2012 * jul/avg/sep * letnik XX

3 UPORABNA INFORMATIKA

20let

Izpitni centri ECDL

ECDL (European Computer Driving License), ki ga v Sloveniji imenujemo evropsko računalniško spričevalo, je standardni program usposabljanja uporabnikov, ki da zaposlenim potrebno znanje za delo s standardnimi računalniškimi programi na informatiziranem delovnem mestu, delodajalcem pa pomeni dokazilo o usposobljenosti. V Evropi je za uvajanje, usposabljanje in nadzor izvajanja ECDL pooblaščena ustanova ECDL Fundation, v Sloveniji pa je kot član CEPIS (Council of European Professional Informatics) to pravico pridobilo Slovensko društvo INFORMATIKA. V državah Evropske unije so pri uvajanju ECDL močno angažirane srednje in visoke šole, aktivni pa so tudi različni vladni resorji. Posebno pomembno je, da velja spričevalo v 148 državah, ki so vključene v program ECDL. Doslej je bilo v svetu izdanih že več kot 11,6 milijona indeksov, v Sloveniji več kot 17.000, in podeljenih več kot 11.000 spričeval. Za izpitne centre v Sloveniji je usposobljenih 11 organizacij, katerih logotipe objavljamo.

AGORA

IZOBRAŽEVANJE
INFORMACIJSKE STORITVE

SREDNJA GRADBENA, GEODETSKA
IN EKONOMSKA ŠOLA LJUBLJANA
1000 Ljubljana, Dunajska 102

LJUDSKA UNIVERZA
MURSKA SOBOTA

ISER

ŠOLSKI CENTER VELENJE

ŠOLSKI CENTER
Novo mesto

spin

LJUDSKA UNIVERZA CELJE

UPI LJUDSKA UNIVERZA ŽALEC
šola prijaznih ljudi

Micro Team

U P O R A B N A
I N F O R M A T I K A
2012 ŠTEVILKA 3 JUL/AVG/SEP LETNIK XX ISSN 1318-1882

Uvodnik

Znanstveni prispevki

Strokovni prispevki

Informacije

U P O R A B N A
I N F O R M A T I K A

2012 ŠTEVILKA 3 JUL/AVG/SEP LETNIK XX ISSN 1318-1882

Ustanovitelj in izdajatelj
Slovensko društvo INFORMATIKA
Litostrojska cesta 54, 1000 Ljubljana

Predstavnik
Niko Schlamberger

Odgovorni urednik
Jurij Jaklič

Gostujoči urednik te številke
Vladislav Rajkovič

Prelom in tisk
Boex DTP, d. o. o., Ljubljana

Naklada
600 izvodov

Naslov uredništva
Slovensko društvo INFORMATIKA
Uredništvo revije Uporabna informatika
Litostrojska cesta 54, 1000 Ljubljana
www.uporabna-informatika.si

Revija izhaja četrtletno. Cena posamezne številke je 20,00 EUR. Letna naročnina za podjetja 85,00 EUR, za vsak nadaljni izvod 60,00 EUR, za posameznike 35,00 EUR, za študente in seniorje 15,00 EUR. V ceno je vključen DDV.

Revija Uporabna informatika je od številke 4/VII vključena v mednarodno bazo INSPEC.

Revija Uporabna informatika je pod zaporedno številko 666 vpisana v razvid medijev, ki ga vodi Ministrstvo za kulturo RS.

Revija Uporabna informatika je vključena v Digitalno knjižnico Slovenije (dLib.si).

Vabilo avtorjem

V reviji Uporabna informatika objavljamo kakovostne izvirne članke domačih in tujih avtorjev z najširšega področja informatike v poslovanju podjetij, javni upravi in zasebnem življenju na znanstveni, strokovni in informativni ravni; še posebno spodbujamo objavo interdisciplinarnih člankov. Zato vabimo avtorje, da prispevke, ki ustrezajo omenjenim usmeritvam, pošljejo uredništvu revije po elektronski pošti na naslov ui@drustvo-informatika.si.

Avtorje prosimo, da pri pripravi prispevka upoštevajo navodila, objavljena v nadaljevanju ter na naslovu http://www.uporabna-informatika.si.

Za kakovost prispevkov skrbi mednarodni uredniški odbor. Članki so anonimno recenzirani, o objavi pa na podlagi recenzij samostojno odloča uredniški odbor. Recenzenti lahko zahtevajo, da avtorji besedilo spremenijo v skladu s priporočili in da popravljeni članek ponovno prejmejo v pregled. Uredništvo pa lahko še pred recenzijo zavrne objavo prispevka, če njegova vsebina ne ustreza vsebinski usmeritvi revije ali če članek ne ustreza kriterijem za objavo v reviji.

Pred objavo članka mora avtor podpisati izjavo o avtorstvu, s katero potrjuje originalnost članka in dovoljuje prenos materialnih avtorskih pravic. Nenaročenih prispevkov ne vračamo in ne honoriramo. Avtorji prejmejo enoletno naročnino na revijo Uporabna informatika, ki vključuje avtorski izvod revije in še nadaljnje tri zaporedne številke.

S svojim prispevkom v reviji Uporabna informatika boste prispevali k širjenju znanja na področju informatike. Želimo si čim več prispevkov z raznoliko in zanimivo tematiko in se jih že vnaprej veselimo.

Uredništvo revije

Navodila avtorjem člankov

Članke objavljamo praviloma v slovenščini, članke tujih avtorjev pa v angleščini. Besedilo naj bo jezikovno skrbno pripravljeno. Priporočamo zmernost pri uporabi tujk in – kjer je mogoče – njihovo zamenjavo s slovenskimi izrazi. V pomoč pri iskanju slovenskih ustreznic priporočamo uporabo spletnega terminološkega slovarja Slovenskega društva Informatika Islovar (www.islovar.org).

Znanstveni članek naj obsega največ 40.000 znakov, strokovni članki do 30.000 znakov, obvestila in poročila pa do 8.000 znakov.

Članek naj bo praviloma predložen v urejevalniku besedil Word (*.doc ali *.docx) v enojnem razmaku, brez posebnih znakov ali poudarjenih črk. Za ločilom na koncu stavka napravite samo en prazen prostor, pri odstavkih ne uporabljajte zamika.

Naslovu članka naj sledi za vsakega avtorja polno ime, ustanova, v kateri je zaposlen, naslov in elektronski naslov. Sledi naj povzetek v slovenščini v obsegu 8 do 10 vrstic in seznam od 5 do 8 ključnih besed, ki najbolje opredeljujejo vsebinski okvir članka. Pred povzetkom v angleščini naj bo še angleški prevod naslova, prav tako pa naj bodo dodane ključne besede v angleščini. Obratno velja v primeru predložitve članka v angleščini.

Razdelki naj bodo naslovljeni in oštevilčeni z arabskimi številkami.

Slike in tabele vključite v besedilo. Opremite jih z naslovom in oštevilčite z arabskimi številkami. Vsako sliko in tabelo razložite tudi v besedilu članka. Če v članku uporabljate slike ali tabele drugih avtorjev, navedite vir pod sliko oz. tabelo. Revijo tiskamo v črno-beli tehniki, zato barvne slike ali fotografije kot original niso primerne. Slik zaslonov ne objavljamo, razen če so nujno potrebne za razumevanje besedila. Slike, grafikoni, organizacijske sheme ipd. naj imajo belo podlago. Enačbe oštevilčite v oklepajih desno od enačbe.

V besedilu se sklicujte na navedeno literaturo skladno s pravili sistema APA navajanja bibliografskih referenc, najpogosteje torej v obliki: (Novak & Kovač, 2008, str. 235). Na koncu članka navedite samo v članku uporabljeno literaturo in vire v enotnem seznamu po abecednem redu avtorjev, prav tako v skladu s pravili APA. Več o APA sistemu, katerega uporabo omogoča tudi urejevalnik besedil Word 2007, najdete na strani http://owl.english.purdue.edu/owl/resource/560/01/.

Članku dodajte kratek življenjepis vsakega avtorja v obsegu do 8 vrstic, v katerem poudarite predvsem strokovne dosežke.

Spoštovane bralke in spoštovani bralci,

tokratna številka revije vsebuje izbor dopolnjenih in razširjenih prispevkov z letošnjega 19. posveta Dnevi slovenske informatike. Srečanje je povezovala rdeča nit Ustvarimo nove rešitve. Vprašamo se lahko: Zakaj nove rešitve? in Kako do njih?

Na različnih koncih sveta se soočamo s človekovim nezadovoljstvom. Gre za nezadovoljstvo v zvezi z demokracijo, voditelji, porazdelitvijo kapitala, zaposlitvenimi možnostmi in ne nazadnje v zvezi s preživetjem v okolju, ki ga imamo. Rešitve, ki predlagajo nenehno in neomejeno rast, pa so vse manj verjetne, saj smo v vseh pogledih omejeni z viri.

Govorimo o »krču« družbe in posameznika tako v duhovnem kot materialnem smislu. Pri tem krč tehnologije, v katero v glavnem spadajo tudi informatika, računalništvo in komunikacije, ni nobena izjema. Svet in posameznik sta v zagati, s katero se soočamo vsi. Starih modelov in rešitev ne gre le popravljati, potrebujemo predvsem nove inovativne rešitve.

Neodvisni svetovalci vidijo informatiko kot motor, ki lahko izvleče človeka in svet iz sedanjega položaja ter potegne gospodarstvo v nov pozitivni ciklus. Pomen, ki ga EU pripisuje informatiki, je izkazan v DAE 2020. Ta je šla mimo nas skoraj neopaženo, čeprav se strinjamo, da je za ustvarjenje novih rešitev nujna jasna razvojna strategija.

Pogosto slišimo, da sta za nove rešitve potrebni ustvarjalnost in inovativnost. V metodološkem pogledu potrebujemo razvojnoraziskovalni pristop, ki je lasten človeku, saj se ta v novih nepredvidljivih okoliščinah znajde bolje kot računalnik. V čem je težava? Vse prevečkrat novo idejo zavrnemo že v kali ali pa jo pretirano poveličujemo. V splošnem je smiselno ideje preverjati sistematično, npr. s kritično analizo prototipne rešitve.

Koren problema je človek sam. Naše dojemanje sveta in odnosi med ljudmi dohitevajo tehnologijo z zamikom. Z drugimi besedami lahko rečemo, da človek sam pri sebi zaostaja za tehnološkimi inovacijami. Tehnološke inovacije se dogajajo nekako zunaj nas. Inovacija poslovnega modela pa je notranja stvar posameznika, podjetja, ustanove, korporacije, države itn. Kar se tiče človeka in vrednot, je jasno (Arrowov izrek o nemogočem), da ni mehanizma, ki bi pripeljal do optimalnih odločitev brez dogovarjanja v smislu iskanja konsenza, ki vsebuje popuščanje z vseh strani. Ko pri rešitvah iščemo najboljšo, ni nič narobe, če mislimo najprej nase, vendar ne smemo pozabiti tudi na druge, ki se jih tiče naša rešitev.

Vladislav Rajkovič,
gostujoči urednik

Vabilo k prijavi prispevkov

ZA JUBILEJNO, 20. KONFERENCO
DNEVI SLOVENSKE INFORMATIKE

V Portorožu bo **15. do 17. aprila 2013**
potekala že 20. konferenca
Dnevi slovenske informatike.

Rdeča nit DSI 2013 je »*Dvajset let pozneje*«. Konferenca je v teh letih postala priložnost za predstavitev dosežkov, izmenjavo idej, primerjavo s svetovnimi dognanji informatike, odkrivanje poslovnih priložnosti in ne nazadnje tudi za neformalno srečevanje v prijaznem okolju slovenskega Primorja. Tako bo tudi v letu 2013.

Vabimo vas k predstavitvi svojih dosežkov in spoznanj v obliki prispevkov na konferenci. Rok za prijavo prispevka v obliki povzetka v dolžini do pol strani A4 je 18. januar 2013. Več informacij o konferenci in oddaji prispevkov bo v kratkem na voljo na spletni strani **www.dsi2013.si.**

Veselimo se srečanja z vami na 20. konferenci DSI.

Slovensko društvo Informatika

ArchiMate® for Integrated Modelling Throughout the Architecture Development and Implementation Cycle

Henk Jonkers, Dick A. C. Quartel, Henry M. Franken
BiZZdesign, Enschede, the Netherlands
{h.jonkers, d.quartel, h.franken}@bizzdesign.nl

Abstract

The ArchiMate standard offers an integrated language for enterprise architecture modelling. It allows for the description and visualization of different architecture domains, as well as their underlying relationships and dependencies. Since its adoption as a standard of The Open Group, the international interest in ArchiMate has been growing rapidly. ArchiMate complements TOGAF, the standard of The Open Group for developing enterprise architectures. To provide modelling support throughout the architecture development and implementation cycle as defined by TOGAF, Version 2.0 of ArchiMate includes two extensions to the original ArchiMate core language: a *Motivation* extension and an *Implementation and Migration* extension. The aim of this paper is to provide the reader with an introduction to the ArchiMate language, highlighting the new features of Version 2.0, and to illustrate how the language can be used in the different phases of the TOGAF architecture development cycle.
Keywords: Enterprise Architecture, modelling, ArchiMate, TOGAF.

Izvleček

ArchiMate – integrirano modeliranje podjetniških arhitektur skozi razvojni in implementacijski cikel

ArchiMate ponuja integrirani jezik za pristop k modeliranju podjetniških arhitektur. Omogoča opisovanje in vizualiziranje različnih arhitekturnih domen kot tudi odnosov in odvisnosti v njihovem ozadju. Mednarodno zanimanje za ArchiMate močno narašča, odkar ga je The Open Group privzel kot standard. ArchiMate namreč dopolnjuje TOGAF, ki je standard skupine The Open Group za razvijanje podjetniških arhitektur. Da bi zagotovili podporo modeliranju skozi celoten razvoj arhitekture in ciklus implementacije, kot ga določa TOGAF, vsebuje ArchiMate verzija 2.0 dve razširitvi izvirnega osnovnega jezika ArchiMate: motivacijsko razširitev ter implementacijsko in migracijsko razširitev. Namen tega prispevka je uvesti bralca v jezik ArchiMate, tako da osvetlimo nove lastnosti verzije 2.0 in prikažemo, kako lahko jezik uporabimo v različnih fazah razvojnega cikla arhitekture TOGAF.
Ključne besede: arhitektura podjetja, modeliranje, ArchiMate, TOGAF.

1 INTRODUCTION

Within larger organizations, one can typically find various architecture domains, such as: organizational structures, products, business processes, information systems, applications and technical infrastructure. Traditionally, each architecture domain employs specific models and visualizations, which simplifies communication, discussion and analysis *within* the domain. However, the relations between these different domains are in many cases unclear. Moreover, these domains tend to (at least partially) overlap.

The ArchiMate language for enterprise architecture modelling language has been developed with the aim to provide a uniform representation for enterprise architecture (EA) descriptions (Lankhorst et al., 2009; The Open Group, 2012). It offers an integrated architectural approach by which organizations can describe and visualize different architecture domains, as well as their underlying relationships and dependencies. ArchiMate provides a unified way to model enterprise architectures, while integrating the various domains and describing them in an easily readable way, as illustrated in Figure 1. In addition, a distinction is made between a business layer, an application layer, and a layer with the underlying (IT) technical infrastructure.

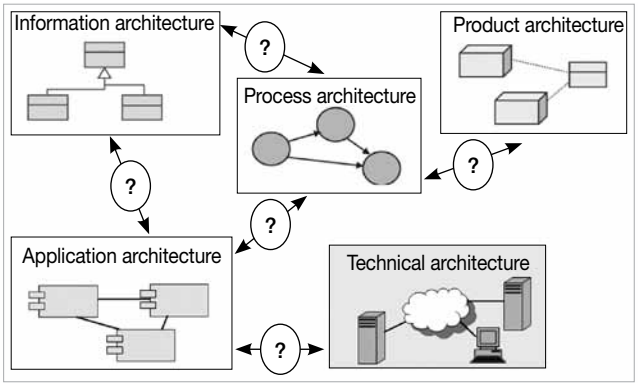


Figure 1. **Integration of architectural domains**

Since ArchiMate is positioned at the level of enterprise architecture, this also implies that the ArchiMate language does not provide the level of detail one would typically find in languages used at the »design level« (Lankhorst, Proper, & Jonkers, 2010). For example, while ArchiMate features concepts such as business event and junction, it does not provide the rich detailed set of gateways, et cetera as offered by a language such as BPMN (Object Management Group, 2008). Similarly, in contrast to languages such as UML (Object Management Group, 2005), it does not provide concepts to model the details of software applications. At the same time, refinement/abstraction mechanisms can be used to maintain the connection between, e.g., a BPMN or UML model and an ArchiMate model (Lankhorst et al., 2009).

The ArchiMate language has been designed in a structured way, by defining a generic structure that is made specific for the different architectural layers (as will be explained in Section III). Also, ArchiMate has a limited set of relation types that are used throughout the language. Finally, ArchiMate provides a standard graphical notation for the modelling concepts and relations.

The purpose of this paper is to provide the reader with an introduction to the ArchiMate language, highlighting the new features of Version 2.0, and to illustrate how the language can be used in the different phases of the TOGAF architecture development cycle. The remainder of this paper is structured as follows. In Section II, we introduce the ingredients of an integrated enterprise architecture approach, and we sketch the TOGAF architecture development cycle. Section III describes the general structure of the ArchiMate language. In Section IV, we show how the different types of architectures as defined by TOGAF can be modelled with the ArchiMate Core language, while Section V introduces the two extensions that have been added in Version 2 of the ArchiMate language: the *motivation* extension and the *implementation and migration* extension. Finally, in Section VI we present some conclusions and directions for future work.

2 AN INTEGRATED APPROACH TO ENTERPRISE ARCHITECTURE

Frameworks for enterprise architecture vary in the types of support that they offer. They may have, among others, any combination of the following ingredients:

- A process (»*way of working*«) for creating architectures; this may be accompanied by guidelines, techniques and best practices.
- A set or classification of viewpoints.
- A language for describing architectures (defining concepts and relationships, but also a notation).
- The concept of a (virtual) architecture repository, possibly containing predefined architectural artefacts and (reference) models.

The core of TOGAF (The Open Group, 2011) is formed by the Architecture Development Method (ADM), a step-wise, iterative process for the development and implementation of an enterprise architecture (see Figure 2).

The ten phases of the ADM can be grouped into four main parts, also shown in Figure 2:

1. »Getting the organization committed and involved«: the Preliminary Phase prepares the organization as a whole for »working under architecture«, and involves activities such as establishing an architecture capability, tailoring the architecture methods and techniques for the specific characteristics of the organization, and defining an initial set of architecture principles; Phase A, Architecture Vision, prepares for a single architecture development cycle, and includes the formulation of an architecture vision with a high-level overview of the change that is envisaged.
2. »Getting the architecture right« concerns the description of the actual baseline and target architectures, and an analysis of the gaps between baseline and target. The three phases in this group are concerned with three main types of architectures: business architecture (Phase B), information systems architectures (Phase C), and technology architecture (Phase D).

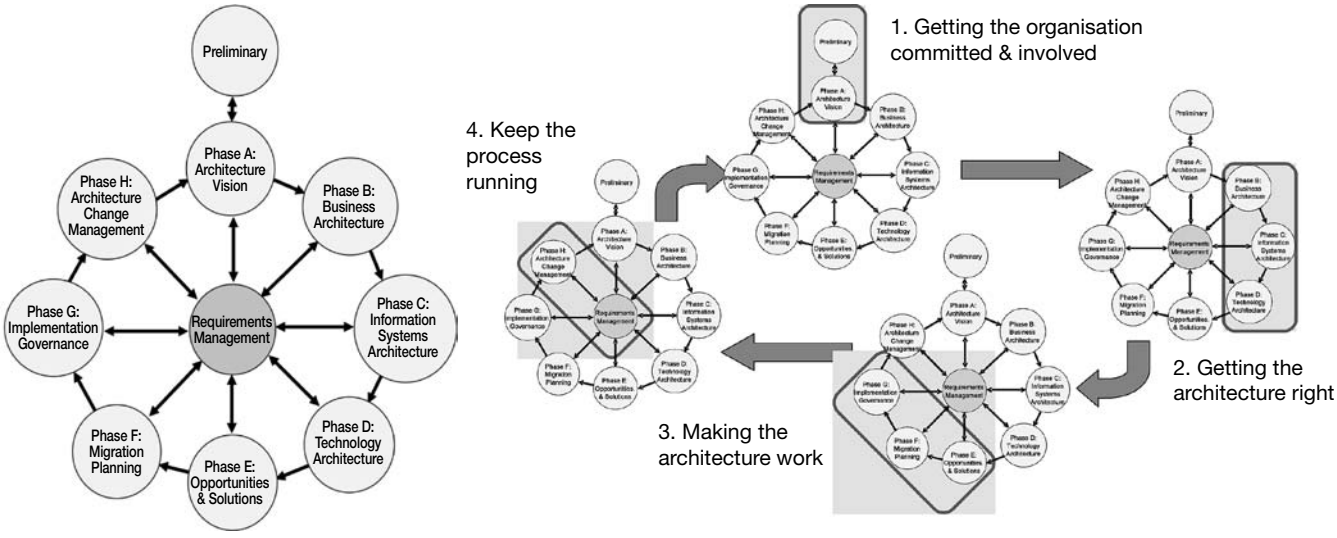


Figure 2. **TOGAF Architecture Development Method** (The Open Group, 2011)

3. »Making the architecture work« is concerned with the implementation of the developed architecture, and planning the migration to the new situation. It includes Phase E, Opportunities and Solutions, in which the gap analysis results are consolidated and potential implementation work packages are identified; Phase F, Migration Planning, in which work packages are prioritized and a migration plan is established; and Phase G, Implementation Governance, safeguarding the compliance of implementation projects with the architecture.
4. »Keep the process running« is concerned with the management, prioritization and version control of requirements on the architecture. The central Requirements Management process manages the requirements during an architecture development cycle. In Phase H, Change Management, new requirements are identified that may lead to the start of the new architecture development cycle.

TOGAF also includes the identification of viewpoints, techniques and reference models. However, it does not define an actual modelling language. The TOGAF Architecture Content Framework does indeed identify relevant architecture building blocks, but it does not constitute a precisely defined language, nor does it provide a notation for these building blocks. ArchiMate complements this by defining a fully worked out (graphical) modelling language, including the definition of relevant viewpoints. This language also provides a concrete visualization of the views identified in TOGAF.

TOGAF and ArchiMate share their view on the use of viewpoints, and the concept of an underlying common repository of architectural artefacts and models; i.e., they have a firm common foundation. However, TOGAF and ArchiMate complement each other with respect to the definition of an architecture development process and the definition of an enterprise architecture modelling language. Together, they make up a complete, integrated approach for delivering enterprise architecture.

3 STRUCTURE OF THE ARCHIMATE LANGUAGE

In this section we briefly discuss the core structures of the ArchiMate language. In (Lankhorst et al., 2010) a more detailed account is provided of the requirements on the language, and the design decisions that underpin its design.

A. Core Concepts

To arrive at a language that is easy to learn and understand, a conscious decision was made to limit the set of core modelling concepts. Therefore, a small number of generic modelling concepts have been created that essentially re-appear (in different variations) on the various layers of the language. First, we distinguish between the *structural* or *static* aspect and the *behavioural* or *dynamic* aspect. Structural concepts are assigned to behavioural concepts, to show who or what performs the behaviour. In addition to *active* structural elements (the business actors, application components and devices that performs actual behav-

iour, i.e., the 'subjects' of activity), we also recognize *passive* structural elements, i.e., the *objects* on which behaviour is performed.

Second, we make a distinction between an *external view* and an *internal* view on systems. The concept of *service* plays a central role in ArchiMate. A service represents a unit of essential functionality that a system exposes to its environment, and can be used to »bind« together the layers (applications provide services to business processes, while applications on their turn use infrastructure services). Furthermore, within a layer, services can be used as well to encapsulate behaviour. This enables the use of a services-oriented architecture (SOA) style from business processes, via applications to the underlying infrastructure. For the external users, only this external functionality, together with non-functional aspects such as the quality of service, costs etc., are relevant. Services are accessible through *interfaces*, which constitute the external view on the structural aspect.

