineering, Using UML Patterns and Java. Pearson Prentice Hall, 2004, 762 str.
12. Bygstad Bendik, Munkvold Bjorn Erik: Software engineering and IS implementation Research, An Analytical Assessment of Current SE Frameworks as Implementation Strategies. Information systems development, Kluwer/plenum press, NY, 2003, str. 227-239.
13. Corcho Oscar, Gomez-Perez Asuncion: A Roadmap to Ontology Specification, [URL: http://www.cs.man.ac.uk/~ocorcho/documents/ekaw00_CorchoGomezPerez.pdf], oktober, 2000.
14. Čuš Franci.: Analiza konkurenčnih prednosti. Maribor, 1997, 213 str.
15. Damij Talib: Poslovna informatika. Ljubljana, Ekonomska fakulteta, 2000, 204 str.
16. Damij Talib: Tabular aplication development for information systems, an object-oriented methodology. Springer, New York, 2001, 190 str.
17. Dimovski Vlado, Penger Sandra, Škerlavaj Miha, Žnidaršič Jana: Učeča se organizacija, ustvarite podjetje znanja. GV Ljubljana, 2005, 387 str.
18. Eddon Guy, Eddon Henry, Programming Components with MS Visual Basic 6.0, 2nd edition. Redmond, USA, 1998, 389 str.
19. Eriksson Hans-Erik, Penker Magnus: Business Modeling with UML, Business Patterns at Work. John Wiley& Sons, New York, 2000, 459 str.
20. Ferlin Franc: Primerjava organiziranosti gozdarstev in številčnosti javno gozdarskega kadra srednjeevropskih držav s stanjem v Sloveniji, [URL:http://www.gozdis.si/departments/silviculture/Primerjava%20gozdarskih%20kadrov%20evropskih%20drzav.pdf ], maj 2004.
21. Gamma, Erich; Richard Helm, Ralph Johnson and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented software. Addison-Wesley, 1995, 395 str.
22. Garlan David, Shaw Mary: An introduction to Software Architecture. School of Computer Science Carneige Mellon University, Pittsburgh, [URL: http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf ], januar 1994.

23. Grad Janez, Dacar France: Podatkovne strukture, skripta. Ekonomska fakulteta, Ljubljana, 1996, 109 str.
24. Grad Janez, Jaklič Jurij: Baze podatkov. Ljubljana, Ekonomska fakulteta. 1996, 254 str.
25. Gradišar Miro, Resinovič Gortan: Informatika v poslovnem okolju. Ljubljana, Ekonomska fakulteta, 2001, 508 str.
26. Graham Ian, Henderson-Sellers Brian, Younessi Houman: The OPEN process Specification. ACM Press New York, Addison-Wesley, 1999, 314 str.
27. Gruber Thomas R.: A Translation Approach to Portable Ontology Specification. Knowledge Acquisition archive vol 5: 199-220, june 1993.
28. Halpin Terry: Object Role Modeling (ORM/NIAM). Microsoft Corporation USA, [URL: http: //www.orm.net/pdf/springer.pdf], 1998.
29. Hay David C.: Data Model Patterns-A metadata map. Elsevier, San Francisco, 2006, 406 str.
30. Henderson-Sellers Brian: The OPEN Framework for Enhancing Productivity. Software IEEE, Vol 17, Issue 2, March-April 2000, str. 53-58.
31. Irwin Gretchen, Turk Daniel: An Ontological Analysis of Use Case Modeling Grammar, [URL: http: //www. biz.colostate.edu/faculty/gretcheni/JAIS04-122.pdf], januar 2005.
32. Jacobson Ivar, Booch Grady, Rumbaugh James: The Unified Modeling Language Reference Manual. Massachusetts, 1999, 550 str.
33. Jacobson Ivar, Booch Grady, Rumbaugh James: The Unified Modeling Language User Guide. Massachusetts, 1999, 482 str.
34. Jacobson Ivar, Booch Grady, Rumbaugh James: The Unified Software Development Process. Addison Wesley, Massachusetts, 1999, 463 str.
35. Kovačič Andrej, Bosilj-Vukšić Vesna: Management poslovnih procesov, Prenova in informatizacija poslovanja. Ljubljana: GV založba, 2005, 487 str.
36. Kruchten Philippe: The Rational Unified Process. Addison Wesley, Massachusetts, 1999, 255 str.
37. Marshall Chris: Enterprise Modeling with UML. Addison-Wesley, Reading, MA 2000, 288 str.
38. Martin James: Strategic Information Planning Methodologies, 2nd Edition. Prentice Hall, 1989, 328 str.
39. Martin James, Odell James J., Object-Oriented Analysis and Design, Prentice-Hall, 1992, 515 str.
40. Mesec Bojana: Življenjski ciklus neprofitne organizacije. Fakulteta za socialno delo, [URL: http: //www.fsd.si/gradivo/4.%20letnik/Neprofitne%20organizacije/], Neprofitne organizacije.doc, februar 2006
41. McNamara Carter: Strategic planning (in nonprofit or forprofit organizations). [URL: http://www.managementhelp.org/plan_dec/str_plan/str_plan.htm#anchor320170 ], 1997-2008.
42. McNay Heather: Information architecture - visual displays. Professional Communication Conference, 2003, IPCC 2003. Proceedings, IEEE International 21-24 sept. 2003, 5 str.
43. Mei Hong: A complementary approach to requirements engineering-software architecture orientation. ACM SIGSOFT Software Engineering Notes, Vol. 25, Issue 2, March 2000, str. 40-45.
44. Medved Mirko: Standardi za opravljanje nalog javne gozdarske službe. Raziskovalni projekt, 2002, 128 str.
45. Microsoft: Visual Basic Component Tools Guide, 1991-1997, 796 str.
46. Mikulič Vid: Strategija informatike ZGS. interno delovno gradivo, 2005.
47. Milićev Dragan: Objektno orientisano modelovanje na jeziku UML. Mikro knjiga, 2001, 254 str.