Figure 3 summarizes the resulting generic core concepts of the language, as well as their main relationships

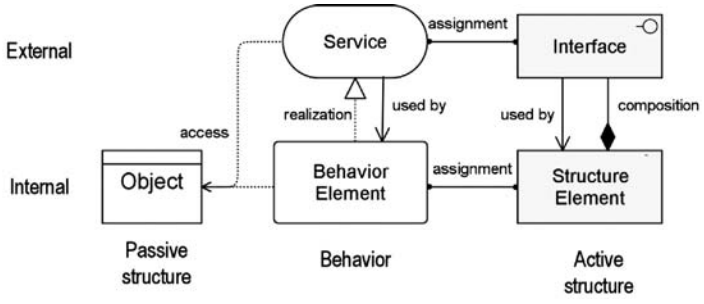


Figure 3. **Core Concepts of the ArchiMate Language**

B. Services as a Linking Pin Between Layers

In ArchiMate, the concept of service is defined as *the externally observable behaviour of a system*[1] *that may have some added value for that system's environment*. It is therefore natural to expect that, in the case of architecture layers, higher layers use the services supplied by the lower layers, since higher layers can be seen as the »environment« of the lower layers (see Figure 4). The enterprise's environment is the »end-user« of the services offered by the business layer of the enterprise. The business layer makes use of the services exposed by application layer (e.g., in order to support and automate its business processes). The application layer uses the services supplied by the technology layer (e.g., to make use of the physical resources – servers, networks etc. – in order to run its applications).

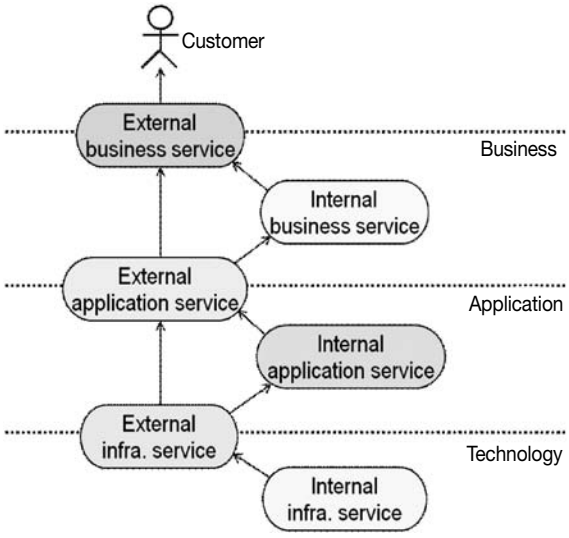


Figure 4. **Services**

In addition to the types of services mentioned above, the ArchiMate language distinguishes within each layer between internal and external services. Internal services are the services (added values) supplied to entities within the same layer. External services are the services made available to entities outside that layer.

4 CREATING ARCHITECTURE MODELS WITH ARCHIMATE

The primary use of ArchiMate in the context of TOGAF is in the representation of architecture models. TOGAF distinguishes four architectures: the Business Architecture (created in Phase B of the ADM), the Application Architecture and Data Architecture (both part of the Information Systems Architectures, Phase C) and the Technology Architecture (Phase D). In all of these phases, baseline (»*as is*«) and target (»*to be*«) architectures are created. In Phase A (Architecture Vision) of the ADM, first global versions of these architectures are already sketched; for this, a simplified subset of ArchiMate can be used.

We illustrate the different architectures with a small example based on a fictitious insurance company. ArchiSurance is a merger of three previously independent companies: Home & Away for home-

[1] System in the general sense, and not just as a synonym to application.

owner's and travel insurance, PRO-FIT for car insurance, and Legally Yours for legal aid insurance. The new company has a single Front Office and three separate Back Offices. ArchiSurance has plans to rationalize their application portfolio, by integrating legacy applications with similar functionality from the old companies that are still in use. Note that these examples give an impression of the ArchiMate language, but do not show all the concepts. For a complete overview of the language, please refer to (The Open Group, 2012).

A. Business Architecture

The Business Architecture provides the context for system development trajectories, showing, among others, the main business processes, the actors (or roles) performing these processes, and the information (objects) exchanged between the processes.

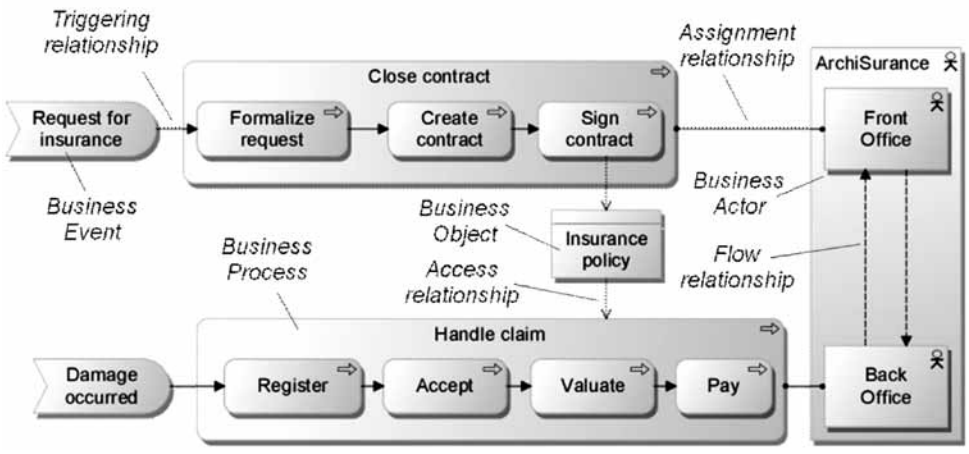


Figure 5. **Baseline and Target Business Architecture**

Figure 5 shows an example of a Business Architecture expressed in ArchiMate. We assume that the business architecture of ArchiSurance does not change in the application rationalization process.

B. Application Architecture

The Application Architecture shows the applications or application components, their relationships and their functionality. Figure 6 shows the baseline Application Architecture of ArchiSurance. The functionality that the applications offer to their environment is modelled with services. The service concept plays a central role in ArchiMate, also in the Business Architecture and the Technology Architecture (although this is not shown in our example), and in particular as a linking pin between the different architectures.

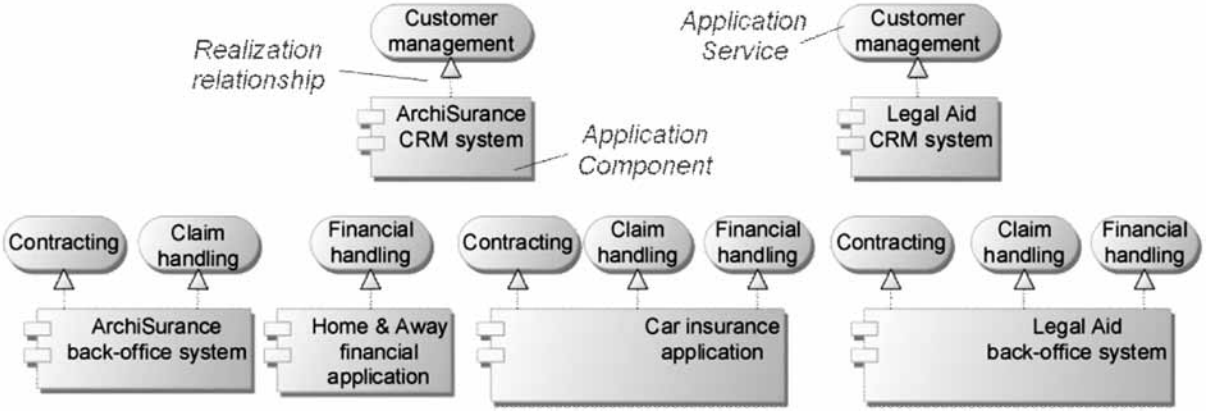


Figure 6. **Baseline Application Architecture**

Figure 7 shows the target Application Architecture of ArchiSurance, in which the legacy applications have been replaced by a single back-office system and a single CRM system for the whole company.

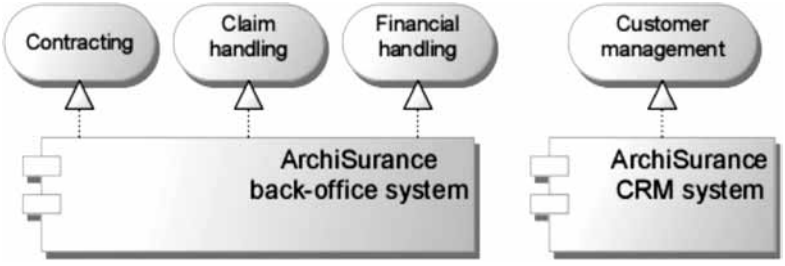


Figure 7. **Target Application Architecture**

In ArchiMate, separate views can be used to show the relationships between the different architectures. As an example of this, Figure 8 shows how the services from the Application Architecture are used in the processes of the Business Architecture.

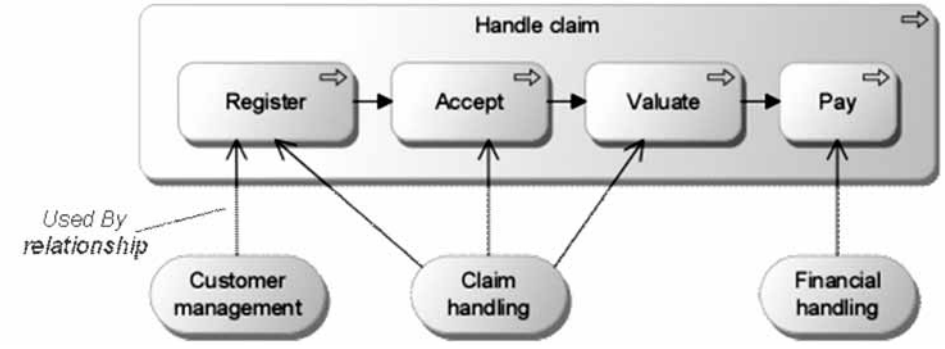


Figure 8. **Business-Application Alignment (Target)**

C. Data Architecture

The Data Architecture shows the main data objects used within the applications, as well as their relationships. Figure 9 shows the Data Architecture of ArchiSurance, which we assume will not change in the application rationalization process.

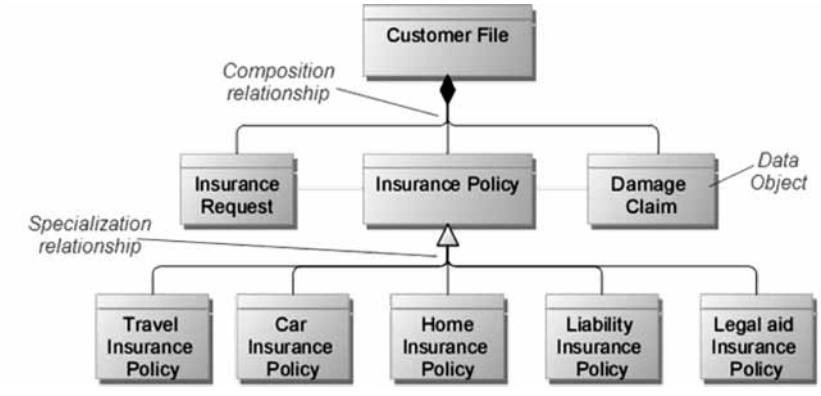


Figure 9. **Baseline and Target Data Architecture**

D. Technology Architecture

The Technology Architecture shows, among others, the devices and system software on which applications run, the networks connecting devices, and artefacts that form the physical implementation of application components or data objects. Figure 10 shows the baseline Technology Architecture of ArchiSurance. There are separate application servers for the different back-office applications.

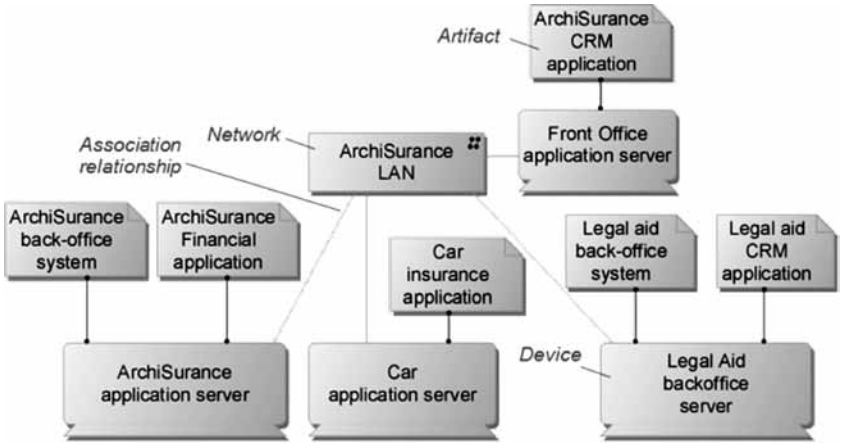


Figure 10. **Baseline Technology Architecture**

In the target Technology Architecture, as shown in Figure 11, some of these application servers become redundant. However, to increase reliability and availability, an additional backup server is introduced.

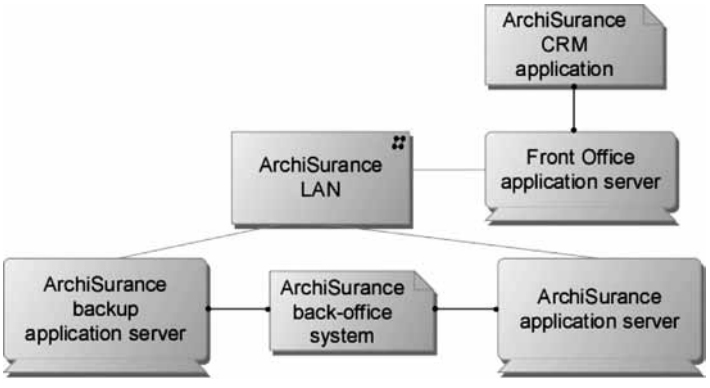


Figure 11. **Target Technology Architecture**

E. Gap Analysis

An important step in Phases B, C and D of the TOGAF ADM is a gap analysis, which reviews the differences between the baseline and target architecture. It shows which building blocks are carried over from baseline to target, which building blocks are new in the target architecture (which can be used as a basis to decide whether to buy or build these building blocks), and which elements have been eliminated from the baseline architecture (on purpose or accidentally; i.e., a gap analysis can also be used as a mechanism for validation of the target architecture). Subsequently, Phases E, F and G of the TOGAF ADM deal with the implementation of the proposed target architecture.

TOGAF suggests the use of a gap matrix as a technique for gap analysis. However, ArchiMate models also form a useful starting point for gap analysis, and the results can also be presented as an ArchiMate view. Figure 12 shows an example of this for the Technology Architecture.

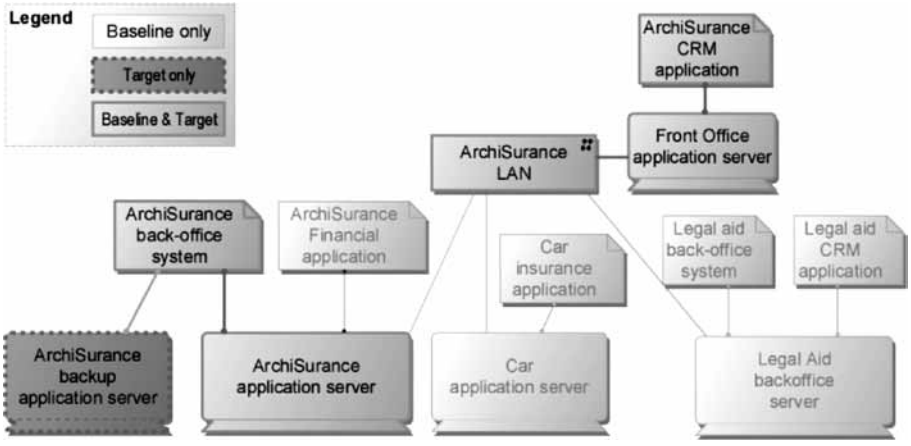


Figure 12. **Technology Architecture Gap Analysis**

5 EXTENDING THE ARCHIMATE COVERAGE OF TOGAF

As described in the previous sections, ArchiMate Version 1.0 chiefly supported modelling of the architectures in Phases B, C and D in the TOGAF ADM (»Getting the architecture right« in Figure 2), as is illustrated in Figure 13. The resulting models are used as input for the subsequent ADM phases. However, modelling concepts specifically aimed at the other phases – e.g., concepts for modelling principles, goals and requirements, or concepts to support migration planning – were still missing in the language. ArchiMate Version 2.0, which was launched early in 2012 by The Open Group, provides two extensions for this: one for describing motivation (e.g. stakeholders, goals and requirements), supporting the phases for »Getting the organization committed and involved« and »Keep the process running« from Figure 2, and one for implementation and migration planning, supporting the phases for »Making the architecture work« from Figure 2. The next subsections outline these two extensions.

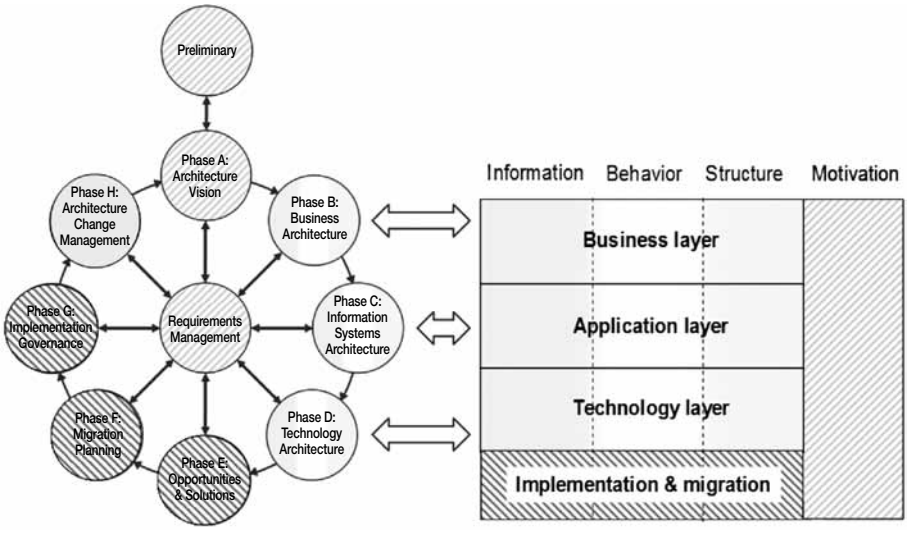


Figure 13. **TOGAF ADM and ArchiMate**

A. Motivation Extension

The motivation extension of ArchiMate (Engelsman, Jonkers, & Quartel, 2011) adds concepts related to goal modelling and business requirements management. They can be used for the identification, description, analysis and validation of goals and requirements at business level and their realization in enterprise architecture models as described with the ArchiMate core concepts.

The motivational concepts, based on sources

such as OMG's Business Motivation Model (Object Management Group, 2006), architecture principles (Proper & Greefhorst, 2010; Greefhorst & Proper, 2011) and goal-driven requirements engineering (Yu & Mylopoulos, 1994; Van Lamsweerde, 2001; Regev & Wegmann, 2005) are used to model the motivations, or intentions, that underlie the design of an enterprise architecture. These intentions influence, guide and constrain the design. Intentions are pursued by stakeholders, which can be individuals or groups such as a project team, enterprise or society. In addition, intentions may be organized into certain areas of interest, called drivers, such as customer satisfaction, compliance to legislation or profitability. Optionally, assessments of these drivers may be used to decide whether existing intentions need to be adjusted or not.

The actual intentions are represented by goals, principles and requirements. Goals represent some desired result – or end – that a stakeholder wants to achieve; e.g., increasing customer satisfaction with 10 percent. Principles and requirements represent desired properties of solutions – or means – to realize the goals. Principles represent desired properties that are required from all possible solutions in a given context; requirements represent desired properties of specific, individual solutions. For example, the requirement »*Use a single CRM system*« is a specialization of the principle »*Data should be stored only once*« by applying it to the current organization's architecture in the context of the management of customer data. Figure 14 shows an example of the use of the motivation concepts and their relationships, and also how requirements can be realized by core concepts.

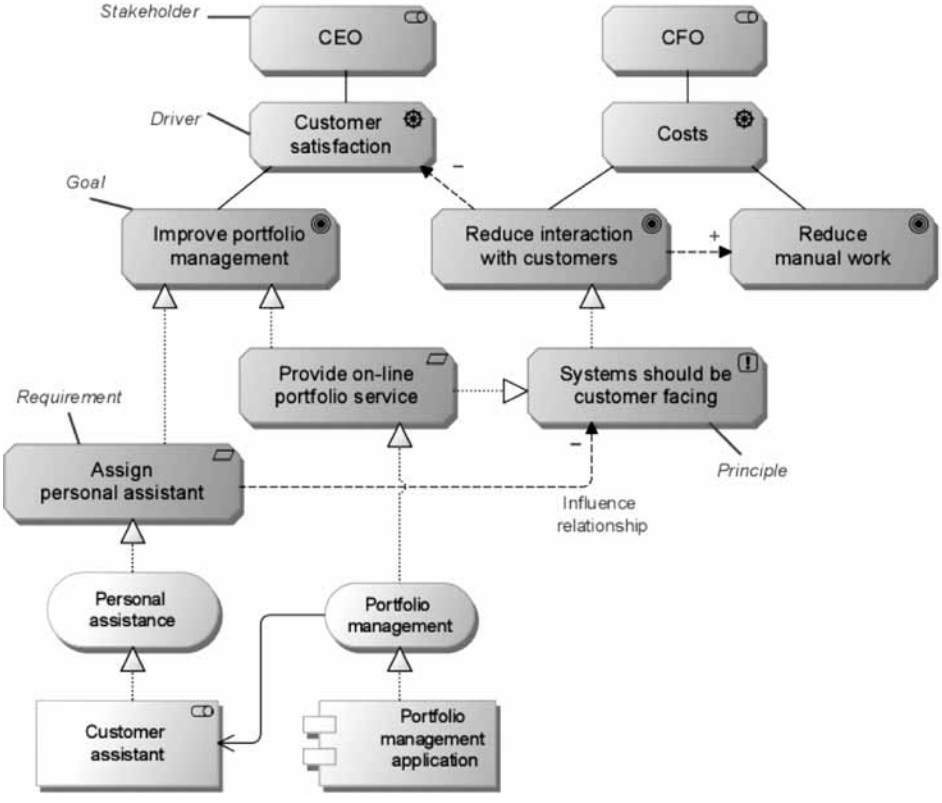


Figure 14. **Motivation Extension Example**

B. Implementation & Migration Extension

The Implementation and Migration Extension defines a number of additional concepts that enable the modelling of the architecture change process and increase the insight into these changes as well as their manageability in terms of portfolio and project management and decision-making. By defining concepts such as work package (to model implementation work at different levels of granularity, e.g., programs, projects, or project tasks), delivera-

ble and plateau it is possible to connect ArchiMate with program and project management standards and best practices, such as MSP (Chittenden & Van Bon, 2006)[1], PRINCE2 (The Stationary Office, 2009), and PMBoK (Project Management Institute, 2001).

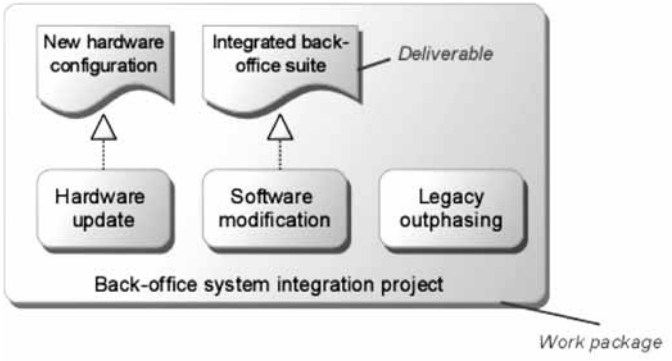


Figure 15. **Programs, Projects, Project Roles and Project Results**

Figure 15 shows an example of the use of work packages and deliverables. A project may be subdivided into a hierarchy of project tasks. Multiple projects which are managed together coherently, and which all contribute to a common outcome, can be grouped into a program. Work packages produce deliverables. These may be results of any kind, e.g., reports, papers, services, software, physical products, etc. A deliverable may also realize (a part of) architecture, or a solution that implements (a part of) architecture. To each work package, one or more business roles can be assigned.

An important premise in TOGAF is that the various architectures are described for different stages in time. In each of the Phases B, C, and D of the ADM, a Baseline Architecture and Target Architecture are created, describing the current situation and the desired future situation. In Phase E, »*Opportunities and Solutions*«, Transition Architectures are defined, showing the enterprise at incremental states reflecting periods of transition between the Baseline and Target Architectures. Transition Architectures are used to allow for individual work packages and projects to be grouped into managed portfolios and programs, illustrating the business value at each stage. In order to support this, the plateau concept was introduced.

Relationships can be established between the enterprise architecture models created at different moments in time and the migration models. Subsequently, analysis tools can be used to emphasize the differences between the different versions of models through the linked plateaus. These differences are captured by the concept of gap. A gap is an important outcome of a gap analysis in Phases B, C, and D of the TOGAF ADM, and forms an important input for the subsequent implementation and migration planning. The gap concept is linked to two plateaus (e.g., baseline and target architecture, or two successive transition architectures), and represents the differences between these plateaus (Figure 16),

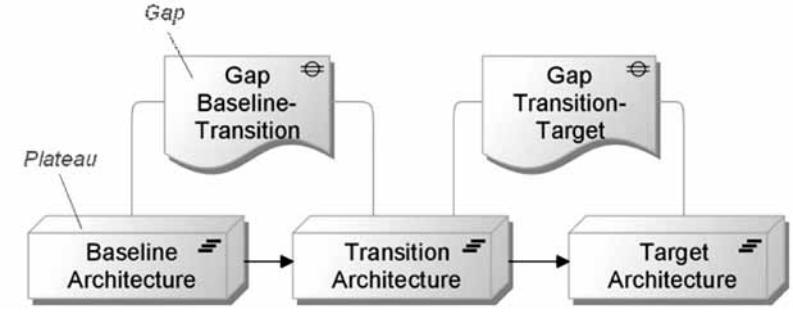


Figure 16. **Migration Planning Concepts**

6. CONCLUSIONS AND FUTURE DIRECTIONS

TOGAF is a leading enterprise architecture method of The Open Group. ArchiMate has recently been adopted as an Open Group standard for modelling enterprise architectures. TOGAF and ArchiMate share their view on the use of viewpoints, and the concept of an underlying common repository of architectural artefacts and models; i.e., they have a firm common foundation. However, they complement each other with respect to the definition of an architecture development process and the definition of an enterprise architecture modelling language. ArchiMate provides a concrete visualization for the architectures and views proposed in TOGAF.

From the previous sections, it is clear that TOGAF and ArchiMate can be used in conjunction and cover much of the same ground. TOGAF itself provides no guidance on creating a consistent overall model of the architecture, but refers to tools that should provide this support (The Open Group, 2011, Chapter 31):

»In order to achieve the goals of completeness and integrity in an architecture, architecture views are usually developed, visualized, communicated, and managed using a tool.

In the current state of the market, different tools normally have to be used to develop and analyze different views of the architecture. It is highly desirable that an architecture description be encoded in a ***standard language****, to enable a standard approach to the description of architecture semantics and their re-use among different tools.« (Emphasis ours).*

This is where ArchiMate nicely complements TOGAF: it provides a vendor-independent set of concepts that would help to create a consistent, integrated model »below the waterline«, which can be depicted in the form of TOGAF's views.

Presently, The Open Group is actively pursuing a closer integration between ArchiMate and TOGAF. An outline of this convergence is given by Jonkers, Proper, and Turner (2009). Some parts of TOGAF were not yet covered by ArchiMate Version 1.0 concepts. Therefore, two extensions to the language have been defined and included in Version 2.0 of the standard to fill these gaps. In particular, these concern on the one hand concepts for modelling the goals, motivations, principles and requirements used as inputs in defining an architecture, and on the other hand concepts for TOGAF's implementation and migration phases. With these two extensions, the new version of ArchiMate has full coverage of TOGAF. Thus, these two complementary open standards will reinforce each other and help to advance the enterprise architecture discipline in general.

Future research is concerned with potential additional extensions of the language in other directions. In the practical use of ArchiMate, a number of fields have been identified in which such future extension of the language may be advisable: e.g., concepts for modelling business policies, decisions and rules, concepts for better support of the design process, or concepts that provide the link to (business) models at a more strategic level.

7. REFERENCES

[1] Chittenden, J. & Van Bon, J. (2006). *Programme Management Based on MSP: A Management Guide*. Van Haren Publishing.

[2] Engelsman, W., Jonkers, H., & Quartel, D.A.C. (2011). *ArchiMate® Extension for Modeling and Managing Motivation, Principles, and Requirements in TOGAF™*, White Paper, The Open Group.

[3] Greefhorst, D. & Proper, H. A. (2011). *Architecture Principles – The Cornerstones of Enterprise Architecture*. Springer, Berlin.

[4] Jonkers, H., Proper, H. A., & Turner, M. (2009). *TOGAF and ArchiMate: A Future Together. A Vision for Convergence & Co-Existence*. Whitepaper, The Open Group.

[5] Jonkers, H., Van den Berg, H., Iacob, M.-E., & Quartel, D. A. C. (2010). *ArchiMate® Extension for Modeling the TOGAF™ Implementation and Migration Phases*, White Paper, The Open Group.

[6] Lankhorst, M. M., et al. (2009). *Enterprise Architecture at Work – Modelling, Communication and Analysis*, Second Edition, Springer.

[7] Lankhorst, M. M., Proper, H. A., & Jonkers, H. (2010). The anatomy of the archimate language. *International Journal of Information System Modeling and Design (IJISMD)*, 1(1), 1–32.

[8] Object Management Group (2005). *Unified Modeling Language: Superstructure v2.0*. Technical Report formal/05-07-04.

[9] Object Management Group (2006). *Business Motivation Model (BMM) Specification*. Technical Report dtc/06-08-03.

[10] Object Management Group (2008). *Business Process Modeling Notation*, v1.1. Technical Report formal/2008-01-17.

[11] Project Management Institute (2001). *Project Management Body of Knowledge*. Technical report.

[12] Proper, H. A. & Greefhorst, D. (2010). The role of principles in enterprise architecture. *Proceedings of the 5th Workshop on Trends in Enterprise Architecture Research*, Delft, The Netherlands.

[13] Quartel, D. A. C., Engelsman, W., Jonkers, H., & Van Sinderen, M. J. (2009). A Goal-Oriented Requirements Modeling Language for Enterprise Architecture. *Proceedings of the 13th IEEE International EDOC Enterprise Computing Conference*, Auckland, New-Zealand, 3–13.

[14] Regev, G. & Wegmann, A. (2005). Where do goals come from: the underlying principles of goal-oriented requirements engineering. *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, Paris, France.

[15] The Open Group (2011). *TOGAF® Version 9.1*, Van Haren Publishing.

[16] The Open Group (2012). *ArchiMate® 2.0 Specification*, Van Haren Publishing.

[17] The Stationary Office (2009). *Managing Successful Projects with PRINCE2*.
[18] Van Lamsweerde, A. (2001) Goal-oriented requirements engineering: A guided tour. *Proceedings of the 5th International Symposium on Requirements Engineering*.
[19] Yu, E. S. K. & Mylopoulos, J. (1994). Understanding 'why' in software process modelling, analysis, and design. *Proceedings of the 16th international conference on Software engineering*, Sorrento, Italy, 159–168.

▪

Henk Jonkers is a Senior Research Consultant at BiZZdesign. In this capacity, he is involved in the company's new developments in the area of enterprise architecture and enterprise engineering. He also participates in multi-party research projects, contributes to training courses, and performs consultancy assignments. Previously, he worked as a Member of Scientific Staff at Telematica Instituut (currently Novay), where he was involved in various applied research projects in the areas of business process modeling and analysis, enterprise architecture, service-oriented architecture, and model-driven development. Henk was one of the main developers of ArchiMate and an author of the ArchiMate 1.0 and 2.0 Specifications, and is actively involved in the activities of the ArchiMate Forum of The Open Group.

▪

Dick Quartel is a Senior Research Consultant at BiZZdesign. In this role he contributes to the development and improvement of BiZZdesign's products and services, is involved in research projects, supervises MSc students and interns, and performs consultancy assignments. In addition, he is an author of many scientific and professional publications, and an author of the ArchiMate 2.0 Specification. Previously, he worked as a Senior Researcher at Novay (formerly Telematica Instituut), where he acted as researcher and project manager and contributed to the definition and acquisition of research projects, and as an Assistant Professor at the University of Twente in the areas of distributed systems design, protocol design and implementation, and middleware systems.

▪

Henry Franken is Managing Director of BiZZdesign, and is responsible for the research and innovation portfolio of the company. As Chair of The Open Group ArchiMate Forum, Henry has led the development of the ArchiMate Version 2.0 standard. He has been a speaker at many conferences and has co-authored several international publications and Open Group White Papers. Henry is co-founder of the BPM Forum in the Netherlands.

▪

Analiza upravljanja poslovnih procesov z BPMN 2.0

Gregor Polančič, Gregor Jošt
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko
{gregor.polancic, gregor.jost}@uni-mb.si

Izvleček

Osnova uspešnega upravljanja poslovnih procesov so kakovostni modeli procesov, saj na njih pogosto temeljijo aktivnosti analiziranja, simuliranja in izvajanja. V širokem naboru razpoložljivih notacij za modeliranje poslovnih procesov smo se v prispevku omejili na notacijo BPMN, ki predstavlja de facto standard za modeliranje, z zadnjo različico 2.0 pa tudi za izvajanje poslovnih procesov. V prispevku smo ovrednotili upravljanje poslovnih procesov z BPMN in predstavili novosti, ki jih prinaša različica 2.0. Predstavili smo pomembnost informacijske podpore BPMN in rešitve za podporo modeliranja in izvajanja procesov BPMN. V obeh skupinah smo med sabo primerjali vidnejše predstavnike. Rezultati primerjave so pokazali, da je na trgu mnogo zrelih lastniških in odprtokodnih orodij, ki so primerna za upravljanje poslovnih procesov v BPMN 2.0. Da bi praktično predstavili funkcionalnosti in zmogljivosti orodij, smo v obeh skupinah rešitev BPMN izbrali predstavnika, ki smo ga predstavili podrobneje. V zadnjem razdelku smo na podlagi analize SWOT ovrednotili upravljanje poslovnih procesov z BPMN 2.0. Analiza je pokazala, da BPMN 2.0 odpira številne priložnosti, pa tudi izzive na področju upravljanja poslovnih procesov.

Ključne besede: poslovni proces, upravljanje poslovnih procesov, BPMN, modeliranje procesov, izvajanje procesov.

Abstract

The Analysis of Business Process Management Based on BPMN 2.0

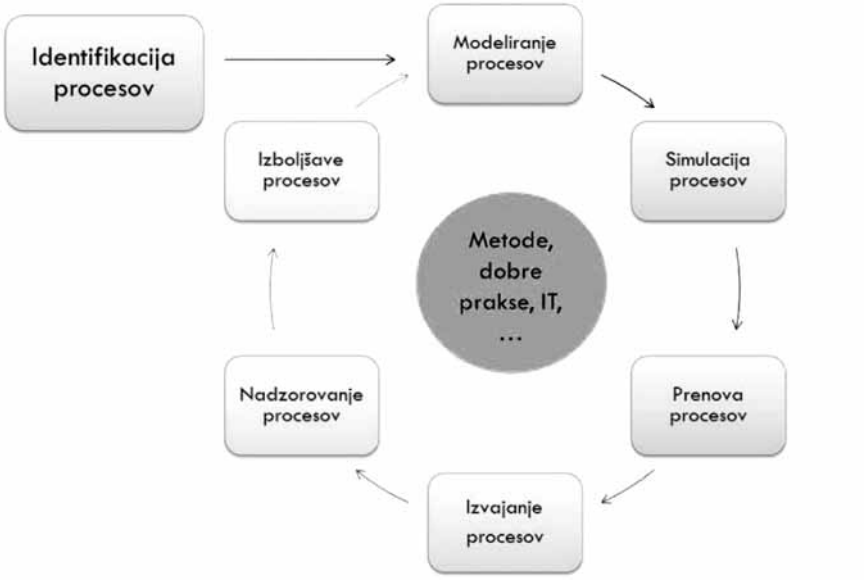
Effective business process management (BPM) depends on quality process models, which are often used for analysis, simulation and implementation of processes. Among many notations that enable business process modeling, we focused on the BPMN specification, which is the »de-facto« standard for business process modeling, and on the 2.0 version, for process execution as well. Our goal was to evaluate the BPM based on BPMN and the novelties, offered by the 2.0 release. In addition, the corresponding IT solutions for both BPMN modeling and implementation were investigated. In both groups, we compared the most popular solutions. The results of the analysis showed, that there are many mature proprietary and open source solutions, which are suitable for BPM in BPMN 2.0. In the light of practical demonstration of the tools' functionalities and performances, we presented in detail a representative from each group. In the last chapter, we evaluated the overall BPM based on BPMN 2.0 in the form of a SWOT analysis. The results showed that BPMN 2.0 opens up many opportunities, as well as challenges in the field of business process modeling and management.

Keywords: business process, business process management, BPMN, process modeling, process execution.

1 UVOD

Procesni pristop in upravljanje poslovnih procesov sta ključna dejavnika uspešnosti sodobnih organizacij. [1] Poslovni proces (angl. *business process*, v nadaljevanju proces) lahko opredelimo kot »skupek aktivnosti, ki iz enega ali več vhodov ustvarijo izhod, ki je pomemben za uporabnika. Proces ima jasen cilj in nanj vplivajo dogodki iz zunanjega sveta ali drugih procesov.« [2] Procesi se izvajajo v vsaki organizaciji, od nje pa je odvisno, ali jih prepozna in ustrezno upravlja. Skratka, vse, kar počnemo v podjetju, upodablja neki proces, ne glede na to, ali ga dokumentiramo.[3] Doseganje visoke stopnje zrelosti procesov je pogojeno z izbiro in izvajanjem uspešnega upravljanja procesov (angl. *business process management*, v nadaljevanju BPM). Področje BPM postaja vse kompleksnejše zaradi povečane dinamike poslovnega okolja in večanja povpraševanja po naprednejših storitvah ter produktih.

BPM je opredeljen kot »organiziran in discipliniran pristop identifikacije, načrtovanja, izvajanja, dokumentiranja, spremljanja, nadzorovanja in merjenja tako avtomatiziranih kot neavtomatiziranih poslovnih procesov, zato da bi zagotovili enakomerne in ciljne rezultate, ki so skladni s strateškimi cilji organizacije.« [4]



Slika 1: **Aktivnosti upravljanja poslovnih procesov [5]**

Iz slike 1 je razvidno, da je upravljanje procesov sestavljeno iz več aktivnosti, ki pogosto temeljijo na dobrih praksah oziroma standardih (npr. ISO 9000, COBIT, ITIL) in so podprte z informacijsko tehnologijo. Aktivnosti so povezane v cikel, ki omogoča, da se procesi nenehno spreminjajo, izboljšujejo oziroma prilagajajo. Organizacije, ki uspešno upravljajo procese, so v primerjavi s tradicionalnimi, funkcijsko ali hierarhično urejenimi organizacijami, uspešnejše, učinkovitejše in bolj prilagodljive [6] [7].

1.1 Modeliranje poslovnih procesov

Med poglavitne faze upravljanja procesov spada modeliranje (angl. *modeling*). Pod modeliranje štejemo izdelavo in uporabo modelov, pri čemer posamezni model predstavlja sliko oziroma posnetek določenega realnega stanja [3]. Glavni namen modeliranja je izdelava trenutnega modela procesa (angl. *as-is*), ki služi lažjemu razumevanju obstoječih procesov in prenovi (izboljšava) obstoječih procesov (angl. *to-be*). Razlogov za modeliranje je več, odvisno od tega, kakšne spremembe si želi organizacija oziroma v kakšnem obsegu je treba prenoviti poslovanje.

Pomembna lastnost modelov procesov je, da so razumljivi in prenosljivi znotraj organizacije kakor tudi med organizacijami. Zato jih je priporočljivo izdelati na podlagi standardiziranih notacij, ki običajno temeljijo na grafičnih simbolih in med katere štejemo predvsem diagram poteka (angl. *flowchart*), diagram toka podatkov (angl. *data flow diagram*), RAD (angl. *Role Activity Diagram*) in IDEF (angl. *Integrated Definition for Function Modeling*), ki pa so vezane pretežno le na manjše število orodij, poslovnih domen ali podpornih organizacij. Vodilni standardizirani notaciji, ki omogočata modeliranje procesov, sta dve, in sicer Unified Modeling Language (UML) in Business Process Model and Notation (BPMN) [8]. Poglavitna razlika med njima je, da je UML orientiran objektno, medtem ko je BPMN procesno orientiran. Zato je BPMN primernejši za modeliranje procesov [8]. BPMN se torej osredinja na procese, UML pa na razvoj programske opreme. Tako ne gre za konkurenčni notaciji, temveč za različne poglede na sisteme. Zaradi omenjenih razlogov je BPMN postal de facto standard na področju modeliranja procesov [9].

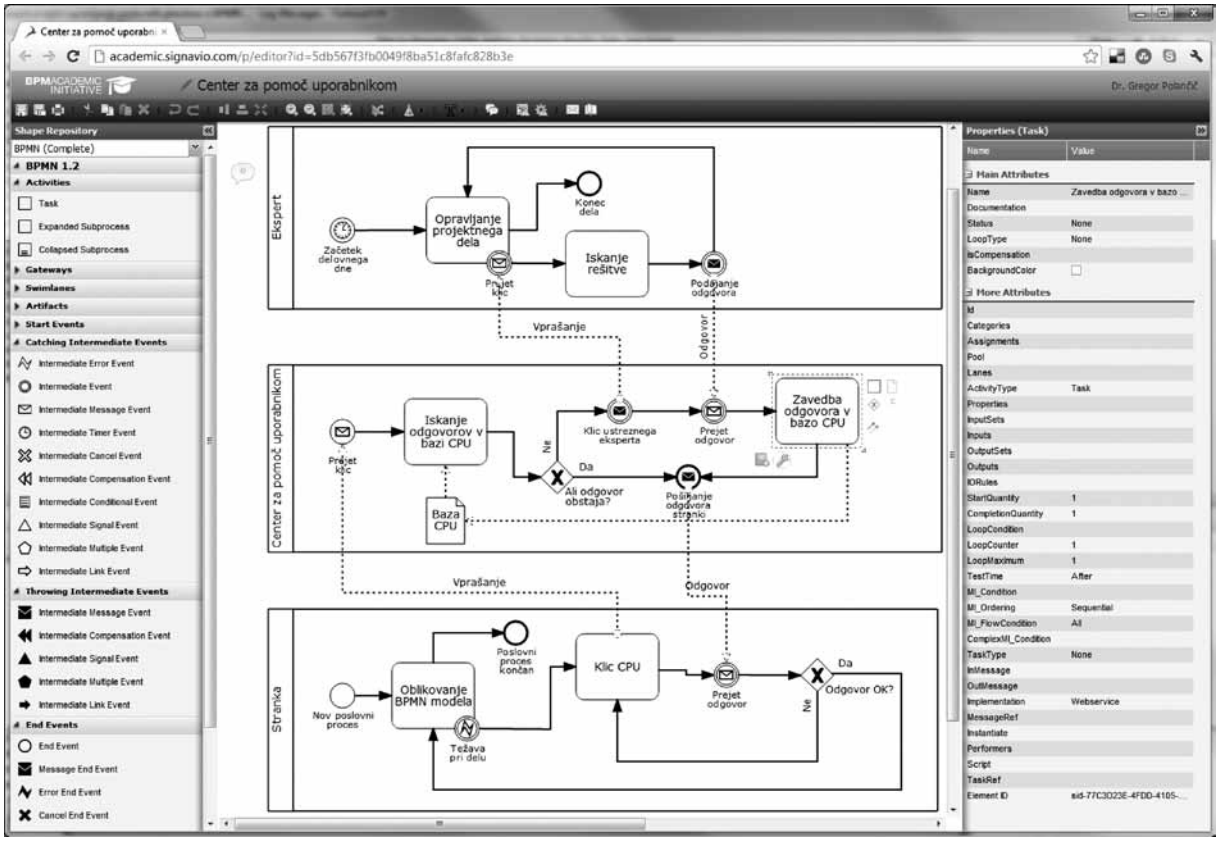
2 NOTACIJA BPMN

Notacija BPMN je nastala leta 2004 pod okriljem skupine BPMI (Business Process Management Initiative), ki se je leta 2006 združila s skupino OMG (Object Management Group). Namen BPMN je zagotoviti standardno notacijo, ki jo bodo razumeli vsi vpleteni v procesu, med katere štejemo poslovne analitike, programske inženirje, upravitelje in izvajalce procesov.

BPMN obsega koncepte modeliranja procesov, ne vključuje pa drugih tipov modeliranja, kot so modeliranje poslovnih pravil, podatkovnih ali informacijskih modelov in strategij [10]. BPMN definira grafično notacijo za modeliranje procesov oziroma za izdelavo diagramov poslovnih procesov (angl. *business process diagram*). Ker je glavni namen BPMN

priskrbeti preprosto notacijo za modeliranje enostavnih in kompleksnih procesov, je bilo treba grafični vidik notacije oblikovati na podlagi predhodnih notacij in elemente razvrstiti v posamezne kategorije. Tako uporabnik lažje prepozna osnovne elemente BPMN, kar mu omogoča razumevanje diagramov BPMN [10]. Poznamo pet glavnih tipov elementov [10]: 1) elementi za opredelitev toka dogodkov (angl. *flow objects*), 2) povezovalni elementi (angl. *connecting objects*), 3) steze (angl. *swim-lanes*), 4) podatki (angl. *data*) in 5) artefakti (angl. *artifacts*). Slika 2 prikazuje primer modela BPMN 1.2, izdelanega z orodjem Signavio, ki ga bomo podrobneje predstavili v nadaljevanju prispevka.

Model vsebinsko prikazuje proces pomoči uporabnikom, ki dejansko sestoji iz treh neodvisnih procesov (proces, ki se odvija v okolju stranke, proces središča za pomoč uporabnikom in delovni proces eksperta), ki med sabo sodelujejo s pomočjo pošiljanja in prejemanja sporočil.



Slika 2: **BPMN-model procesa pomoč uporabnikom, izdelan z orodjem Signavio**

3 NOVOSTI, KI JIH PRINAŠA BPMN 2.0

Zadnja različica BPMN je 2.0. Izšla je januarja 2011 in predstavlja največji preskok med izdanimi različicami BPMN. BPMN 2.0 temelji na BPMN 1.2 in razširja njene zmogljivosti v številnih pogledih. Spremembe so opazne že v sami kratici, saj BPMN v različici 2.0 ne pomeni več Business Process Modeling Notation, ampak Business Process Model and Notation. S to spremembo so želeli poudariti, da gre za več kot samo notacijo. Poglavitne novosti, ki jih prinaša BPMN 2.0, so [10]:

- novi elementi modelov procesov,
- diagram koreografije (angl. *choreography*),
- diagram pogovora (angl. *conversation*),
- semantika izvajanja procesov (angl. *execution semantics*),
- tipi skladnosti (angl. *conformance levels*) in
- standardizirana shema XML.

3.1 Novi elementi modelov procesov

BPMN 2.0 uvaja nekatere nove elemente, med poglavitne štejemo:

- nove vrste dogodkov (eskalacija, paralelni sestavljeni dogodek, neprekinjajoči dogodki),
- opcijski dogodkovni podprocesi,
- nove vrste vrat (ekskluzivna in paralelna začetna dogodkovna vrata),
- grafične oznake za različne vrste aktivnosti (ročna aktivnost, uporabniška aktivnost, storitev, skripta, poslovno pravilo, pošiljanje sporočila in prejemanje sporočila),
- novi podatkovni objekti (podatkovna baza, zbirka podatkov, sporočilo).