48. Montilva Jonas A., Hamzan Khalid, Gharawi Mohammed: The Watch Model for Developing Business Software in Small and Midsize Organizations. Proc. of the IV World Multiconference on Systemics, Cybernetics and Informatics - SCI´2000. Vol. XII. Orlando, Florida, July (2000), 263-268.
49. Možina Stane, Kavčič Bogdan, Tavčar Mitja I., Pučko Danijel, Ivanko Štefan, Lipičnik Bogdan, Gričar Jože, Repovž Leon, Vizjak Andrej, Vahčič Aleš, Rus Veljko, Bohinc Rado: Management, nova znanja za uspeh. Didakta, Radovljica, 2002, 867 str.
50. Mursu Anja, Luukonen Irmeli, Toivanen Marika, Korpela Mikko: Activity theory in information systems research and practice, theoretical underpining for an information systems development model. Informationresearch Vol 12, no 3, April 2007, [URL: http://informationr.net/ir/12-3/paper311.html].
51. Noy Natalya F., McGuiness Deborah L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report, marec 2001, 25. str.
52. Osterwalder Alexander, Pigneur Yves, Tucci Chrispopher L.: Clarifying Business Models: Origins, Present, And Future Of The Concept. Communications of AIS, Volume 15, maj 2005, 43 str.
53. Parviainen Paivi, Hulkko Hanna, Kaariainen Jukka, Takalo Juha, Tihinen Maarit: Requirements engineering - Inventory of technologies. VTT Publications, 2003, 106 str.
54. Peterlin Judita: Razvoj voditeljstva v učeči organizaciji. EF, magistrsko delo, marec 2007, 65 str.
55. Pras Aiko, Schoenwaeldner Juergen: On the Difference between Information Models and Data Models. RFC 3444, [URL: http://tools.ietf.org/html/rfc3444 ], januar 2003.
56. Pugh Kenneth: Interface Oriented Design. The Pragmatic Bookshelf, USA, 2006, 215 str.
57. Royce Winston W.: Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on software engineering, Monterey, California, USA, 1987, str. 328-338 (Reprinted from Proceedings, IEEE Wescon, TRW, avgust 1970).
58. Runeson Per, Greberg Peter: Extreme Programming and Rational Unified Process : Contrasts or Synonyms?, [URL: http: //serg.telecom.lth.se/research/publications/docs/282_runeson_XP_RUP.pdf], 5. julij 2000.
59. Schikorr Uwe: EU Twinning-Light Sl2004-IB-AG-05, Activity 3: Analysis of soft- and hardware: content and information, 2007, 27 str.
60. Schmidt Douglas C., Buschmann Frank: Patterns, Frameworks and Middleware: Their SynergisticRelationsship. IEEE Computer Society, 25th International Conference on SoftwareEngineering, Portland Oregon, (2003), 694-704.
61. Sowa John F.: Knowledge representation: Logical, Philosophical, and Computational foundations. Pacific Grove, CA: Brooks/Cole Publishing, 2000, 594 str.
62. Stare Metka, Bučar Maja: Učinki Informacijsko komunikacijskih tehnologij. Založba Hermina Kranjc, FDV, 2005, 186 str.
63. Šumrada Radoš: Tehnologija GIS. FAGG, Ljubljana, 2005, 330 str.
64. Taylor A. David: Business Engineering with Object Technology. Wiley and Sons, 1995, 188 str.
65. Teng James T. C., Kettinger William J.: Business Process Redesign and Information Architecture: Exploring the relationships. Data Base Advances vol. 26, No 1, February 1995, (str. 30-42).
66. Thai Vinh Tuan, O'riain Sean, Davis Brian, O'Sullivan David: Personalized Question Answering: A Use Case for Business Analysis. Proceedings of the 1st International Workshop on Applications and Business Aspects of the Semantic Web, Workshop at 5th International Semantic Web Conference, 2006, 13 str.