Razlogi za vpeljavo novih elementov izhajajo tudi iz potreb izvajanja poslovnih procesov, ki ga podpira BPMN 2.0. S tem se je že tako obsežni nabor elementov BPMN še dodatno razširil (BPMN 2.0 vsebuje več kot sto elementov) in pripomogel k povečanju kompleksnosti notacije in na njej temelječih modelov. Zato so v BPMN 2.0 razvrstili elemente v tri skupine [8] [11]:

- **skupina opisnih (angl. *descriptive*) elementov;** vključuje vizualne elemente, ki so namenjeni visokonivojskemu modeliranju in dokumentiranju procesov. Predpostavljamo, da množico teh elementov vizualno prepozna večina izvajalcev procesov, saj so v večji meri skladni z drugimi notacijami (npr. z diagrami poteka). Primeri elementov skupine opisnih elementov so preprost začetni in končni dogodek, bazeni in steze, aktivnosti, tok zaporedja in sporočil, ekskluzivna in paralelna vejitev;
- **skupina analitičnih (angl. *analytic*) elementov;** dodatno vključuje vizualne elemente, ki omogočajo natančno modeliranje procesov za potrebe analiziranja, nadziranja in za predstavitev procesov, ki se izvajajo na procesnih strojih. Primeri elementov skupine opisnih elementov so preostali dogodki, pogojni tokovi, izjeme in vse vrste vejitev;
- **skupina izvajalnih (angl. *executable*) elementov;** vsebuje še preostale elemente, ki so potrebni za modeliranje procesov do natančnosti, ki zadostuje za neposredno izvajanje modelov na procesnih strojih. Zaradi tega se določeni elementi iz analitične skupine razširjajo z dodatnimi atributi; npr. dogodkovni vejitvi, ki je v skupini analitičnih elementov vsebovala tri atribute (id, ime in tip dogodkovne vejitve), dodamo še četrti atribut – usmerjenost dogodkovne vejitve.

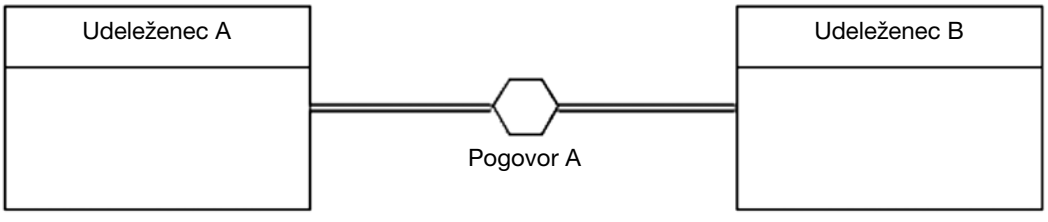
3.2 Novi modeli

Poleg modelov, ki jih je bilo mogoče izdelovati v prejšnjih različicah BPMN (interni procesi, sodelovanje, javni procesi), vpeljuje BPMN 2.0 še model koreografije in model pogovora.

Koreografija je tip procesa, ki se od običajnega BPMN procesa razlikuje v namenu in obnašanju. Koreografija formalizira način, kako udeleženci v procesu koordinirajo svoje interakcije. Poudarek je na izmenjavi informacij med udeleženci. Za potrebe koreografije je v BPMN 2.0 dodan nov element in sicer opravilo koreografije (angl. c*horeography task*). Predstavlja atomarno aktivnost znotraj koreografije, pri čemer sta prisotna vsaj dva udeleženca (slika 3). Podopravilo koreografije je lahko zloženo (angl. *collapsed*) ali razširjeno (angl. *expanded*) [10].

Slika 3: **Gradnik »opravilo koreografije« [10]**

Diagram **pogovor** je podoben diagramu sodelovanja, s to razliko, da steze ne vsebujejo procesov in med stezami ni dovoljeno vstaviti koreografije. Dodana sta dva nova gradnika: pogovor in povezava pogovora (slika 4) [10].



Slika 4: **Sintaksa diagrama »pogovor« [10]**

3.3 Semantika izvajanja procesov

V preteklih specifikacijah BPMN so ostale mnoge podrobnosti, ki so ključne za izvajanje procesov, brez jasne definicije. Različica BPMN 2.0 uvaja semantiko, potrebno za izvajanje procesov (interpretacija in izvajanje modelov BPMN sta opisani jasno in natančno), ki je v bistvu namenjena razvijalcem orodij za implementacijo simulacij, animacij in izvajanja poslovnih procesov na procesnih strojih. Razvijalci orodij se morajo namreč strogo držati standarda definirane semantike izvajanja, saj se medobratovalnost (angl. *interoperability*) lahko doseže samo, če ohranimo standardno semantiko in sintakso [10].

3.4 Tipi skladnosti

V preteklih različicah BPMN ni bilo definirano, katere zahteve mora podpirati programsko orodje, da je skladno z BPMN. Zato so v BPMN 2.0 jasno definirali merila ustreznosti za ugotavljanje skladnosti programskega orodja s specifikacijo BPMN. Nova različica standarda definira štiri tipe ustreznosti, ki bodo v prihodnje igrali pomembno vlogo pri izbiranju orodij BPMN [10]:

- ustreznost modeliranja procesov (angl. *process modeling conformance*),
- ustreznost izvajanja procesov (angl. *process execution conformance*),
- ustreznost izvajanja procesov BPEL (angl. *BPEL process execution conformance*) in
- ustreznost modeliranja koreografije (angl. *choreography modeling conformance*).

3.5 Standardizirana shema XML

BPMN je že od samega začetka standard, ki definira obliko in pomen posameznih simbolov, vendar vse do BPMN 2.0 ni bilo na voljo standardiziranega formata za izmenjavo diagramov. Združenje WfMC (Workflow Management Coalition) je sicer ponudilo standardiziran format XPDL (XML Process Definition Language), ki je namenjen prenosljivosti poslovnih modelov med različnimi orodji, vendar predhodne različice BPMN niso vsebovale sheme XML za izvoz in uvoz modelov. Težave glede prenosljivosti so naslovili v BPMN 2.0 standardu tako, da so definirali standardizirano shemo XML za izmenjavo izvršljivih ali neizvršljivih modelov [10].

4 ANALIZA PROGRAMSKIH REŠITEV ZA PODPORO BPMN 2.0

Za uspešno upravljanje poslovnih procesov je smiselno upoštevati tale načela [7]:

- procesi so dobrine *(angl. assets)* organizacije, ki strankam prinašajo vrednost, zato je
- procese treba upravljati in nenehno spreminjati (prilagajati in izboljševati) in
- informacijska tehnologija je ključni dejavnik uspešnega upravljanja procesov.

Uradna spletna stran BPMN beleži 76 orodij, ki podpirajo modeliranje poslovnih procesov, vendar večina ne podpira njihove informatizacije. Še pred različico BPMN 2.0 je bilo na voljo več lastniških rešitev, v okviru katerih je vsako orodje ponujalo svoj pristop k informatizaciji poslovnih procesov [13]. Primera takšnih orodij sta 1) IBM Web Sphere, ki omogoča preslikavo iz notacije BPMN v izvršljivi format BPEL in 2) Bizagi BPM Suite, ki zagotavlja nestandardiziran način izvajanja procesov BPMN. Kot že omenjeno, imamo z različico BPMN 2.0 na voljo tudi semantiko izvajanja, kar razvijalcem orodij omogoča enotno interpretacijo posameznih elementov BPMN z vidika avtomatiziranega izvajanja.

Informacijska tehnologija lahko torej podpre BPMN 2.0 na dveh glavnih področjih: pri modeliranju procesov in izvajanju procesov. V nadaljevanju bomo predstavili najbolj uveljavljenje lastniške in odprtokodne informacijskotehnološke rešitve za potrebe modeliranja in izvajanja procesov.

4.1 Modeliranje procesov

Na področju modeliranja procesov zgolj manjši delež orodij podpira modeliranje na podlagi BPMN 2.0. Večina orodij še vedno temelji na ustaljeni različici BPMN 1.2. Poleg specifičnih orodij BPMN pa podpirajo modeliranje na podlagi BPMN še številna splošna modelirna orodja, kot sta npr. MS Visio in Magic Draw. V splošnem so specifična orodja BPMN učinkovitejša, medtem ko so splošna orodja primernejša za podjetja, ki poleg BPMN uporabljajo še druge notacije.

V spodnji tabeli predstavljamo in primerjamo tri orodja, ki podpirajo modeliranje z BPMN, in sicer splošno orodje za modeliranje (MS Visio), namensko namizno orodje za modeliranje BPMN (BizAgi) in namensko storitev (SaaS) za modeliranje BPMN (Signavio).

Tabela 1: **Primerjava treh priljubljenih orodij za modeliranje BPMN 2.0**

Polni naziv orodja	Microsoft Visio 2010 Premium	BizAgi Process Modeler	Signavio Process Editor
Vrsta orodja	Generično orodje za modeliranje, ki privzeto podpira BPMN	Namensko namizno orodje BPMN	Namenska storitev za modeliranje BPMN
Spletna stran izdelka	office.microsoft.com	www.bizagi.com	www.signavio.com
Ali se nahaja na OMG seznamu orodij BPMN	Ne, ker je generično orodje za modeliranje	Da	Da
Razlog za izbor	Izkazal se je za najbolj priljubljeno orodje za modeliranje BPMN procesov [12] in kot de facto orodje za poslovno modeliranje [13].	Priljubljeno orodje za modeliranje BPMN [14] [15]	Temelji na odprtokodnem orodju Oryx, ki je bilo izbrano za najboljše odprtokodno orodje za modeliranje BPMN 2.0 [15].
Podprte platforme	Windows	Windows	Spletno orodje
Prva verzija	2003	2008	2009
Zadnja verzija	2010	2.2	5.3.1
Licenca	Lastniška	Brezplačna (angl. freeware)	Plačljiva storitev
Cena licenc	999,99 dolarja	/	81,95 evra na mesec
Podpora BPMN	BPMN 1.2	BPMN 2.0	BPMN 2.0
Generiranje poročil	Ni podprto	Podprto	Podprto
Hranjenje modelov	Lokalno	Lokalno	Lokalno/deljeno
Beleženje različic	Ni podprto	Ni podprto	Podprto
Izmenjava modelov	Ni podprto	XPDL	XPDL, BPMN 2.0

Na podlagi primerjave in preizkusa zgoraj predstavljenih orodij smo v nadaljevanju izpostavili storitev Signavio, ki je dostopna prek brskalnika. Signavio spada med prva orodja, ki so podprla BPMN 2.0, in je zaradi spletne zasnove še posebno primeren za delo v skupini. Temelji na odprtokodni rešitvi Oryx, ki je bila razglašena za najboljšo odprtokodno orodje za modeliranje poslovnih procesov v BPMN 2.0 [15].

Orodje Signavio prihaja v dveh različicah, in sicer programska oprema kot storitev (javni oblak) in lokalna namestitev (zasebni oblaki) [16]. V primeru javnega oblaka nameščanje programske opreme ni potrebno, saj se sistem nahaja na strežnikih podjetja Signavio. Vsi podatki so shranjeni v centralni shrambi (angl. *central repository*), za dostop do orodja pa potrebujemo spletni brskalnik (priporočljiv je Mozilla Firefox) in povezavo z internetom.

Lokalna namestitev (angl. *On-Premise Installation*) omogoča, da vključimo Signavio v lastno infrastrukturo. Strežniški del aplikacije namestimo na strežnik, ki ga potem vzdržuje ekipa znotraj organizacije. Dostop do aplikacije lahko omejimo samo na zaposlene v podjetju ali omogočimo, da je aplikacija dostopna tudi prek spleta. Podobno kot pri različici »programska oprema kot storitev« tudi pri tem ni treba nameščati aplikacije na odjemalčevo stran, ampak je dostopna prek spletnega brskalnika.

Signavio podpira modeliranje v BPMN 2.0, BPMN 1.2 in EPC. Skladno s standardom BPMN 2.0 uvaja tudi dva nova diagrama, in sicer diagram koreografije (angl. *choreography*) in diagram pogovora (angl. *conversation*). Orodje ponuja jedrni in celotni prikaz elementov izbranega diagrama. Pri celotnem prikazu se vsi elementi razvrstijo glede na kategorije (slika 2), v katere pripadajo, pri jedrnem pa je na voljo 12 glavnih elementov BPMN.

Signavio omogoča, da posamezni diagram sočasno ureja več oseb, za vsak diagram pa vodijo revizijo sprememb. Posamezni uporabnik lahko pregleduje, komentira ali ureja diagram, odvisno od dodeljenih pravic.

4.2 Izvajanje procesov

Izvajanje procesov je mogoče uvesti z uporabo konvencionalnih tehnik in tehnologij (npr. razvoj aplikacije Java ali .NET, ki podpre specifični model procesa), vendar imajo takšne rešitve nekatere omejitve:

- izvajanja procesov ni mogoče realizirati na podlagi modelov procesov, ki so uporabni kvečjemu kot zahteve za uvedbo rešitve, ki podpre specifičen proces;
- procesi, ki so implementirani v takšnih rešitvah, so težko prenosljivi v druga izvajalna okolja;
- dejanski procesi se nenehno spreminjajo, nadgra-

jujejo in prilagajajo. Ker takšne rešitve implementirajo proces v obliki programskega koda, se težko in počasi prilagajajo spremembam procesov;
- poleg izvajanja vključuje upravljanje procesov še številne druge aktivnosti (analiziranje, simuliranje in nadzorovanje), ki jih ne podpirajo takšna izvajalna okolja.

Drugi pristop k izvajanju procesov je uporaba procesnih strojev (angl. *process engine*). Eden izmed poglavitnih jezikov za izvajanje poslovnih procesov na procesnih strojih je BPEL (angl. Business Process Execution Language), ki je na voljo že od leta 2003 [17]. Ker BPEL ne zagotavlja standardiziranega prikaza poslovnih procesov, se je v ta namen pogosto uporabljal BPMN, saj omogoča preslikavo iz BPMN v BPEL [10]. Procesni stroji so sestavnih del sistemov za podporo upravljanja procesov (angl. *business process management systems* – BPMS). Skupaj s podpornimi orodji se takšni sistemi združujejo v pakete (angl. *suites*), ki podpirajo vse poglavitne faze upravljanja poslovnih procesov: modeliranje, analiziranje, simuliranje, izvajanje in nadzor nad procesi. Sodobni BPMS-i se usmerjajo v podporo BPMN 2.0, kar v praksi pomeni, da lahko znotraj BPMS izdelamo (ali uvozimo) model procesa BPMN in ga izvajamo neposredno brez preslikave v BPEL.

Z »obogatitvijo« modela BPMN z dodatnimi informacijami (npr. definiranje spremenljivk, uvedbe ali povezovanja aktivnosti s storitvami obstoječih informacijskih rešitev, izdelave uporabniških vmesnikov) je BPMS sposoben izvajati primerke modela procesa. S tem BPMS-ji izničijo ali vsaj minimizirajo zgoraj navedene omejitve konvencionalnih rešitev za izvajanje procesov. Poglavitna prednost BPMS je podpora več fazam upravljanja procesov, izvajanje na podlagi modelov procesov in hitre spremembe izvajanja na podlagi sprememb modelov procesov. Zato so BPMS primerni za »procesne organizacije«, ki so usmerjene v izdelke ali storitve in se pogosto prilagajajo spremembam na tržišču. Tabela 3 predstavlja nabor takšnih odprtokodnih in lastniških rešitev, ki temeljijo na notaciji BPMN.

Tabela 3: **Primerjava orodij BPMS**

Orodje	Licenca	Razvojno okolje	Podprti spletni strežniki	Semantika izvajanja	Podprti programski jezik
Intalio	Odprtokodna različica z omejenim številom modulov, lastniška	Namizno in SaaS	AWS, Intalio|Application Server (temelji na odprtokodnem strežniku Jetty)	BPEL 2.0, BPMN 2.0	Java, X#
Bonita Open Solution	Odprtokodna različica z omejenim številom modulov, lastniška	Namizno	Poljuben javanski spletni strežnik, privzeto Tomcat	BPMN 2.0	Java
IBM Web Sphere	Lastniška	Namizno	WebSphere aplikacijski strežnik	BPMN za modeliranje, BPEL za izvajanje	Java
Oracle BPM 11g	Prosto programje	Namizno in spletno	Oracle WebLogic Server	BPMN 2.0	Java
BizAgi BPM Suite	Prosto programje za modeliranje, lastniško za izvajanje	Namizno	Internet Information Services (IIS), Web Logic, Web Sphere, JBoss in Glassfish	Notacija BPMN 2.0 za modeliranje, lastniška implementacija poslovnih procesov	Platforma .NET ali JEE
jBPM	Odprtokodna	Namizno in spletno	Poljubni javanski spletni strežnik, privzeto Tomcat	BPMN 2.0	Java
Activiti	Odprtokodna	Namizno in spletno	Poljubni javanski spletni strežnik	BPMN 2.0	Java

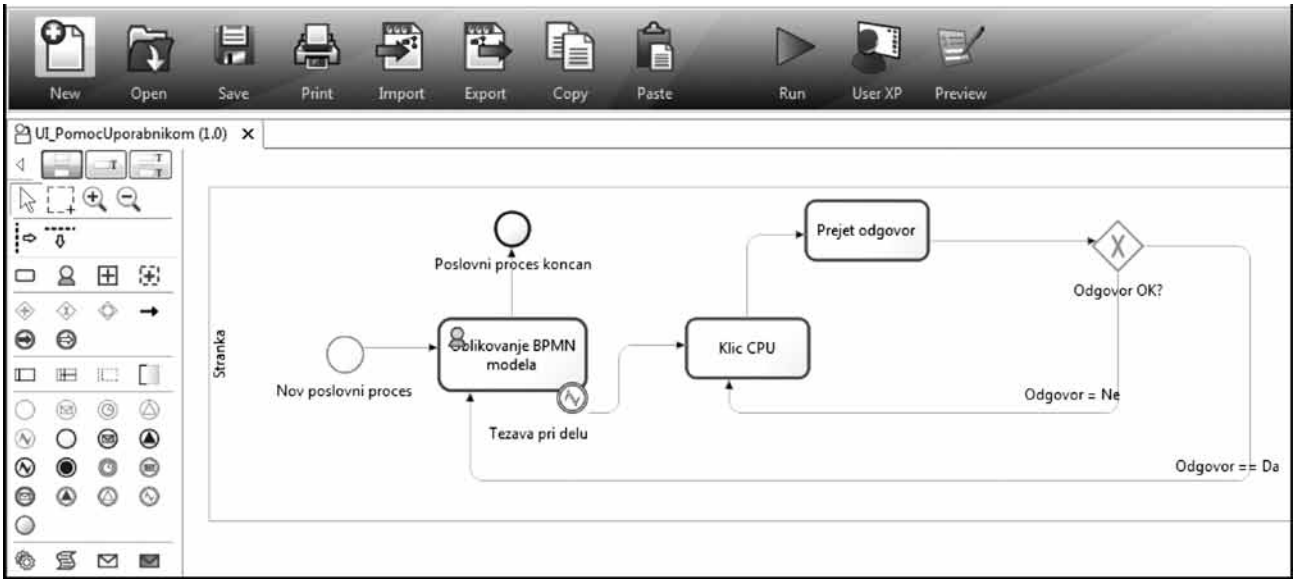
Kot je razvidno iz tabele, večina orodij že podpira BPMN 2.0 in omogoča razvoj oz. dopolnitev modela v programskem jeziku Java. V sklopu prispevka smo v nadaljevanju izpostavili orodje Bonita Open Solution (v nadaljevanju Bonita), ki predstavlja vodilno odprtokodno rešitev na področju izvajanja poslovnih procesov [18].

Bonito razvija podjetje Bonita Soft, ki ga je

angleška revija CIO umestila na lestvico dvajsetih najobetavnejših podjetij leta 2011. Bonita je odprtokodni BPMS, ki podpira modeliranje, razvoj, izvajanje in nadzor poslovnih procesov.

V fazi modeliranja poslovnih procesov Bonita omogoča [19]:

- modeliranje poslovnih procesov v notaciji BPMN (slika 5),
- simulacijo poslovnih procesov z atributi, kot so cena, trajanje in poraba virov,
- preverjanje veljavnosti modelov BPMN in
- uvoz in izvoz modelov BPMN v oblikah XPDL, jBPM3 in BPMN2.



Slika 5: **Segment modela procesa BPMN »pomoč uporabnikom«, izdelanega v orodju Bonita**

Vsebinsko je model na zgornji sliki ekvivalenten modelu iz razdelka 2 (slika 2) s to razliko, da prikazuje le vidik stranke. Iz modela je razvidno, da vsebuje tri aktivnosti, izmed katerih ena zahteva interakcijo uporabnika (oblikovanje modela BPMN), ki se realizira s spletnim obrazcem (vnos zapisa XML modela BPMN, slika 6). Preostali dve aktivnosti in drugi elementi se v našem primeru izvedejo samodejno. Prav tako je iz slike 5 razvidna razlika med orodjema Signavio in Bonita, saj ima drugo bistveno manj elementov za potrebe modeliranja.

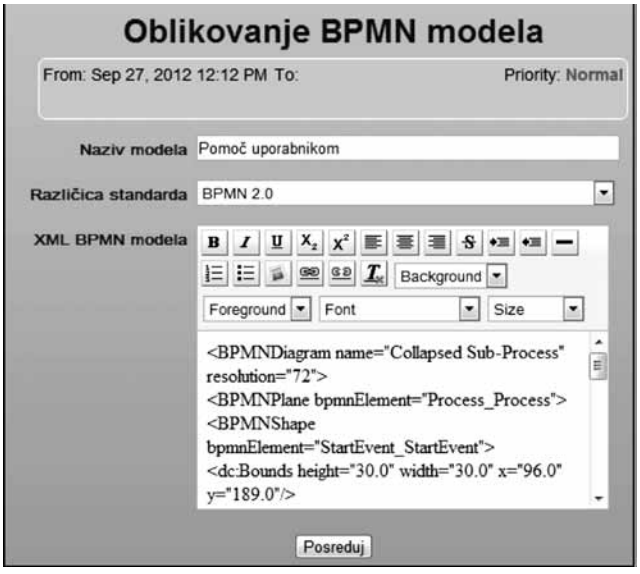
V fazi razvoja omogoča Bonita te funkcionalnosti [20]:

- napredno ustvarjanje in urejanje spletnih obrazcev,
- prilagoditev vizualnih predlog,
- ustvarjanje in prilagoditev obstoječih priključkov (angl. *connectors*) in
- generiranje popolno delujoče aplikacije, ki temelji na procesu.

V fazi izvajanja poslovnega procesa nam Bonita omogoča:

- oblikovanje uporabniškega vmesnika za končne uporabnike,
- integracijo v obstoječe rešitve (npr. spletne storitve, različne podatkovne baze, LDAP, storitve Google, sporočilni sistemi, Pay Pal, SAP, Share Point ipd.),
- »družbeno« upravljanje procesov (povezava procesov z družbenimi omrežji) in
- spremljanje izvajanja procesov v realnem času.

Slika 6 prikazuje uporabniški vidik spletne aplikacije, ki temelji neposredno na modelu BPMN »pomoč uporabnikom«, izdelanem v Boniti (slika 5).

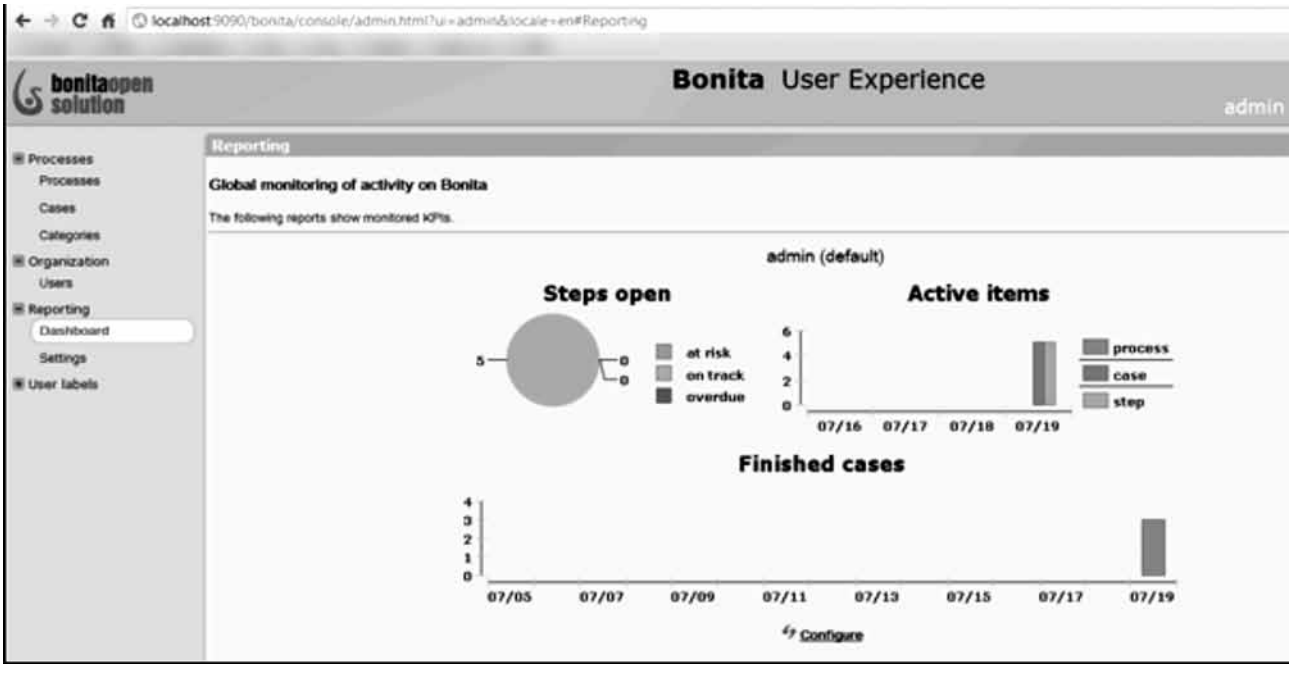


Slika 6: **Spletni obrazec procesa, ki ga generira orodje Bonita**

Kot je razvidno na sliki 6, se definirani model BPMN samodejno preslika v obrazce spletne aplikacije. Primer prikazuje informatizirano izvajanje aktivnosti »oblikovanje modela BPMN«. Vnosna polja so skladna z definiranimi atributi v modelu BPMN, pri čemer smo opredelili, da uporabnik v sklopu omenjene aktivnosti vnese naziv procesa, izbere verzijo modela BPMN in predloži zapis XML želenega modela.

V fazi nadzora in administracije imamo v Boniti na voljo te funkcionalnosti [20]:

- pregled nad procesi s pomočjo BAM (angl. *Business Activity Monitoring*), kot prikazuje slika 7,
- upravljanje življenjskega cikla poslovnih procesov,
- upravljanje s podatki, z opravili in uporabniki in
- definiranje lastnih poslovnih in tehničnih preglednih plošč (angl. *dashboard*).



Slika 7: **Administracija poslovnih procesov v orodju Bonita**

Slika 7 prikazuje pregledno ploščo Bonite (angl. *dashboard*), na kateri je vidno število procesov v izvajanju brez zamud in z zamudami ter število aktivnih in izvedenih procesov glede na opredeljeno prioriteto.

5 SKLEP

V prispevku smo analizirali upravljanje poslovnih procesov v notaciji BPMN, ki je de facto standard za modeliranje poslovnih procesov. Z zadnjo različico BPMN 2.0 se je standard razširil z novostmi, ki prinašajo prednosti, slabosti, priložnosti in grožnje, ki so prikazane v tabeli 5.

Tabela 5: **SWOT-analiza upravljanja poslovnih procesov z BPMN 2.0**

Prednosti	**Slabosti**
Standardizirana notacija Veliko število gradnikov Možnost modeliranja različnih tipov procesov (notranji, sodelovanje, koreografija, javni, pogovor) Veliko razpoložljivih orodij za modeliranje in dobra podpora industrije Standardiziran zapis diagramov XML Možnost izvajanja procesov Možnost razširjanja in prilagajanja notacije	Kompleksnost standarda (prek 500 strani) Kompleksnost notacije (prek 100 elementov) Interoperabilnost med orodji je kljub standardiziranemu zapisu XML še vedno nepopolna. Ne obstaja (še) 100-odstotna podpora izvajanju procesov BPMN 2.0.
Priložnosti	**Grožnje**
Standardizirana notacija à izboljšani medorganizacijski procesi (sodelovanje) Standardiziran zapis XML à večja neodvisnost od proizvajalcev in orodij za modeliranje in izvajanje Možnost izvajanja procesov à prehod na procesno orientirane IT-rešitve Veliko razpoložljivih orodij za modeliranje, dobra podpora industrije in možnost razširjanja notacije à široko uveljavljena in uporabna notacija	Kompleksnost standarda à redke in neskladne BPMN rešitve Kompleksnost notacije à slaba razumljivost sprejetosti (enostavnost uporabe, uporabnost) BPMN 2.0 v praksi Težave z interoperabilnostjo med orodji à odvisnost od specifičnega orodja ali proizvajalca Neskladno izvajanje BPMN 2.0 à nezmožnost prehoda na druga razvojna in izvajalna okolja

Iz tabele je razvidno, da BPMN 2.0 odpira številne priložnosti kakor tudi izzive na področju modeliranja in upravljanja poslovnih procesov. Neodvisno od kakovosti specifikacije BPMN 2.0 in nadaljnjih različic pa je uspeh BPMN odvisen tudi od kakovosti in skladnosti konkretnih implementacij (orodij za modeliranje in izvajanje procesov) specifikacije, njihove sprejetosti in nadaljnje podpore velikih IT-podjetij. V prispevku smo primerjali več priljubljenih rešitev na področju modeliranja in izvajanja procesov BPMN. Iz primerjav je razvidno, da lahko poslovni analitiki in razvijalci informacijskih rešitev za podporo poslovnih procesov izbirajo med različnimi konkurenčnimi rešitvami.

S predstavitvijo predstavnikov rešitev za modeliranje in izvajanje procesov (Signavio in Bonita) smo želeli podrobneje predstaviti trenutne funkcionalnosti in zmogljivosti rešitev na področju modeliranja in izvajanja procesov BPMN. Pri tem je pomembno poudariti, da smo rešitvi izbrali namenoma, zato ni mogoče posploševanje na preostale sorodne ali vse rešitve.

Iz prispevka lahko povzamemo, da uspešno upravljanje poslovnih procesov lahko zagotovimo z različnimi tehnikami, tehnologijami in pristopi. Glede na preostale notacije za modeliranje procesov je BPMN 2.0 nedvomno najbolj popolna in napredna notacija, ki poleg modeliranja omogoča še pretvorbo v izvedljive procese.

6 VIRI IN LITERATURA

[1] Kalpic, B. Business process modelling in industry - the powerful tool in enterprise management. *Computers in Industry.* 2002; 47(3): 299–318.

[2] Hammer, M., Champy, J., Champy, J. Reengineering the Corporation: A Manifesto for Business Revolution. Rev Upd. Harper Business; 2003.

[3] Breskvar, N. Modeliranje poslovnih procesov z uporabo Business Process Modelling Notation : magistrsko delo. 2009.

[4] Yvonne Lederer, A., Bariff, M., Benedict, T. Business Process Management Common Body Of Knowledge. CreateSpace Independent Publishing Platform; 2009.

[5] Netjes, M., Reijers, H. A., Aalst, W. M. P. Supporting the BPM Lifecycle with FileNet. Conference on Advanced Information Systems Engineering. 2006.

[6] Oba, M., Onoda, S., Komoda, N. Evaluating the quantitative effects of workflow systems based on real cases. Proceedings of the 33rd Annual Hawaii International Conference on System Sciences [Splet]. Maui, HI, USA; 2000 [pridobljeno 16. sep. 2012]. p. 1–7. Dosegljivo na: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=926852.

[7] Chang, J. F. Business Process Management Systems: Strategy and Implementation. 1st ed. Auerbach Publications; 2005.

[8] OMG. Business Process Model and Notation version 2.0 [Splet]. 2011 [pridobljeno 16. sep. 2012]. Dosegljivo na: http://www.omg.org/spec/BPMN/2.0/.

[9] Decker, G., Puhlmann F. Extending BPMN for Modeling Complex Choreographies. On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS [Splet]. Berlin, Heidelberg: Springer-Verlag; 2007. p. 24–40.

[10] OMG. BPMN Information Home [Splet]. 2010 [pridobljeno 16. sep. 2012]. Dosegljivo na: http://www.bpmn.org/.

[11] Silver, B. BPMN method and style. Aptos Calif.: Cody-Cassidy Press; 2009.

[12] Recker, J. BPMN Modeling – Who, Where, How and Why. BP Trends. 2008:1–8.

[13] Kannengiesser, U. Evaluation of BPMN tools. Proceedings of the International Workshop on the Management of Business Processes in Government. Brisbane, Australia; 2007. p. 19–32.

[14] Yan, Z., Reijers, H. A., Dijkman, R. M. An Evaluation of BPMN Modeling Tools. In: Mendling, J., Weidlich, M., Weske, M., eds. Business Process Modeling Notation [Splet]. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010 [pridobljeno 16. sep. 2012]. p. 121–128. Dosegljivo na: http://www.springerlink.com/index/10.1007/978-3-642-16298-5_12.
[15] Chinosi, M., Trombetta, A. BPMN: An introduction to the standard. Computer Standards & Interfaces. 2012; 34(1):124–134.
[16] Signavio. BPM + SaaS by Signavio [Splet]. 2012 [pridobljeno 16. sep. 2012]. Dosegljivo na: http://www.signavio.com/en.html.
[17] Jurič, M. B. A Hands-on Introduction to BPEL [Splet]. Oracle Technology Network. 2006 [pridobljeno 16. sep. 2012]. Dosegljivo na: http://www.oracle.com/technetwork/articles/matjaz-bpel1-090575.html.
[18] Huang, L. Leading Industry Analyst Firm Names BonitaSoft a "Cool Vendor" for Business Process Management [Splet]. 2011 [pridobljeno 16. sep. 2012]. Dosegljivo na: http://www.bonitasoft.com/blog/.
[19] Bonitasoft. Bonita Open Solution, Open Source BPM [Splet]. 2012 [pridobljeno 16. sep. 2012]. Dosegljivo na: http://www.bonitasoft.com/products/bonita-open-solution-open-source-bpm.
[20] Bonitasoft. Welcome to Bonita Open Solution Documentation [Splet]. 2012 [pridobljeno 16. sep. 2012]. Dosegljivo na: http://www.bonitasoft.org/wiki/doku.php.

▪

Gregor Polančič je docent na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Med njegova interesna področja spadajo tehnološki in netehnološki vidiki sistemov za komuniciranje, sodelovanje in upravljanje informacijskih procesov vključno z implikacijami sodobnih storitveno usmerjenih informacijskih rešitev na omenjena področja.

▪

Gregor Jošt je asistent in doktorski študent na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Med njegove interesne dejavnosti spadajo področja upravljanja poslovnih procesov, računalništvo v oblaku, storitveno usmerjena arhitektura in mobilne tehnologije

Podatkovne baze NoSQL

Aljaž Zrnec, Lovro Šubelj, Slavko Žitnik, Aleš Kumer, Marko Bajec
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Laboratorij za podatkovne tehnologije, Tržaška 25, 1000 Ljubljana
{aljaz.zrnec, lovro.subelj, slavko.zitnik, ales.kumer, marko.bajec}@fri.uni-lj.si

Izvleček

Kratica NoSQL pomeni Not Only SQL in jo pogosto povezujemo z novo skupino podatkovnih baz, ki so se pojavile kot odgovor na težave, s katerimi se srečujemo pri uporabi relacijski baz podatkov. Za podatkovne baze NoSQL je težko podati natančno opredelitev, lahko pa navedemo, katere so njihove pomembne lastnosti: nimajo opredeljene sheme, prožnost, drobljenje, asinhrona replikacija, pristop BASE namesto ACID in arhitektura brez skupne rabe.

Podatkovne baze NoSQL niso najboljša rešitev za reševanje vsakdanjih problemov, povezanih z upravljanjem s podatki. Nastale so na podlagi zahtev po visokih učinkovitosti in skalabilnosti v okolju, kakršno je svetovni splet. Testirali smo več vrst podatkovnih baz NoSQL, ki se razlikujejo med seboj glede na namen uporabe. Za izbrane scenarije smo primerjali zmogljivosti podatkovne baze ključ-vrednost, podatkovne baze vrste razširljivi zapis in dokumentno usmerjene podatkovne baze. Rezultate smo primerjali tudi z zmogljivostmi klasične relacijske podatkovne baze MySQL. Na podlagi ugotovitev smo podali priporočila, kdaj je smiselno uporabiti katero izmed naštetih rešitev.

Ključne besede: NoSQL, podatkovna baza, nerelacijska podatkovna baza, podatkovne baze ključ-vrednost, podatkovna baza vrste razširljivi zapis, dokumentne podatkovne baze.

Abstract

NoSQL Databases

NoSQL acronym stands for »Not Only SQL« and is associated with a new group of databases that have arisen in response to problems encountered in the use of relational databases. It is difficult to give a precise definition for NoSQL databases, but it can be argued that they are subject to the following important features: no defined schema, flexibility, shredding, asynchronous replication, BASE approach instead of ACID approach and architecture without sharing.

NoSQL databases do not represent the best solutions to solve everyday problems related to data management. They evolved from requirements for high performance and scalability in an environment such as the World Wide Web. We tested several types of NoSQL databases, which differ depending on their applications. For a given scenarios we tested the performance of the following NoSQL solutions: key-value database, extensible record database and document-oriented database. The results were also compared with the performance of classical relational database MySQL. Based on the findings we made the recommendation when it makes sense to use any of the above solutions.

Keywords: NoSQL, database, nonrelational database, key-value database, extensible record database, document-oriented database.

1 UVOD

Podatkovne baze NoSQL niso bile razvite z namenom popolne zamenjave relacijskih baz. Pri nekaterih rešitvah bi nas lahko toga shema relacijskih podatkovnih baz ovirala in so nerelacijske baze s svojo fleksibilnostjo prava izbira (Žlender, 2011). Prav tako relacijske baze niso najbolj primerne za reševanje problemov, pri katerih se srečujemo s količinami podatkov reda petabajtov. Podatkovne strukture nerelacijskih podatkovnih baz se lahko nahajajo na tisočih računalnikih in vsebujejo več tera- ali petabajtov podatkov. Pri tem podatkovne baze samodejno skrbijo za želeno stopnjo celovitosti in poskrbijo, da problemi s posameznimi strežniki ne ogrozijo celotne gruče.

V prispevku predstavimo ključne lastnosti posameznih vrst testiranih podatkovnih baz NoSQL in podamo glavne prednosti pred relacijskimi podatkovnimi bazami. V nadaljevanju primerjamo zmogljivosti štirih vidnejših predstavnikov podatkovnih baz NoSQL z uporabo štirih različnih scenarijev in rezultate primerjamo z zmogljivostmi klasične relacijske baze. V sklepnem delu prispevka navedemo področja, na katerih je uporaba rešitev NoSQL bolj smiselna od uporabe relacijskih podatkovnih baz.

2 KLJUČNE LASTNOSTI TESTIRANIH PODATKOVNIH BAZ NOSQL

Čeprav podatkovne baze NoSQL prinašajo veliko prednosti, to ne pomeni, da bi morali opustiti relacijske podatkovne baze (Leavitt, 2010; Knez, 2012;

Lavbič, 2012). Relacijske podatkovne baze in podatkovne baze NoSQL se med seboj bistveno razlikujejo in so izdelane za različne potrebe. Relacije in transakcije ACID so še vedno potrebne v določenih primerih, kot so borze in bančni sistemi. Podatki morajo biti namreč v teh primerih vedno razpoložljivi in pravilni, saj pri upravljanju s finančnimi podatki ne sme prihajati do napak. Na drugi strani poznamo aplikacije za socialna omrežja, pri katerih sta bolj kot celovitost pomembni skalabilnost in visoka razpoložljivost. Če se zadnja objava uporabnika pojavi na strani šele čez nekaj minut, to ni ključnega pomena na uporabo storitve.

V nadaljevanju so predstavljene ključne lastnosti podatkovnih baz NoSQL, ki so izpeljanke osnovnega tipa podatkovnih baz NoSQL – podatkovne baze ključ-vrednost. Namen testiranja je bil primerjati učinkovitosti posameznih podatkovnih baz NoSQL iz te skupine in jih primerjati z učinkovitostjo najbolj razširjene relacijske podatkovne baze MySQL. Prav zaradi tega v primerjavo nismo vključili tudi kakšnega primerka podatkovne baze NoSQL iz skupine baz NoSQL, ki temeljijo na grafih.

Podatkovne baze ključ-vrednost

Podatkovne baze ključ-vrednost veljajo za temelj, na katerem so zasnovane vse druge vrste podatkovnih baz NoSQL. Večina podatkovnih baz NoSQL namreč za shranjevanje uporablja mehanizem ključ-vrednost, čeprav morda to ni razvidno na prvi pogled.

Podatkovni model pri teh bazah je preprost in temelji na množici parov ključ-vrednost. Vsak ključ je unikaten in se preslika v pripadajočo vrednost. Pogosto je dolžina ključev, ki jih je treba shraniti, omejena na določeno število bajtov, medtem ko je glede vrednosti manj omejitev. Vrednosti so tako lahko poljubnega podatkovnega tipa, saj jih podatkovna baza vedno shranjuje kot objekt BLOB.

Shranjevanje podatkov v obliki ključ-vrednost je pravzaprav mehanizem, ki ga uporabljajo predpomnilniki. Predpomnilnik je hitri pomnilnik, ki vsebuje največkrat uporabljene podatke aplikacije, z namenom razbremeniti počasnejši disk. Podatkovne baze vrste ključ-vrednost so podobne predpomnilnikom, saj omogočajo hiter dostop do podatkov, ki so običajno majhne in preproste zbirke atributov in vrednosti.

Membase, Tokyo Cabinet in Redis so trije predstavniki implementacije podatkovnih baz vrste ključ-vrednost. Med omenjenimi je Redis, ki smo ga testirali, poseben primer, saj lahko ključ nastopa v obliki različnih podatkovnih struktur, kot so nizi, seznami in množice. Prav tako omogoča bogat nabor operacij za dostopanje do podatkov teh različnih vrst podatkovnih struktur.

Podatkovne baze vrste razširljivi zapis

Podatkovnim bazam vrste razširljivi zapis (Extensible record stores) rečemo tudi shrambe s širokimi stolpci (Wide column datastores), vendar večina avtorjev uporablja prvo poimenovanje (Cattell, 2010; Knez, 2012). Uvrščamo jih v skupino stolpično usmerjenih shramb (Column oriented datastores). Sem spadajo tudi stolpično usmerjene podatkovne baze (Column oriented databases), ki ne spadajo v skupino NoSQL. Gre namreč za relacijske podatkovne baze, pri katerih se operacije izvajajo nad stolpci in ne nad vrsticami.

Podatkovne baze vrste razširljivi zapis temeljijo na modelu BigTable (Chang, 2006), tj. Googlovem porazdeljenem sistemu za shranjevanje podatkov. Osnovni koncept njihovega podatkovnega modela je delitev tako vrstic kot stolpcev preko več vozlišč (vertikalno in horizontalno particioniranje).

Poleg Google BigTable, ki je temelj večini shramb vrste razširljivi zapis, sem spadajo še Cassandra, HBase in Hypertable. Med naštetimi smo testirali podatkovni bazi Cassandra in HBase.

Dokumentno usmerjene podatkovne baze

Dokumentne podatkovne baze so v bistvu podatkovne baze vrste ključ-vrednost, ki za shranjevanje podatkov uporabljajo bolj kompleksne podatkovne strukture, tj. dokumente. Beseda dokument v primeru dokumentne podatkovne baze pomeni nabor parov vrste ključ-vrednost, ki so strukturirani v obliko dokumenta. Povod za tako strukturo so podatki, ki danes ne nastopajo več v tako preprostih oblikah, kot so vrstice ali stolpci. Pogosto se namreč nahajajo v obliki datotek XML ali JSON (Andersen, 2010; Osborne, 2011; Terrastore, 2012), saj so te tehnologije visoko prenosljive, kompaktne in standardizirane. Namesto da dokumente XML in JSON preslikamo v relacijsko obliko, je veliko bolj smiselno uporabiti dokumentne podatkovne baze. Te so brez sheme, saj za dokumente XML/JSON ni vnaprej določenega formata, tako da je vsak dokument neodvisen od drugih.

Danes je na voljo nekaj različnih odprtokodnih dokumentnih podatkovnih baz, kot so npr. SimpleDB, Terrastore, najbolj obetavni med njimi sta

MongoDB in CouchDB. V okviru testiranja smo uporabili MongoDB.

3 TESTNO OKOLJE

Ker nismo imeli na razpolago dovolj fizičnih strežnikov, smo se odločili, da testno okolje namestimo v Amazonov oblak EC2. Amazon ponuja različne konfiguracije virtualnih strežnikov, in sicer od povsem osnovnih s 600 MB pomnilnika in eno računsko enoto, do takih z 68 GB pomnilnika in 26 računskimi enotami. Za potrebe testiranja smo izbrali konfiguracijo Large, ki ima te lastnosti:

- 7,5 GB pomnilnika,
- 4 računske enote EC2,
- 850 GB lokalne hrambe podatkov,
- visoka hitrost dostopa do trajne podatkovne shrambe,
- 64-bitna platforma.

Pojem računske enote EC2 je Amazon uvedel zato, da bi uporabnikom zagotovil približno enake lastnosti na zelo različni fizični opremi. Tako naj bi bila ena računska enota EC2 primerljiva z 1,0–1,2 GHz procesorjem 2007 Opteron ali 2007 Xeon (Amazon EC2, 2012). V času izvajanja testiranj je operacijski sistem v Large primerku virtualnega strežnika prepoznal dvojedrni 2,66 Hz procesor Intel Xeon, kar ustreza približno štirim računskim enotam EC2, kot jih ponuja Amazon.

Testiranje zmogljivosti je potekalo na štirih virtualnih strežnikih Large. Arhitektura virtualnih strežnikov Amazon EC2 omogoča, da so nekateri viri dodeljeni posameznemu strežniku, npr. procesorska moč in pomnilnik, nekateri pa se delijo med več strežnikov, npr. mrežna povezava in diskovni podsistem. Na svojih testnih strežnikih smo uporabili diskovni podsistem EBS (Amazon Elastic Block Store) (Amazon EBS, 2012). Do naprav EBS dostopa strežnik prek mrežnih povezav. Konfiguracija strežnika Large ima hitrost dostopa omejeno na 1Gbit/s. Predvidevamo lahko, da imajo baze, ki pogosto pišejo ali berejo s trdega diska, zaradi te lastnosti slabše izhodišče pri testiranju. Amazon sicer ponuja možnost povezovanja več enot EBS v polje RAID, s čimer je mogoče povečati zmogljivost vhodno-izhodnega sistema, vendar se v okviru izvajanja meritev nismo spuščali v to vrsto optimizacije.

Amazon za spremljanje uporabe virov na virtualnih strežnikih ponuja svojo storitev CloudWatch. Viri, ki smo jih spremljali na strežnikih, so uporaba CPU-ja, število pisanj in branj na disk ter količina podatkov, ki se bere ali piše. Kot klienta za izvajanje testov smo uporabili virtualni strežnik EC2 Large na lokaciji, na kateri so tekle tudi podatkovne baze. En sam klient je imel dovolj zmogljivosti, da je lahko generiral zahteve in ni predstavljal omejitve.

Programska oprema

Na virtualne strežnike smo namestili operacijski sistem Ubuntu 10.04 LTS, za katerega obstajajo tudi zagonske slike za Amazon EC2. V okviru primerjave zmogljivosti smo testirali te podatkovne baze NoSQL:

- Redis 2.4.3, Jedis JAVA klient 2.0.0 (podatkovna baza ključ-vrednost),
- MongoDB 2.0.1, MongoDB Java klient 2.6.5 (dokumentna podatkovna baza),
- Cassandra 0.8.7 podpora YCSB (podatkovna baza vrste razširljivi zapis) in
- HBase 0.90.4 (podatkovna baza vrste razširljivi zapis).

Za primerjavo rezultatov z zmogljivostjo relacijske podatkovne baze smo uporabili podatkovno bazo MySQL 5.1.41-3 Ubuntu 12.10, MySQL JDBC klient Connector/J 5.1.18.

Za zagon gruče s HBase in Cassandro smo uporabili odprtokodni projekt Whirr. Apache Whirr je skupek knjižnic, ki omogočajo preprost zagon podprtih podatkovnih baz na strežnikih Amazon. Trenutno podprta programska oprema je Cassandra, Hadoop, ZooKeeper, HBase, elasticsearch, Voldamort in Hama. Za zagon Mongo, Redis in MySQL gruč so bila izdelana lastna skripta.