67. Thompson Arthur A., Strickland Alonzo J.: Strategic management: concepts and cases, 11 th edition, Boston, Irwin/McGraw-Hill, 1999, 683 str.
68. Vidmar Tone: Informacijsko komunikacijski sistem. Založba Pasadena d.o.o., 2002, 841 str.
69. Vinoski Steve: "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments," IEEE Communications Magazine, vol. 14, no. 2, February 1997, [URL: http: //www.cs.wustl.edu/~schmidt/PDF/vinoski.pdf]
70. Withall Stephen: Software Requirement Patterns. Microsoft Press, 2007, 366 str.
71. Wang Guijun, Fung Casey K.: Architecture paradigms and Their Influences and impacts on component-based software systems. System sciences 2004, Vol. Issue 5-8, januar 2004, Proceedings of the 37th Hawaii International Conference on System Sciences, 10 str.
72. Wieringa Roel: A Survey of Structured and Object-Oriented Software Specification Methods and Techniques. ACM Computing Surveys, vol. 30, Issue 4, december 1998, str. 459-527.
73. Yourdon Edward, Whitehead Katharine, Thomann Jim, Oppel Karin, Nevermann, Peter: Mainstream Objects: An Analysis and Design Approach for Business. Yourdon Press Upper Saddle River, NJ, 1995, 331 str.
74. Zhang Charles, Jacobsen Hans-Arno: Refactoring middleware with aspects, Parallel and distributed systems. IEEE Transactions on, vol. 14, Issue 11, Nov. 2003, str. 1058-1073.
75. Zowghi Didar, Firesmith Donald G., Henderson-Sellers Brian: Using the OPEN process framework to produce a situation-specific requirements engineering method. Proceedings of SREP'05, Paris, France, 29-30. avgust 2005, str. 59-74.

# VIRI


1. Agile Adoption Survey 2007, [URL: http://www.ambysoft.com/surveys/agileMarch 2007.html] , januar 2008.
2. Bridging the Gap between Business Models and System Models, [URL: http: //www.cems.uwe.ac.uk/~sjgreen/MohammedOdehEtAlFiguresAndtext.pdf ].
3. Business Rules and Object Role Modeling, [URL: http: //www.orm.net/pdf/dppd.pdf], oktober 1996.
4. IDA-conference »eGovernment in the service of European citizens and enterprises - June 2001, IDA Sandhamn Conference Conclusions, [URL: http://ec. europa.eu/idabc/en/ document/316], 13.-14. 6. 2001.
5. IEEE Recomended practice for architecture description of software-intensive systems, IEEE Std 1471-2000, 21. sept. 2000, 23 str.
6. IEEE Std 1471 Frequently Asked Questions, [URL: http: //www.iso-architecture.org/ieee-1471/ieee-1471-faq.html], ver, 5.0, 19. julij 2007.
7. IEEE Standard for Conceptual modeling language syntax and semantics for IDEF1X97 (IDEFobject), Std. 1320.2-1998, 25. junij 1998, 303 str.
8. IT Project Success Rates Survey August 2007, [URL: http: //www.ambysoft.com/surveys/success2007.html] , januar 2008.
9. Manifesto for Agile Software Development, [URL: http: //agilemanifesto.org/], januar 2008.
10. MDA Guide version 1.0.1, [URL: http: //www.omg.org/docs/omg/03-06-01.pdf], 12. junij 2003.
11. Ministrska konferenca eGovernment, deklaracija sprejeta 29. 11. 2001, [URL: http: //mid.gov.si/mid/mid.nsf/V/KE45C522A20516EADC1256C0C0073B4AE/$file/eGov_min_deklaracija_bru2001_si.pdf].
12. Model-Driven Architecture: Vision, Standards And Emerging Technologies [URL: http://www.cwmforum.org/Model-Driven+Architecture.pdf], april, 2001.
13. OASIS: Reference model for service oriented Architecture 1.0, Comittee Specification1, [URL: http: //www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf].
14. OGC Reference Model: Open gepospatial consortium Inc., 16. sept. 2003 [URL: http: //portal.opengeospatial.org/files/?artifact_id=3836], 2. august 2006.
15. Unified process, [URL: http://en.wikipedia.org/wiki/Unified_process], januar 2008.
16. Unified Modeling Language, [URL: http://en.wikipedia.org/wiki/Unified_Modeling_Language], januar 2008.
17. Web Service Design, [URL: http: //www.serviam.se/serviam2/ServiamDocuments/LIT-03%20Service%20Design.pdf?POSTNUKESID=ff37f1f7e55f6c0e94a38e93a70422bf], 2. december, 2004.
18. When is a document agile, [URL: http://www.agilemodeling.com/essays/agileDocumentation.htm#WhenIsADocumentAgile], januar 2008.
19. Zakon o gospodarskih javnih službah (Uradni list RS, št. 32/93).
20. Zakon o gozdovih (Uradni list RS, št. 30/93,110/2007).
21. Zakon o zavodih (Uradni list RS, št. 12/91).