Orodje za izvajanje testiranja

Za izvajanje testov smo uporabili odprtokodno orodje YCSB (Yahoo! Cloud System Benchmark) (Github, 2006). Pri Yahooju so orodje razvili prav z namenom iskanja najboljših rešitev za shranjevanje podatkov v porazdeljenih podatkovnih bazah. Že napisani odjemalci obstajajo za podatkovne baze MongoDB, Cassandra, HBase, Voldemort, Infinispan, kot tudi za podatkovne baze s podporo JDBC. Podpora za Redis prav tako obstaja, vendar smo morali dodati podporo za porazdelitev obremenitve na več strežnikov Redis. Prav tako lahko sami definiramo lastne delovne obremenitve, s čimer lahko bolje simuliramo svojo aplikacijo. Delovne obremenitve definirajo podatke, katere smo vnašali v podatkovne baze, in transakcije, katere smo izvajali v okviru meritev.

Nastavitve podatkovnih baz

Podatkovne baze smo namestili na štiri podatkovne strežnike. Naše poznavanje nastavitev za optimizacijo posameznih podatkovnih baz ni bilo enako za vse baze. Ker bi lahko z večjo optimizacijo posameznih baz nepravično vplivali na rezultate meritev, smo se odločili, da podatkovne baze nastavimo le toliko, da delujejo v porazdeljenem načinu na štirih strežnikih. Arhitekturo gruč smo poskušali izbrati tako, da so med seboj čim bolj primerljive. Še posebno Cassandra in HBase omogočata zelo natančno nastavitev želenih lastnosti. Določimo lahko, na koliko strežnikov se replicirajo določeni podatki, in s tem nastavljamo razmerje med hitrostjo zapisovanja in branja ter varnostjo podatkov.

Testni scenariji

Izbrane podatkovne baze se zelo razlikujejo po svoji arhitekturi in zaradi tega ni vsaka primerna za vse vrste uporabe (Žlender, 2011). Za ugotavljanje primernosti delovanja posameznih podatkovnih baz v določenih okoliščinah smo opredelili štiri scenarije uporabe, ki jih prikazuje tabela 1.

Tabela 1: **Uporabljeni testni scenariji**

Scenarij A	**Scenarij B**	**Scenarij C**	**Scenarij Č**
Scenarij A simulira pisanje velike količine podatkov v dnevnik.	Scenarij B simulira uporabo, pri kateri večino dostopov predstavlja branje. Zahtevani podatki so novejši. Scenarij predstavlja spletno stran, na kateri je zanimiva novejša vsebina.	Scenarij C predstavlja sporočilni sistem, pri katerem se podatki berejo in pišejo v približno enakem razmerju.	Scenarij Č uporablja za porazdelitev branj Zipfov zakon. Prevladujejo branja, pisanj je le en odstotek.
Delež branj 0 % Delež posodobitev 0 % Delež pisanj 95 % Delež iskanj 5 % Iskanje uporablja novejše podatke.	Delež branj 98 % Delež posodobitev 1 % Delež pisanj 1 % Delež iskanj 0 % Branje uporablja novejše podatke.	Delež branj 50 % Delež posodobitev 0 % Delež pisanj 50 % Delež iskanj 0 % Branje uporablja novejše podatke.	Delež branj 99 % Delež posodobitev 0 % Delež pisanj 1 % Delež iskanj 0 % Berejo se podatki, porazdeljeni po Zipfovem zakonu.

Struktura testnih podatkov

Zapisi v podatkovnih bazah so bili sestavljeni iz ključa in petnajstih besedilnih polj. Vsako besedilno polje je vsebovalo natančno sto znakov. Približna velikost enega zapisa je tako znašala 1,5 kB. Pri tem so lahko nastopila odstopanja, saj vsaka podatkovna baza shranjuje podatke v svojem formatu. Cassandra je npr. v okviru vsakega polja shranila tudi časovno oznako zadnjega popravka. V podatkovne baze smo pred izvajanjem testiranj shranili 10,000.000 zapisov.

4 ANALIZA

Začetno polnjenje s podatki

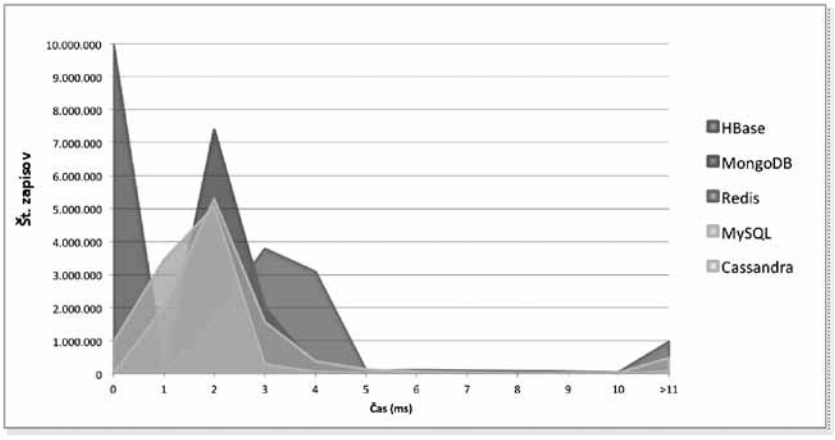
Začetni vnos podatkov kaže, kako hitro lahko izbrane podatkovne baze shranjujejo podatke. Tabela 2 prikazuje podatke o hitrosti vnosa podatkov v opazovane podatkovne baze. Najhitreje je 10 milijonov zapisov shranila gruča MySQL s precej višjim povprečnim številom operacij na sekundo kot vse druge podatkovne baze. Prav tako je pri tej bazi dosežen

Tabela 2: **Statistika vnosov v podatkovne baze**

	Čas izvajanja	**Štev. operacij/s**	**Povprečni čas pisanja zapisa**	**99 % zahtev (ms)**	**95 % zahtev (ms)**
Redis	66,6 min.	2734	7,025 ms	61	16
MongoDB	61,5 min.	2771	7,1165 ms	24	4
Cassandra	23,8 min.	7053	5,477 ms	71	9
HBase	19,4 min.	8908	4,3945 ms	0	0
MySQL	14,4 min.	11963	3,275 ms	7	3

najnižji povprečni čas zapisa. HBase je druga na seznamu, kljub temu da je gruča HBase nekoliko drugačna kot vse druge in ima na voljo samo tri vozlišča (eno vozlišče se uporablja izključno za izvajanje nadrejenih procesov) za shranjevanje podatkov. Zanimivo je tudi, da je HBase zapisal 99 odstotkov zahtev v manj kot 1 ms/zapis. To prikazuje tudi slika 1.

Podatkovna baza Redis je bila edina, ki ni shranila vseh zapisov. Približno 800.000 zapisov ni bilo uspešno shranjenih. Časovna omejitev povezave je pri Redisu nastavljena na precej majhno vrednost in je bila pri tako velikem številu zahtev pogosto prekoračena. Podobno zakasnitev smo opazili tudi pri podatkovni bazi HBase, pri kateri je bil najdaljši čas izvajanja ene zahteve kar 58 sekund.



Slika 1: **Porazdelitev časov zapisovanja pri začetnem vnosu podatkov**

Scenarij A

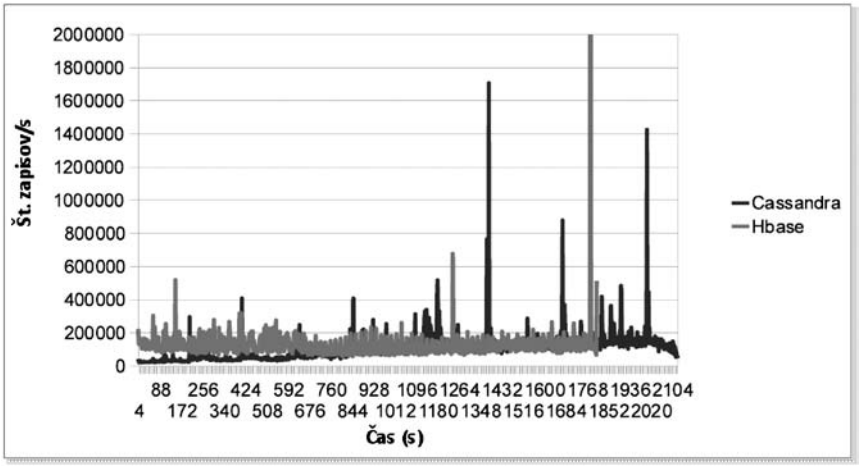
Prve meritve scenarija A smo izvajali tako, da je orodje YCSB maksimalno obremenilo strežnike. To se je kmalu izkazalo za problematično. Strežniki so po nekaj minutah postali preobremenjeni in število operacij na sekundo je padlo na 0. Zaradi tega smo se odločili, da število operacij na sekundo, ki jih sproži orodje YCSB, omejimo na 1500 in 2,700.000 operacij skupno. Če bodo podatkovne baze zmogle izvesti to število operacij, bo test končan po pol ure. Gonilnik JDBC, ki smo ga uporabili, ni dokončal nobene operacije iskanja, saj so ta vrnila preveč podatkov in je klientu zmanjkalo pomnilnika. Scenarija A z bazo MySQL nismo dokončali. Pri podatkovni bazi Redis prav tako nismo mogli dokončati testa. Strežniški procesi Redis so se konstantno ugašali in po nekaj minutah smo prekinili test. Rezultate podatkovnih baz HBase, Cassandra in MongoDB prikazuje tabela 3.

Tabela 3: **Statistika scenarija A**

	Čas izvajanja	**Štev. operacij/s**	**Povprečni čas iskanja (ms)**	**Povprečni čas pisanj (ms)**
Redis	/	/	/	/
MongoDB	97 min.	461	136,545	15,5
Cassandra	36,5 min.	1232	91,578	3,041
HBase	30,2 min.	1493	109,832	0,162
MySQL	/	/	/	/

Pri bazi HBase kljub dodatnim iskanjem ni bilo opaziti nobenih zakasnitev pri hitrosti zapisov. Po drugi strani se je pri podatkovni bazi MongoDB zelo podaljšal čas zapisov, čeprav smo izvajali precej manj pisanj kot pri začetnem vnosu podatkov. Cassandra je bila najhitrejša pri iskanjih, pri pisanju pa je bila približno devetnajstkrat počasnejša od baze HBase, vendar še vedno petkrat hitrejša od najpočasnejše baze MongoDB.

Slika 2 prikazuje povprečno hitrost iskanja v določenem trenutku izvajanja testa. Vidi se, da Cassandra na začetku vrača rezultate počasneje kot HBase, kasneje pa se približno izenačita. Čeprav ima Cassandra več trenutkov, ko se poveča čas iskanja, je skupni povprečni čas še vedno manjši.



Slika 2: **Povprečna hitrost iskanja**

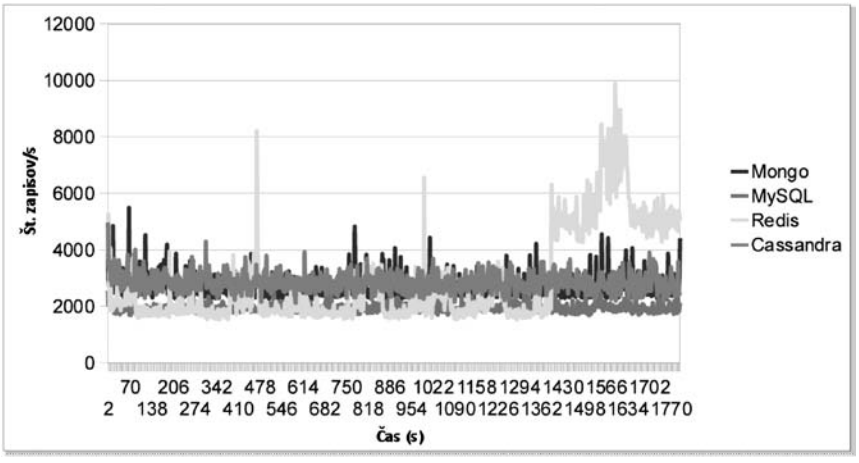
Scenarij B

Enako kot pri scenariju A smo tudi pri scenariju B omejili skupno število operacij na 2,700.000 in 1500 operacij na sekundo. Vse podatkovne baze razen HBase so s scenarijem opravile v optimalnem času. Pri HBase nas je zelo presenetil slab čas branj, ki jih je bilo v scenariju B kar 98 odstotkov. Kot je razvidno iz tabele 4, je HBase v povprečju potreboval kar 19 ms za operacijo branja. Interno HBase operacije branja razume kot iskanje. HBase nima indeksov, ki bi omogočali dostop do posamezne vrstice ali stolpca. Najmanjša enota je blok v datoteki HFile, katero moramo nato preiskati, da dobimo želeni podatek. Kljub temu da branja trajajo nekaj dlje, HBase še vedno zapisuje najhitreje.

Tabela 4: **Statistika scenarija B**

	Čas izvajanja	Štev. operacij/s	Povprečni čas posodobitve (ms)	Povprečni čas pisanj (ms)	Povprečni čas branj (ms)
Redis	30 min.	1500	2,768	6,281	2,87
MongoDB	30 min.	1500	4,153	4,291	2,62
Cassandra	30 min.	1500	2,3	2,488	2,84
HBase	85 min.	529	0,024	0,047	19,22
MySQL	30 min.	1500	2,016	2,165	1,94

Slika 3 prikazuje razmerje med podatkovnimi bazami pri branju. MySQL in Redis sta na začetku testiranja še nekako enakovredna, vendar je Redis proti koncu najhitrejši od vseh. Redis tudi pri tem testu ni uspešno dokončal vseh operacij. Neuspešno je bilo izvedenih 3000 pisanj in posodobitev ter približno 460.000 branj.



Slika 3: **Povprečna hitrost branja**

Scenarij C

Pri scenariju C smo skupno število operacij zmanjšali na 1,500.000, število operacij na sekundo je ostalo nespremenjeno, to je 1500. Ponovno smo imeli veliko težav s podatkovno bazo Redis. Nikakor nam ni uspelo uspešno končati izvajanja meritev. Redis pri velikem številu pisanj ne zmore dovolj hitro shranjevati podatkov na disk, zaradi česar se procesi ugasnejo. HBase trpi za enako hibo kot v scenariju B, branja podatkov niso dovolj hitra in tudi v tem scenariju jih je bilo veliko. Pri Cassandri se lepo vidi, kako večje število pisanj vpliva na čas branj. Branja vzamejo skoraj osemkrat več časa kot pri scenariju B.

Tabela 5: **Statistika scenarija C**

	Čas izvajanja	Štev. operacij/s	Povprečni čas pisanj (ms)	Povprečni čas branj (ms)
Redis	/	/	/	/
MongoDB	16,6 min.	1500	2,981	2,596
Cassandra	30 min.	839	2,183	21,157
HBase	45 min.	551	0,106	36,011
MySQL	16,6 min.	1500	4,768	3,788

Časi branj posameznih podatkovnih baz, ki jih prikazuje tabela 5, se precej razlikujejo. Eno izmed mogočih razlag ponuja tudi vpogled v to, koliko podatkov se prebere iz diska. Podatke smo pridobili iz storitve Amazon CloudWatch za posamezna vozlišča v gruči in jih sešteli po podatkovnih bazah in časih, ko se je izvajal scenarij C.

- MySQL: 68 kilobajtov
- MongoDB: 99 megabajtov
- Cassandra: 26 gigabajtov
- HBase: 62,5 gigabajtov

Razlike so skoraj neverjetne. MySQL se v malo več kot 16 minutah in skoraj 800.000 branjih skoraj ne dotakne diska – vse zahteve se postrežejo iz pomnilnika –, medtem ko HBase prebere 62 gigabajtov.

Scenarij Č

Pri scenariju Č so se vse podatkovne baze razen HBase zelo dobro odrezale, zato smo povečali število operacij na sekundo v testu na 2500. Pri testiranju HBase se test ni izvedel v celoti niti pri 1500 operacijah na sekundo. Redis je kljub dobrim rezultatom in s časi, primerljivimi z drugimi, neuspešno opravil približno 10 odstotkov operacij. Z MySQL, Redis in MongoDB smo naknadno izvedli še dodaten test pri večjih obremenitvah (rezultati teh testov niso prikazani v tabeli). Največje število operacij scenarija Č je zmogel MySQL (kar 9166), sledi Redis s 4650 in MongoDB s 3577 operacijami. Rezultate testiranja prikazuje tabela 6.

Tabela 6: **Statistika scenarija Č**

	Čas izvajanja	Štev. operacij/s	Povprečni čas pisanj (ms)	Povprečni čas branj (ms)
Redis	18 min.	2500	5,082	2,017
MongoDB	18 min.	2500	4,424	2,651
Cassandra	18 min.	2500	2,229	3,265
HBase	/	/	/	/
MySQL	18 min.	2500	2,252	1,901

5 SKLEP

Rezultati kažejo, da pri strukturiranih podatkih, preprostih poizvedbah in indeksu samo na primarnem ključu, nerelacijske baze ne uspejo slediti hitrosti podatkovne baze MySQL. Ob tem je treba poudariti, da sta bila Redis in MySQL v privilegiranem položaju, saj je za porazdelitev podatkov na pravo vozlišče skrbel odjemalec in ne podatkovna baza. MongoDB, Cassandra in HBase imajo vse vgrajene algoritme, ki podatke samodejno porazdelijo po vozliščih. Pri sto ali več vozliščih bi bil ročni nadzor porazdeljevanja podatkov tako rekoč nemogoč. Vendar je pri tako velikem številu vozlišč izpad vozlišča zelo verjeten dogodek. Zato bi bilo tudi zanimivo primerjati, kako se hitrosti poizvedb in zapisovanja spremenijo v primeru izpada vozlišča in kako hitro v takem primeru podatkovne baze podatke prenesejo na druga vozlišča.

V splošnem lahko kot prednosti podatkovnih baz NoSQL izpostavimo njihovo izjemno skalabilnost in razpoložljivost ter nizko ceno. Najprimernejše situacije za uporabo podatkovnih baz NoSQL so: preprost podatkovni model, prilagodljivost je pomembnejša od strogega nadzora nad opredeljeno podatkovno strukturo, zahtevana visoka zmogljivost, stroga podatkovna celovitost ni nujna in računalništvo v oblaku.

Glede na rezultate lahko ugotovimo, da je podatkovna baza ključ-vrednost Redis primerna za uporabo v primerih, ko imamo večinoma opravka z branjem velikih količin podatkov (scenarij A). Podobno velja tudi za Cassandro, ki spada v množico podatkovnih baz vrste razširljivi zapis in dokumentno usmerjeno podatkovno bazo MongoDB. Hbase (baza vrste razširljivi zapis) lahko priporočimo v primeru, ko imamo opravka z velikim številom pisanj podatkov v podatkovno bazo (scenarija A in C).

Relacijska baza MySQL se je odrezala presenetljivo dobro pri branju podatkov (scenarija B in Č), slabše pa pri pisanju, kar je lepo razvidno iz scenarija C.

Iz tega lahko povzamemo, da torej ne gre za vprašanje, ali so podatkovne baze NoSQL boljše od relacijskih, ampak predvsem za vprašanje, kdaj je smiselno uporabiti baze vrste NoSQL. Odgovor je, kadar se soočamo z zahtevami za poizvedbe po obsežnih podatkovnih naborih z veliko hitrostjo izvajanja poizvedb. Pri tem nastopijo baze NoSQL. Gre za podatkovne nabore, med katere spadajo npr. spletni dnevniki, transakcije nakupov, proizvodni podatki iz naprav na tekočem traku, znanstvene zbirke podatkov za izvajanje različnih analiz itd., ki se kopičijo v velikem številu vsako sekundo in za njihovo shranjevanje in obdelavo relacijska podatkovna baza ni dovolj zmogljiva in hitra rešitev.

Po drugi strani pa je tradicionalna relacijska podatkovna baza še vedno najboljša rešitev, če potrebujemo konsistentne podatke, obstojna pisanja, transakcije ACID in imamo zapletene podatkovne strukture in razmerja. Področja, ki so primerna za relacijske podatkovne baze, lahko vključujejo katere koli zapletene poslovne aplikacije, zlasti finančno usmerjene aplikacije, pri katerih so celovitost transakcij ter skladnost in trajnost podatkov nujne zahteve. Primeri so tudi obdelave OLTP, pri katerih še vedno zmaguje kombinacija kakovosti podatkov in zmogljivosti dobro načrtovanega SUPB-ja.

6 VIRI IN LITERATURA

[1] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., Gruber, R. E. (2006). Bigtable: A Distributed Storage System for Structured Data, OSDI 2006.

[2] Github (2012). Yahoo! Cloud Serving Benchmark (YCSB), Dostopno 10. 1. 2012 na https://github.com/brianfrankcooper/YCSB/wiki.

[3] Knez, N. (2012). Analiza podatkovnih baz NoSQL in izdelava odločitvenega modela za izbiro med relacijskimi in NoSQL podatkovnimi bazami. Diplomsko delo. Univerza v Ljubljani, Fakulteta za računalništvo in informatiko.

[4] Leavitt, N. (2010). Will NoSQL Databases Live Up to Their Promise?, IEEE Xplore digital library, Vol. 43, No. 2, pp. 12–14.

[5] Stonebraker, M. (2010). SQL databases vs. NoSQL databases, ACM Digital Library, Vol. 53, No. 4.

[6] Žlender, R. (2011). Primerjava zmogljivosti nerelacijskih podatkovnih baz. Diplomsko delo. Univerza v Ljubljani, Fakulteta za računalništvo in informatiko.
[7] Lavbič, D., Vasilecas, O., Rupnik, R. (2012). Ontology-based Multi-Agent System to support business users and management. Technological and Economic development of Economy, št. 16 (2), str. 327–347.
[8] Andersen, C., Lehnardt, J., Slater, N. (2010). CouchDB: The Definitive Guide, 1st ed., Sebastopol: O'Reilly Media, Inc., pogl. 2.
[9] Osborne, R. (2011). »Playing around with MongoDB and MapReduce functions« Dostopno 20. 12. 2011 na: http://rickosborne.org/blog/2010/02/playing-around-with-mongodb-and-mapreducefunctions/.
[10] Cattell, R. (2010). Relational Databases, Object Databases, Key-Value Stores, Document Stores, and Extensible Record Stores: A Comparison, Dostopno 16. 1. 2012 na: http://www.odbms.org/download/Cattell.Dec10.pdf.
[11] Terrastore (2011). Dostopno 15. 12. 2011 na: http://code.google.com/p/terrastore/.
[12] Amazon EBS (2012). Dostopno 30. 1. 2012 na: http://aws.amazon.com/ebs/.
[13] Amazon EC2 FAQs (2012). Dostopno 30. 1. 2012 na: http://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and_why_did_you_introduce_it.

▪

Aljaž Zrnec je magistriral leta 2002 na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Leta 2006 je doktoriral s področja konstruiranja metodologij. Zaposlen je v Laboratoriju za podatkovne tehnologije kot asistent za področje podatkovnih baz. Na raziskovalnem področju se ukvarja s konstruiranjem metodologij, podatkovnimi bazami NoSQL in računalništvom v oblaku. Je avtor ali soavtor številnih prispevkov v strokovnih in znanstvenih publikacijah.

▪

Lovro Šubelj je mladi raziskovalec in asistent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Raziskovalno se ukvarja z analizo realnih omrežij, natančneje z odkrivanjem skupin vozlišč v velikih kompleksnih omrežjih.

▪

Slavko Žitnik je doktorski študent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani in je hkrati zaposlen kot mladi raziskovalec iz gospodarstva v podjetju Optilab, d. o. o. Raziskovalno se ukvarja predvsem s procesiranjem besedil, bolj natančno z razpoznavanjem entitet in povezav med njimi, z uporabo metod iz strojnega učenja in semantičnih tehnologij.

▪

Aleš Kumer je asistent na Fakulteti za računalništvo in informatiko Univerze v Ljubljani.

▪

Marko Bajec je izredni profesor na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer poučuje dodiplomske in podiplomske predmete s področja razvoja informacijskih sistemov in podatkovnih baz. Raziskovalno se ukvarja z metodami in pristopi k snovanju in razvoju informacijskih sistemov, obvladovanjem informatike ter v zadnjih letih predvsem s podatkovnimi tehnologijami za predstavitev, analizo in vizualizacijo podatkov. Leta 2009 je ustanovil Laboratorij za podatkovne tehnologije ter prevzel njegovo vodenje. Je član številnih domačih in tujih združenj, komisij in odborov. V okviru fakultete je vodil več aplikativnih in raziskovalnih projektov. Svoje raziskovalne rezultate in dosežke iz prakse redno objavlja v domačih in mednarodnih znanstvenih in strokovnih krogih.

Heterarhije in relacijski podatkovni modeli

Tomaž Lajovic, Iztok Lajovic
KRES Kreativni sistemi, d. o. o., Ljubljana
tomaz.lajovic@kres-ks.si; iztok.lajovic@kres-ks.si

Izvleček

Pri razvoju poslovnih rešitev z lastno informacijsko platformo Panorama smo uspešno presegli vrsto omejitev tradicionalnih relacijskih modelov. V standardni relacijski bazi smo združili prednosti relacijskega na eni in objektnega modela na drugi strani: preglednost in učinkovitost relacijskega smo nadgradili z objektnim načinom oblikovanja informacijskega prostora.

Gradnja relacijskih modelov je praviloma osredinjena na zadovoljitev minimalnih potreb dejanskih poslovnih procesov, in sicer predvsem v luči zahtevanih poročil oziroma poizvedb. Naš pristop se rahlo razlikuje, saj je primarna zahteva pri gradnji podatkovnega modela popolnost opisa (mogočih) medsebojnih odnosov, nato pa podatkovni model dopolnimo z izvedenimi (posrednimi) povezavami, s katerimi je uporabniku omogočen neposreden vpogled v zanj zanimive podatke o opazovanem predmetu. Pri tem smo naleteli na zanimiv izziv kot posledico različnih vrst in vsebine podatkov, ki so lahko uporabljeni kot atribut opazovanih predmetov – izziv heterarhij oziroma uvrščenosti v množico hierarhij, ki definirajo opazovani predmet. Gre za vprašanja interpretacije lastnosti opazovanega predmeta, ki so dejansko lastnosti predmetov, katerim pripada sam, na eni strani ter lastnosti predmetov, ki pripadajo opazovanemu predmetu, na drugi strani.

S preprostim primerom predstavimo kritične zahteve in želeni rezultat ter izhodišča za ureditev zahtevanih podatkovnih struktur, s katerimi podpremo upravljanje heterarhičnih struktur in njihovo pravilno interpretacijo.

Ključne besede: podatkovne baze, objektno-relacijsko mapiranje, sklici, razvejani sklici, hierahije, heterarhije, upravljanje matičnih podatkov.

Abstract

Heterarchies and Relational Data Models

In the development of business solutions with our own IT platform Panorama we have successfully overcome limitations of the traditional relational models. In a standard relational database, we combine the advantages of relational on the one hand and of object-oriented models on the other: the transparency and efficiency of the relational model has been upgraded with an object-oriented method of creating an information space.

Construction of the relational models is generally focused on meeting the minimal needs of real business processes, especially in the light of the required reports or queries. Our approach is slightly different because the primary requirement for building the data model is a complete description of the (potential) relationships and then the data model is amended with derivative (indirect) relations, allowing the user direct insight into the information on the observed object. In doing so, we came across an interesting challenge as a result of different types of content and data that can be used as an attribute of the observed objects – the challenge of heterarchies or multitude of hierarchies, which define the observed object. It is a question of interpretation of the properties of observed objects, that are properties of objects that itself (hierarchically) belongs to, on the one hand and of the properties of objects that (hierarchically) belong to the observed object.

A simple example is used to present critical requirements and the desired outcome as well as underlying principles for the regulation of the required data structures to support management of heterarchical structures and their proper interpretation.

Keywords: databases, object-relational mapping, links, branched links, hierachies, heterarchies, master data management.

1 UVOD

Uporaba enotnih podatkovnih modelov za več informacijskih namenov je zaradi sprotne implicitne medsebojne sinhronizacije vedno bolj aktualna. Dostopnih virov podatkov je namreč vedno več in medsebojno usklajevanje in/ali dostopanje do vsakega vira posebej je nekako neskladno s pričakovanji, ki jih gojimo do informatizacije.

Pri razvoju poslovnih rešitev z lastno informacijsko platformo Panorama smo v podjetju KRES uspešno presegli vrsto

omejitev tradicionalnih relacijskih modelov. V standardni relacijski bazi smo združili prednosti enostavnosti vzdrževanja relacijskega na eni in sistematičnosti gradnje objektnega modela na drugi strani: preglednost in stabilnost relacijske baze smo nadgradili z opisovanjem dejanskosti prilagodljivim objektnim načinom podatkovnega modeliranja.

Tako smo omogočili podatkovno modeliranje neodvisno od končnega cilja – podatkovne strukture naj čim bolj verno odslikavajo realnost, dejanske informacijske potrebe pa zadovoljujemo z opisom zahtev na metanivoju nad enotnim podatkovnim modelom. Še več, zaradi zahtevnosti upravljanja matičnih podatkov (angl. master data management) in z njim povezanega usklajevanja zatečenih oziroma obstoječih podatkovnih baz smo želeli še najmanj enostavno dopolnjevanje enotne podatkovne strukture ter obvladovanje posameznih časovnih in jezikovnih različic atributov predmetov. Vsega naštetega sicer še nismo izpolnili, v tem članku pa smo se osredinili na nekatere probleme, povezane s hierarhijami.

Panorama nastaja od leta 1984 kot nadgradnja projekta izgradnje centralnega podatkovnega slovarja za vse lastne aplikacije tedanjega podjetja Iskra Delta, ki ga je razvijala skupina pod vodstvom avtorja Panorame in soavtorja tega članka, Iztoka Lajovca. Izhaja iz implementacije konceptov asociativnega razmišljanja, kar je neodvisno v svojem delu iz okvirno istega časa opisal Minsky (Minsky, 1986). Razvoj misli in razvoja Panorame je bil večkrat objavljen (Lajovic, 2002; Lavbič, Lajovic & Krisper, 2010), ta članek pa temelji na prispevku, ki sva ga avtorja predstavila na Dnevih slovenske informatike 2012 (Lajovic & Lajovic, 2012).

V nadaljevanju z nekaj primeri predstavimo bistvene prednosti panoramskega objektno-relacijskega pristopa. Na kratko so to štiri:

- **pregledno urejene datoteke metapodatkovnega nivoja**, ki služijo kot podatkovni slovar in vodilo upravljanja s podatki;
- **dosledna dvosmernost povezav** je omogočena s t. i. *vezmi*, ki vsakemu *sklicu* (atribut v tabeli A, ki je posamična vrednost iz tabele B) priredi *seznam* (atribut v tabeli B, ki je seznam vrednosti iz tabele A), in s čimer je omogočena hitra in nedvoumna interpretacija vseh povezav na vsebinskem nivoju;
- **podpora razvejanim povezavam**, ki omogočajo enotnost interpretacije inherentno raznovrstnih medsebojnih odnosov, in sicer v obliki:
  - *razvejanih sklicev*: atribut v tabeli M, ki je posamezna vrednost iz katere koli izmed tabel N, O, P, ter
  - *skupnih seznamov*: atribut v tabeli W je nabor vrednosti iz tabel X, Y, Z;
- **omogočanje *oddaljenih sklicev* in *seznamov***, s katerimi se poenostavi pridobivanje podatkov v sicer kompleksnem visoko strukturiranem podatkovnem modelu.

V sklepu nakažemo smer nadaljnjega razvoja misli o heterarhijah in njene implementacije v panoramskih strukturah.

2 IZHODIŠČA ZA GRADNJO PANORAMSKIH MODELOV

2.1 Osnovno o Panorami

Panorama je podatkovna platforma, ki uporabniku omogoča prosto prilagajanje in nadgrajevanje podatkovnega modela. Vgrajeno ima relacijsko bazo, skladno z ANSI SQL-92 specifikacijo, tako da se vse strukture in datoteke zapisujejo v standardnih relacijskih tabelah s pripadajočimi indeksi, uporabnik pa vse upravlja prek grafičnega vmesnika, ki ne zahteva znanja programskih jezikov.

Kot dinamični sistem sproti in avtomatsko generira vse potrebne tabele, ukaze, poti in interpretacije, in sicer z uporabo slovarskih tabel, s katerimi so opisana vodila za sprotno avtomatsko pripravo potrebnih ukazov SQL. Na eni strani s tem dosežemo, da po kompleksnih podatkovnih strukturah lahko potuje tudi programiranja nevešč uporabnik, na drugi strani pa omogočamo tudi nadaljnjo gradnjo za realizacijo jutrišnjih podatkovnih in informacijskih zahtev brez vplivanja na že obstoječi in delujoči sistem.

Primarno je Panorama namenjena poslovnim aplikacijam in je izvedena v standardni arhitekturi odjemalec/strežnik v okoljih Windows. Dostop z mobilnih platform oziroma prek spleta nameravamo podpreti omejen nabor funkcionalnosti, in sicer predvsem v smislu učinkovite podpore pregledovanja ter ažuriranja podatkov na nivojih II in VI (nivoji so opisani v nadaljevanju tega razdelka).

Večina postopkov priprave ukazov za bazo podatkov in celotno grafično okolje se izvaja na strani odjemalca, na strani strežnika pa se nahajajo datoteke ter izvajajo ažuriranja in obdelave zahtev posameznih odjemalcev. Za upravljanje uporabniških pravic in preferenc posameznikov je uporabljen inovativni

samoreferenčni sistem pravil, ki omogoča učinkovito obvladovanje in izmenjavo strukturiranih podatkov.

Tabele in datoteke panoramskega modela so po funkcijah razvrščene v te ravni:

I. slovarska raven – definicija podatkovnega modela in poizvedb;
II. podatkovna raven – podatki in rezultati poizvedb;
III. pristopna raven – upravljanje s pravicami dostopa, overjanje in avtorizacija uporabnikov ter uvozi in sinhronizacije z zunanjimi viri;
IV. oblikovalni nivo – prikazi, filtri, uporabniške nastavitve, izpisi in izvozi;
V. evidenčni nivo – beleženje dostopov in uporabe;
VI. dokumentni nivo – dokumenti in datoteke;
VII. arhivski nivo – arhivske datoteke celotne baze.

Vse datoteke nivojev I do V so del enotne baze, medtem ko so datoteke dokumentnega in arhivskega (VI in VII) nivoja ločene, in sicer so dokumenti in datoteke shranjeni v obliki posebne odvisne baze, arhivi pa so povsem neodvisne datoteke (posnetki stanja celotne baze v danem trenutku). V članku so predstavljene nekatere rešitve slovarskega (I) in podatkovnega (II) nivoja, ki sta temelja vsebinske prilagodljivosti in enostavnosti upravljanja panoramskih modelov.

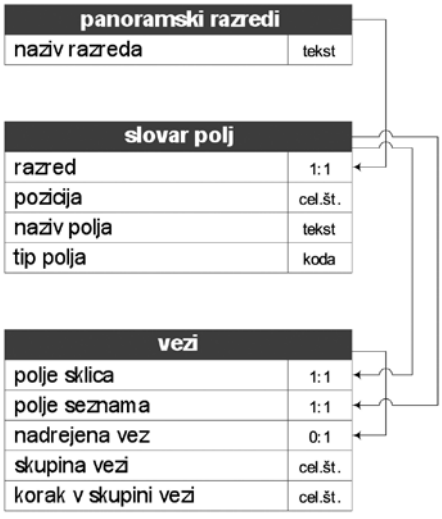
2.2 Upravljanje podatkovnega modela

Tabele podatkovnega modela (nivo II) so opredeljene z zapisi v treh slovarskih (nivo I) tabelah:

1. *panoramski razredi* s poimenovanji tabel;
2. *slovar polj* z navodili za oblikovanje stolpcev posameznih podatkovnih tabel;
3. *vezi*, v katerih so podrobno opisane vse povezave med polji tabel.

Medsebojna razmerja treh temeljnih tabel slovarskega nivoja so predstavljena shematično (slika 1), podrobnejša legenda pa se nahaja v nadaljevanju (slika 2).

Način gradnje podatkovnega modela z opisom logičnega modela zahteva enoličnost interpretacije vseh parametrov fizičnega modela, saj se ta zgradi sam v skladu s pravili, ki so kot temeljna pravila vgrajena v program. Zapisi v tabelah slovarskega nivoja popolnoma izpolnjujejo ta kriterij, istočasno pa zagotavljajo tudi nedvoumnost interpretacij vsebine povezav prek vezi oziroma parov polj *sklic* in *seznam*:



Slika 1: **Panoramske tabele slovarskega nivoja (poenostavljena shema)**

- 'naziv polja' za tip *sklic:*
  naziv atributa predmeta,
- 'naziv polja' za tip *seznam:*
  vsebina povezanosti predmeta s predmeti, navedenimi v pripadajočem seznamu.

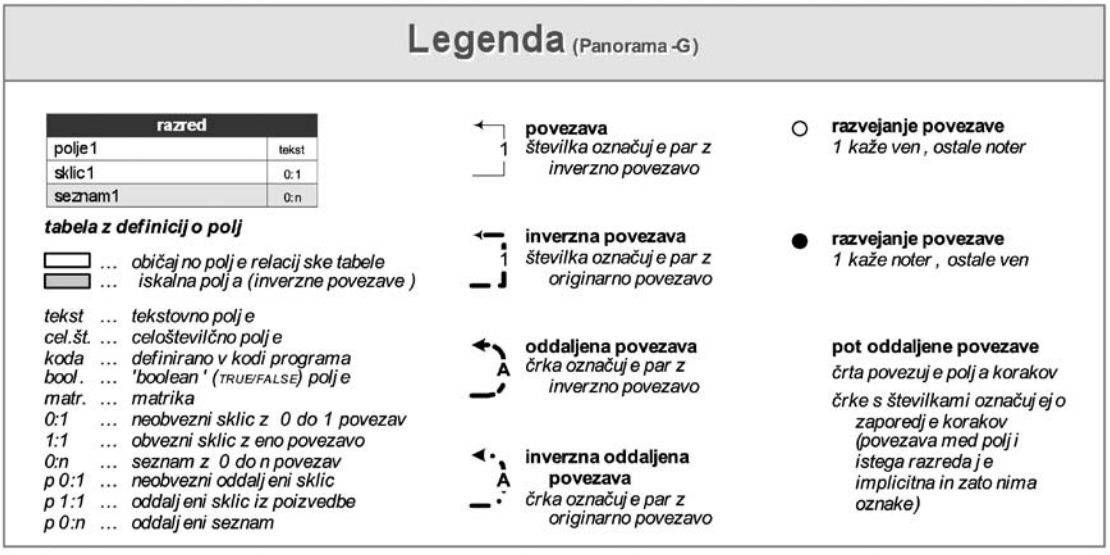
Ob tem namena slovarske datoteke *vezi* še ne moremo razložiti v celoti, saj za opis običajnih relacijskih modelov povsem zadostujeta prvi dve datoteki. Prav slovarski datoteki se bomo v nadaljevanju posebej posvetili; nepogrešljiva je namreč tako za gradnjo razvejanih relacijskih struktur kot za predlagani pristop k formalizaciji heterarhičnih struktur.

V zvezi z normalizacijo oziroma veljavnostjo posameznih normalnih form je treba poudariti, da Panorama sama zase ne zagotavlja katere koli stopnje normalizacije baze – prek samoumevne 1NF (Date, 2012) – ki jo zgradimo z njo. Seveda pa je – poleg tega, da že sama vodi razmišljanje ob modeliranju v bolj »normalno« smer – lahko s svojimi slovarskimi datotekami v veliko pomoč pri za normalizacijo zahtevani analitiki in posledični optimizaciji podatkovnih baz.

3 PRIMERI PANORAMSKIH PODATKOVNIH STRUKTUR

S Panoramo kot informacijsko platformo smo želeli uporabnikom omogočiti čim večjo svobodo pri oblikovanju lastnih podatkovnih modelov. Panoramske podatkovne strukture so omejene na neposredno podprte v implementaciji samega programa, pri čemer smo uvedli tudi nekaj posebnosti, ki so shematično s preprostimi primeri predstavljeni v nada-

ljevanju. Za namene predstavitve panoramskih podatkovnih struktur smo razvili poseben shematski zapis, imenovan Panorama-G, katerega glavni elementi so predstavljeni v legendi (slika 2). V takem shematskem zapisu bodo predstavljeni vsi primeri v nadaljevanju.



Slika 2: **Legenda shematskega zapisa Panorama-G**

Zaradi preglednosti smo ob shemah tabelarično navedli tudi pripadajoče zapise v slovarski tabeli »vezi« (slika 1), v kateri smo za lažje razumevanje dodali zadnji stolpec »tip vezi«, ki ga sicer ni v tabeli. Vrednosti v tem stolpcu so lahko:

- *elementarna* – osnovni gradnik podatkovnega modela;
- *oddaljena – temeljna* – izvedeni gradnik metapodatkovnega modela, in sicer začetna oziroma končna (v primeru oddaljenega sklica je to ena in ista) vez poti skozi podatkovni model;
- *oddaljena – vmesna* – izvedeni gradnik metapodatkovnega modela, in sicer vmesna vez na poti skozi podatkovni model;
- *korak* – opis poti od začetne temeljne oddaljene vezi do končne temeljne oddaljene vezi s koraki prek elementarnih vezi podatkovnega modela.

3.1 Sklic in seznam

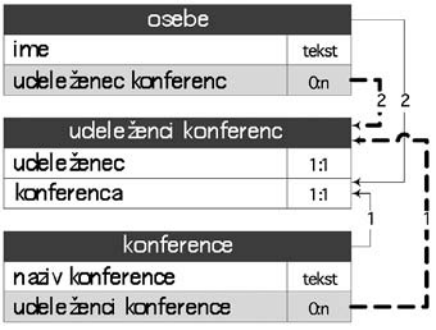
V standardnem relacijskem modelu so *sklici* povezave med dvema tabelama, pri čemer se vsebina polja v prvi tabeli polni s predmeti iz druge tabele. Ker ima tudi obratna pot svojo specifično informacijsko vrednost – kateri predmeti iz prve tabele se sklicujejo na posamezne predmete iz druge tabele in s kakšnim namenom –, smo uvedli izrecno deklaracijo *seznamov*: vsako polje *sklica* ima na drugi strani povezave, v razredu, na katerega se sklicuje, polje navedbe pripadajočega *seznama* (slika 3, tabela 1). Vsebina *seznama* je, skladno s standardi, ki veljajo v relacijskih bazah, rezultat poizvedbe v tabeli *sklica*: polje tipa *seznam* namreč obstaja le kot vodilo v slovarskih datotekah, tabela, ki ji pripada *seznam*, pa nima ustreznega stolpca.

Slika 3: **Povezava [ena : mnogo] z izrecno deklaracijo *seznama* v razredu, na katerega kaže *sklic***

Tabela 1: **Vezi k sliki 3**

VEZI	polje sklica	polje seznama	nadrejena vez	skupina vezi	korak v skupini	tip vezi
1	barva las	osebe s to b. las				*elementarna*

Na prvi pogled ta pristop prinaša zgolj odstop od obstoječih normalnih form, vendar pa gre pri tem dejansko za posebno dopolnitev, ki ponuja množico dodatnih možnosti in optimizacij. Ena izmed njih je lahko razvidna že iz preprostega primera povezave, pri kateri se poljubno mnogo predmetov prvega razreda povezuje s poljubno mnogo predmeti drugega (slika 4, tabela 2). Če nas zanima, kateri predmeti preostalih razredov (danega podatkovnega modela) se sklicujejo na predmete razreda »osebe«, nam polje tipa *seznam* v definiciji razreda »osebe« navede, da se od obeh ostalih dveh razredov v našem modelu nanj lahko sklicujejo le predmeti razreda »udeleženci konferenc«, zato bomo pri zadani poizvedbi preiskali zgolj ta razred v zadevnem stolpcu (z uporabo pripravljenega indeksa) in preprosto izpustili drugega (ter morebitne preostale).

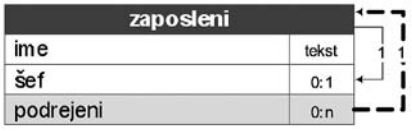


Slika 4: **Povezava [mnogo : mnogo]**

Tabela 2: **Vezi k sliki 4**

VEZI	polje sklica	polje seznama	nadrejena vez	skupina vezi	korak v skupini	tip vezi
1	konferenca	udeleženci konf.				*elementarna*
2	udeleženec	udeleženec konf.				*elementarna*

Dodatno, zgolj za ilustracijo načina delovanja, podajamo tudi sliko razreda s preprosto rekurzivno povezavo (slika 5, tabela 3) kot eno izmed običajnih relacijskih struktur. Predstavlja zaposlene, pri čemer ima vsak zaposleni lahko samo enega šefa (ki je prav tako eden izmed zaposlenih), vsak šef pa poljubno število neposredno podrejenih.



Slika 5: **Rekurzivna povezava [ena : mnogo]**

Tabela 3: **Vezi k sliki 5**

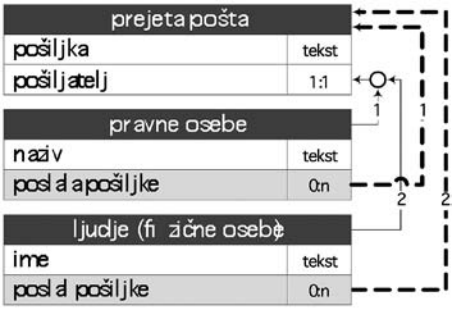
VEZI	polje sklica	polje seznama	nadrejena vez	skupina vezi	korak v skupini	tip vezi
1	šef	podrejeni				*elementarna*

3.2 Razvejane povezave

Pari polj v povezavah niso nujno ekskluzivni, kar je prvi izmed razlogov za potrebnost slovarske datoteke »vezi« (slika 1). Vsako posamezno polje namreč lahko nastopa v več povezavah. Posamezno polje tipa *sklic* je lahko v paru/-ih z enim ali več polji tipa *seznam* iz istega ali različnih razredov ter vsako polje tipa *seznam* je lahko v paru/-ih z enim ali več polji tipa *sklic*. Takšna polja imenujemo *razvejani sklic* in *razvejani seznam* (za tega je morda bolj ilustrativno poimenovanje *skupni seznam*).

Razvejanost povezav ima za posledico, da je treba pri vsakem sklicevanju oziroma poizvedbah poleg identifikacije zapisa v tabeli identificirati tudi tabelo samo – v Panorami je to urejeno prek slovarske tabele »panoramski razred« (slika 1). Ob tem je treba povedati, da so procedure, ki vodijo izvajanje operacij, povezanih z razvejanimi strukturami, precej kompleksnejše od procedur, ki jih zahtevajo standardne (nerazvejane) povezave.

Razvejani sklic je vrsta povezave, ki se uporabi, kadar je vsebina posameznega polja določenega razreda lahko predmet iz različnih razredov, kakršen je primer prejete pošte (slika 6, tabela 4). Pošto nam namreč lahko pošljejo tako pravne kot fizične osebe, vendar imajo pravne in fizične osebe precej različne atribute, zato jih je smiselno voditi v ločenih razredih.



Slika 6: ***Razvejani sklic* (polje sklica se nanaša na več razredov)**

Tabela 4: **Vezi k sliki 6**

VEZI	polje sklica	polje seznama	nadrejena vez	skupina vezi	korak v skupini	tip vezi
1	pošiljatelj	poslala pošiljke				*elementarna*
2	pošiljatelj	poslal pošiljke				*elementarna*

Razvejanost lahko nastopa tudi na strani *seznama*, in sicer kadar je posamezno polje tipa *seznam* v parih z več polji tipa *sklic* – polje tipa *skupni seznam*. *Skupni seznami* so malce bolj abstraktna oblika, katere namen in potrebnost postaneta jasnejša ob preiskovanju kompleksnejših podatkovnih struktur.

Redek primer na elementarnem podatkovnem nivoju so rodbinska drevesa, pri katerih nastopa rekurzivna povezava osebe z osebo kar dvakrat, in sicer s svojima materjo in očetom (slika 7, tabela 5). V tem primeru sta povezavi različni (mati, oče), *seznam* obeh pa je skupen (otroci).

Slika 7: **Razvejani ali skupni seznam (dva ločena sklica imata skupni pomen seznama)**

Tabela 5: **Vezi k sliki 7**

VEZI	polje sklica	polje seznama	nadrejena vez	skupina vezi	korak v skupini	tip vezi
1	oče	otroci				*elementarna*
2	mama	otroci				*elementarna*

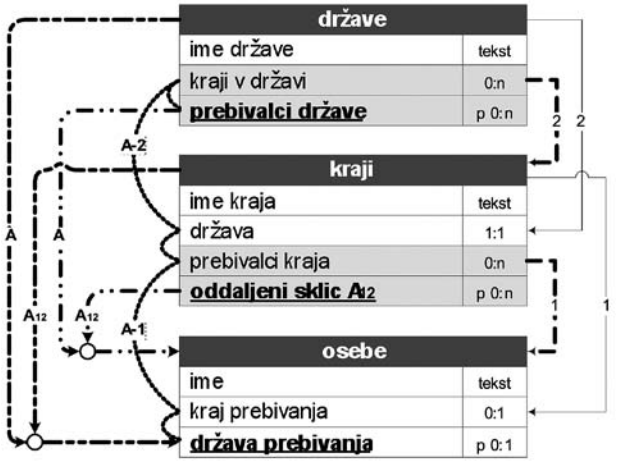
3.3 Oddaljene povezave

Oddaljene povezave so posredne oziroma sestavljene povezave med predmeti dveh oddaljenih razredov, ki obstojijo le, če v danem trenutku obstojijo vsi koraki povezave. Služijo za neposreden prikaz povezav, ki so sicer posredne, in so po obliki materializirani pogledi (*angl.* materialized view). Glede na kompleksnost procedur, ki so potrebne za njihovo vzpostavitev in vzdrževanje, ločimo dva tipa, in sicer *oddaljeni sklic* in *oddaljeni seznam*.

Za obe vrsti velja, da jih ne moremo ažurirati neposredno, saj so vrednosti vedno rezultat obstoja neprekinjene povezave po definirani poti skozi podatkovni model. Njihova vrednost je v preprostosti definicije in razumljivosti za uporabnika, bistvena posledica pa, da z dodajanjem oddaljenih povezav višamo informativnost tako povezanih razredov.

Prvi primer prikazuje vzpostavitev oddaljenega sklica med predmeti razredov »osebe« in »države« prek predmetov razreda »kraji« (slika 8, tabela 6). V tem primeru ima namreč vsaka oseba lahko določen največ en kraj prebivanja, vsak kraj pa ima določeno državo, v kateri se nahaja. Če torej za posamezno osebo obstaja vnos kraja prebivanja, lahko znotraj našega modela pridobimo tudi podatek o državi prebivanja, sicer pa je do tega podatka brez *oddaljenega sklica* mogoče priti le posredno. Za vzpostavitev neposrednega prikaza povezav se med polji razreda

»osebe« definira *oddaljeni sklic* na razred »države« in določi pot te posredne povezave. Kakor je razvidno iz tega primera, v razredu »osebe« pridobimo podatek o državi prebivanja, v razredu »države« pa *seznam* prebivalcev države.



Slika 8: **Oddaljena povezava tipa oddaljeni sklic**

Tabela 6: **Vezi k sliki 8**

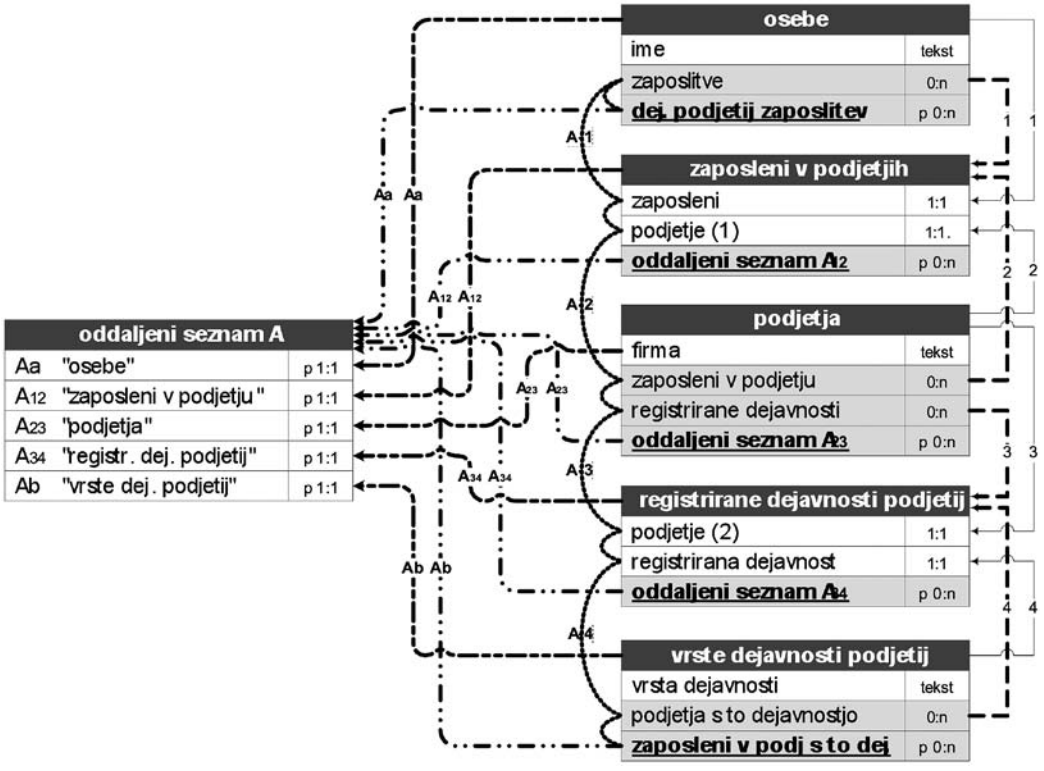
VEZI	polje sklica	polje seznama	nadrejena vez	skupina vezi	korak v skupini	tip vezi
1	kraj prebivanja	prebivalci kraja				*elementarna*
2	država	kraji v državi				*elementarna*
A	država prebivanja	prebivalci države		1	1	*oddaljena elementarna*
A-1	kraj prebivanja	prebivalci kraja	A	1	2	*korak*
A_{12}	država prebivanja	oddaljeni sklic A_{12}	A-1	1	0	*oddaljena vmesna*
A-2	država	kraji v državi	A-1	1	3	*korak*

Zahteva po strukturiranem zapisu poti je poleg že zgoraj omenjene razvejanosti drugi razlog za potrebnost datoteke »vezi« (slika 1), v katero zapišemo navodila poti oddaljene povezave. Ker je v prehodih med koraki jasno, da gre za prehod med polji znotraj razreda, ni potrebe po izrecnem zapisu teh korakov, saj je popolno in nedvoumno navodilo podano že implicitno. Vsekakor pa ima takšen prehod znotraj razreda za posledico avtomatsko vzpostavitev nove *vmesne oddaljene vezi*, ki služi kot vodilo pri ažuriranju in siceršnjem vzdrževanju pravilnosti vseh zapisov v bazi: tako namreč hitro in nedvoumno najdemo vse zapise, ki bi lahko bili potrebni posodobitve ob posodobitvi katerega koli od predmetov, vključenih v vezi oddaljenega sklica. Takšna vez je v gornjem primeru vez A12, ki si dva parametra (»nadrejena vez« in »skupina vezi«) sicer deli z vezjo A-2, kar jo vse nedvoumno umešča v ustrezno oddaljeno povezavo, vendar pa je njena vrednost parametra »korak v skupini« enaka 0, kar jo opredeli kot *vmesno oddaljeno povezavo*.

Druga oblika oddaljene povezave, *oddaljeni seznam*, sledi isti logiki, vendar pa je zaradi možnosti, da za vsak predmet oddaljeno povezanih dveh razredov obstaja več povezav s predmeti oddaljenega razreda (povezava [mnogo : mnogo]), potrebna vzpostavitev novega povezovalnega razreda, v katerega se zapisujejo vse obstoječe kombinacije povezav oddaljeno povezanih predmetov. Ti metapodatki omogočajo neposredni vpogled in nadaljnje analize zadevnih oddaljenih povezav.

V primeru oddaljenega seznama (slika 9, tabela 7) smo osebam v oddaljeni seznam pripisali združene sezname dejavnosti podjetij vseh njihovih zaposlitev in istočasno vrstam dejavnosti podjetij v oddaljeni seznam pripisali osebe, ki so bile zaposlene v podjetjih z ustreznimi registriranimi dejavnostmi. Z materializacijo »oddaljenega seznama A«, v katerem so posamično zavedene vse obstoječe poti med predmeti razredov »osebe« in »vrste dejavnosti podjetij«, lahko hitro in preprosto pridobimo nove podatke,

npr. navedbo zgolj različnih parov vrednosti začetnega in končnega polja, brez razlikovanja po vrednosti polj na poti med njima (namesto vse množice različnih peterk, ki so zapisane v razredu oddaljeni seznam A, na podlagi transformacije dobimo nov razred vrednosti z vsemi različnimi dvojicami).



Slika 9: **Oddaljena povezava tipa *oddaljeni seznam***

Tabela 7: **Vezi k sliki 9**

VEZI	polje sklica	polje seznama	nadrejena vez	skupina vezi	korak v skupini	tip vezi
1	zaposleni	zaposlitve				*elementarna*
2	podjetje (1)	zaposleni v podj.				*elementarna*
3	podjetje (2)	registr. dejavnosti				*elementarna*
4	registrirana dej.	podjetja sto dej..				*elementarna*
Aa	"osebe"	dej. podjetij zap.		1	1	*oddaljena temeljna*
A-1	zaposleni	zaposlitve	Aa	1	2	*korak*
A12	"zaposleni v p."	odd. seznam A12	A-1	1	0	*oddaljena vmesna*
A-2	podjetje (1)	zaposleni v podj.	A-1	1	3	*korak*
A23	"podjetja"	odd. seznam A23	A-2	1	0	*oddaljena vmesna*
A-3	podjetje (2)	registr. dejavnosti	A-2	1	4	*korak*
A34	"registr. dej.p."	odd. seznam A34	A-3	1	0	*oddaljena vmesna*
A-4	registrirana dej.	podjetja sto dej.	A-3	1	5	*korak*
Ab	"vrste dej. pod."	zaposl. v p. sto d.	A-4	1	6	*oddaljena temeljna*

V primerjavi z *oddaljenim sklicem* so potrebne procedure za vzdrževanje *oddaljenega seznama* bistveno bolj zahtevne, saj ob tem, da terjajo avtomatsko vzpostavitev in vzdrževanje novega razreda (tabele), tudi *temeljna oddaljena vez* med izbranima razredoma razpade na dva dela (v gornjem primeru na vezi Aa in Ab; začetni in končni korak *oddaljenega seznama*).

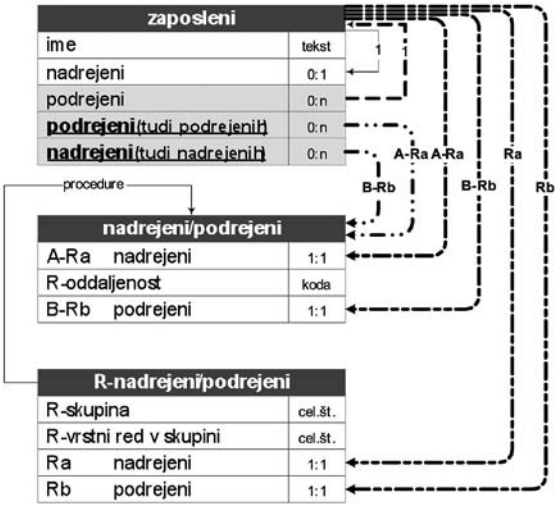
3.4 Obvladovanje rekurzivnih struktur v oddaljenih povezavah

Rekurzivne povezave pri preiskovanju kompleksnih struktur povzročajo težave. Vsak izmed predmetov rekurzivnih povezav ima namreč toliko posrednih povezav s predmeti istega razreda, kolikor je teh istovrstnih predmetov na eni strani njemu neposredno podrejenih in na drugi strani njemu neposredno nadrejenih, zmnoženo z istovrstnimi vrednostmi vsakega izmed teh. Težava, na katero naletimo, je, da se tovrstnih povezav ne da analizirati na podlagi definicije podatkovnega modela (slovarski nivo), ampak na tem nivoju lahko samo ugotovimo, da obstaja rekurzivnost. Da pridobimo podatek o dejanskem številu nadrejenih in podrejenih moramo namreč predhodno analizirati dejanske vrednosti podatkov.

Doslej smo se srečali z dvema primeroma rekurzivnih povezav (sliki 5 in 7), nismo pa se dotaknili posledic tovrstnih struktur na gradnjo oddaljenih povezav. Pri vseh oddaljenih strukturah, predstavljenih zgoraj (sliki 8 in 9), smo namreč z običajnimi strukturami, zapisanimi v slovarskih datotekah, pojasnili vse mogoče oddaljene povezave; rekurzivne strukture pa zaradi svoje narave, ki določene funkcije definicijske (slovarske) ravni privzame tudi na podatkovnem nivoju, v relacijskem modelu terjajo posebno obravnavo. Takšno preiskovanje rekurzivnih struktur poleg razumevanja povezav na slovarskem nivoju zahteva tudi analizo dejanskih povezav na podatkovnem nivoju. Za predstavitev bomo uporabili že uporabljeni preprosti model zaposlenih in njihovih šefov, in sicer z dodano oddaljeno povezavo posredno nadrejenih in podrejenih (slika 10).

Najprej ugotovimo, da gre v primeru podrejenih in nadrejenih za dve plati ene same povezave. V vsakem paru je namreč ena oseba nadrejeni in druga podrejeni. Vse to smo lahko ugotovili iz definicije, za zmožnost nadaljnjih podatkovnih operacij v teh strukturah pa potrebujemo še analizo dejanskih predmetov in njihove povezanosti na podatkovnem nivoju, saj šele tako lahko predvidljivo korakamo skozi podatkovni prostor brez skrbi, da se bomo začeli vrteti v krogu ali bomo nezmožni identificirati pot naprej.

Omejitve modeliranja v relacijskih strukturah pripeljejo do spoznanja, da podatkov o rekurzivnih povezavah ni mogoče obvladati v eni sami tabeli, saj je število korakov vnaprej povsem neznano – števila stolpcev (za zapis nivojev rekurzivnih pod- in nadrejenosti) ne moremo določiti vnaprej. Tako smo rešitev poiskali v sestavljenih relacijskih strukturah, ki neposredno sledijo vzpostavitvi rekurzivne strukture na slovarskem nivoju: z definicijo rekurzivne strukture se vzporedno s podatkovno tabelo vzpostavita še dve metapodatkovni tabeli, v kateri z vsakim vpisom podatka v podatkovno tabelo zapišemo analitične (meta)podatke o rekurzivnih razmerjih.



Slika 10: **Model obvladovanja razmerij v rekurzivnih strukturah**

Tabela R-nadrejeni/podrejeni služi zapisu celotne poti, in sicer z navedbo vsakega posameznega nadrejenega v vrsti, druga (v gornjem primeru nadrejeni/podrejeni) pa je analitična in navaja le pare prvega in zadnjega v vrstah (skupinah) iz prve tabele ter njuno oddaljenost (tj. koliko korakov skozi rekurzivno zanko je bilo opravljenih za povezavo dveh predmetov). Analitična tabela nam omogoča neposredno naslavljanje razmerij iz rekurzivnih struktur in s tem nadaljnje analitične postopke in interpretacije razmerij skozi razrede rekurzivnih povezav.

Ker sta obe tabeli vzpostavljeni in ažurirani povsem avtomatsko, so vezi med njima enosmerne. Povezave so namreč urejane in interpretirane programsko.

4 OD HIERARHIJE DO HETERARHIJ

Pri razvoju podatkovnih rešitev z uporabo platforme Panorama smo ugotovili, da so za dobro uporabniško izkušnjo pri potovanju skozi visoko strukturirane informacijske prostore oddaljene povezave bistvenega pomena. Na abstraktni ravni so to hierarhije, katerim pripadajo posamezni predmeti in so nosilke informacij, ki tem predmetom dodajajo (izvedene) lastnosti oziroma atribute.

Z objektno relacijskim pristopom in zgoraj omenjenimi nadgradnjami smo uspešno razširili vrste struktur, ki jih lahko oblikujemo za zapis dejanskih odnosov med predmeti posameznih vrst v podatkovni model. S tem smo združili podatkovni model in poizvedbe ter odprli prostor za svobodnejši, od dejanskih aplikacij in končnih informacijskih potreb neodvisen pristop k oblikovanju relacijskih baz podatkov. Pri gradnji podatkovnega modela namreč lahko brezkompromisno izhajamo iz dejanskih razmerij v realnosti, ki jo opisuje podatkovni sistem, potem pa znotraj tega verističnega modela oblikujemo izvedene strukture, ki se prilagajajo informacijskim potrebam posamičnih poslovnih procesov oziroma aplikacij.

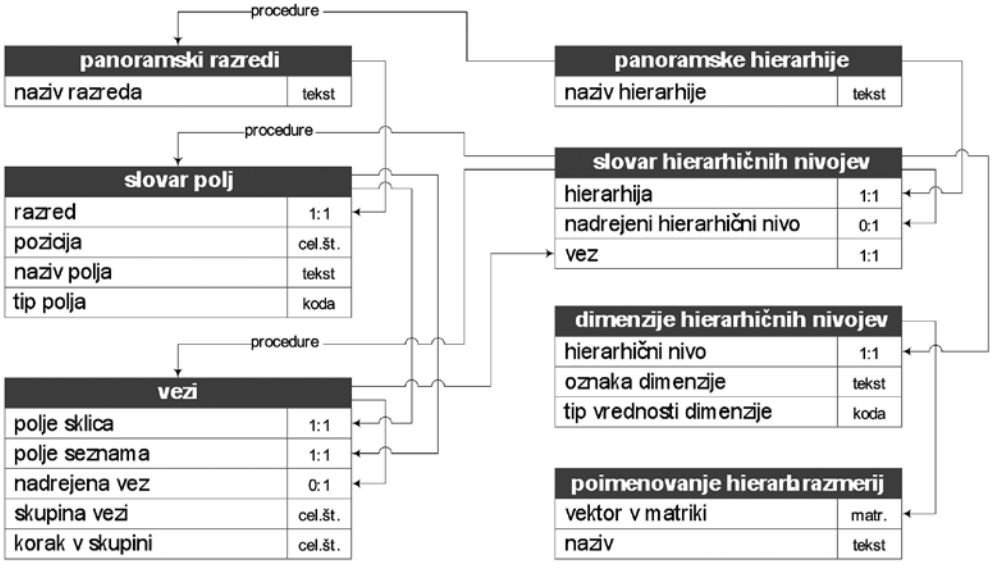
To smo dosegli tako, da smo omogočili definicijo hierarhij, katerim lahko pripada posamezen predmet, ki ima dejansko lastnost povezave prek določene vezi (v smislu vezi iz slovarske datoteke »vezi«; slika 2), s tem pa ta vez pridobiva lastne atribute ter z njimi povezane možnosti njihove interpretacije. Vsaki vezi moramo omogočiti uvrstitev v poljubno mnogo hierarhij oziroma v njihove poljubne kombinacije ali heterarhije.

Preliminarni testi so pokazali, da takšna strategija sicer zagotavlja možnost popolne in pravilne interpretacije vseh medsebojnih razmerij, da pa je lahko precej procesorsko požrešna, kar odpira povsem nova polja hevurističnih pristopov k optimizaciji. Pričakujemo namreč, da bomo morali podpreti dva načina pridobivanja podatkov o hierarhičnih pripadnostih:

- sprotnega, z izvedbo na ukaz ob zahtevi za vpogled v hierarhijo, ter
- preventivnega, ki neprenehoma zagotavlja popolno posodobljenost materializiranega (v)pogleda v namensko datoteko.

V prvem primeru moramo za sprejemljive odzivne čase zagotoviti v kratkem času izvedbo množice ukazov, v drugem pa stalno skrbeti za posodobljenost na eni in dodaten spominski prostor za (z vidika podatkovnega modela podvojene) analitične podatke na drugi strani. Tako poleg samega modela implementacije z dimenzijo hierarhij razširjenih slovarskih datotek razvijamo tudi ustrezne procedure.

Poenostavljena shema razširjenih slovarskih datotek je predstavljena na sliki 11.



Slika 11: **Panoramske tabele slovarskega nivoja z dodanimi vodili hierarhičnih struktur**

5 SKLEP

Panorama je uporabniku skoraj v celoti odprta podatkovna platforma za obvladovanje strukturiranih podatkov. Že danes omogoča oblikovanje visoko kompleksnih podatkovnih struktur ter ponuja precej funkcionalnosti razkrivanja informacij iz kompleksnih podatkovnih modelov.

S predlagano razširitvijo strukture slovarskih datotek in spremljajočimi procedurami bomo omogočili bistveno preglednejše in enostavnejše urejanje hierarhične pripadnosti. Tako bomo uporabnikom še bolj približali sproten vpogled v sestavljene informacije poljubne natančnosti.

Pred implementacijo se bomo posvetili še uporabniškemu vmesniku – preprosta dostopnost in razumljivost je pač edina učinkovita pot do sprejetosti pri uporabnikih. Abstraktnost, ki ga zahteva implementacija heterarhij v relacijskem podatkovnem modelu, k razumljivosti sama zase ne pripomore kaj dosti.

6 VIRI IN LITERATURA

[1] Date, C. J. (2012). Database Design and Relational Theory: Normal Forms and All That Jazz. Sebastopol, CA: O'Reilly Media, Inc.

[2] Lajovic, I. (2002). Urejanje in preiskovanje baz podatkov na asociativni osnovi. Uporabna informatika, 10 (3), 174–178.

[3] Lajovic, T. & Lajovic, I. (2012). Združevanje podatkovnega modela in poizvedb: izziv heterarhij. V Slovensko društvo informatika. (2012), *Ustvarimo nove rešitve! / 19. konferenca Dnevi slovenske informatike – DSI*, Portorož–Ljubljana: Slovensko društvo Informatika. Pridobljeno iz Zbornika predavanj DSI 2012 na CD-ROM-u.

[4] Lavbič, D., Lajovic, I. & Krisper, M. (2010). Facilitating information system development with panoramic view on data, ComSIS, 7, 737–767.

[5] Minsky, M. (1986). The Society of Mind. New York, NY: Simon and Schuster.

▪

Tomaž Lajovic je direktor družinskega podjetja KRES Kreativni sistemi. Pred tem je bil direktor energetske borze BSP SouthPool, potem ko je bil sedem let na mestih svetovalca direktorja energetske borze BSP SouthPool oziroma odgovornega pravnika pri slovenskem organizatorju trga z električno energijo Borzen odgovoren za pravne zadeve in razvoj trgovalnih sistemov v razmerah omejenih prenosnih zmogljivosti. V preteklosti je bil zaposlen kot pravni svetovalec v avdio/video inženiring podjetju TSE, kot specialist za pogodbe in pogajanja v družbi IBM Slovenija ter kot sekretar evropskega združenja borz električne energije EuroPEX.

▪

Iztok Lajovic je lastnik in dolgoletni direktor podjetja KRES Kreativni sistemi. Od samega začetka se ukvarja z razvojem programske opreme za finančno področje in objektno orientiranih podatkovnih baz. Pod njegovim vodstvom je bil že leta 1979 izdelan prvi celoviti in delujoči on-line sistem v gospodarstvu v Jugoslaviji. Zatem je deset let vodil razvoj programske opreme v Ljubljanski banki. Leta 1989 je ustanovil svoje podjetje za razvoj programske opreme, ki živi izključno od lastnega razvoja. Podjetje načeloma ne razvija programske opreme na podlagi naročil, ampak jo razvija na podlagi lastnih vizij za neznanega kupca. Ko se porodi ideja za nov produkt, ga izdelajo od začetne zasnove do končnega izdelka in ponudijo na trg, ki se odzove glede na svoje potrebe in možnostirešitev in informacijskih sistemov.

Ocena prednosti metode Scrum in njenih tipičnih praks

Janez Urevc, Viljan Mahnič
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Tržaška cesta 25, 1000 Ljubljana
janez@janezurevc.name; viljan.mahnic@fri.uni-lj.si

Izvleček

Čeprav je Scrum v praksi najbolj razširjena agilna metoda za razvoj programske opreme, je le malo empiričnih študij o njeni uporabi, prednostih in pomanjkljivostih. Večina poročil temelji na opisu izkušenj, ki niso podprte z empiričnimi podatki. Da bi empirično ovrednotili mnenja uporabnikov o koristih, ki jih prinaša Scrum, in poiskali tiste njegove prakse, ki največ prispevajo k uspehu projektov, so avtorji izvedli anketo med 75 uporabniki te metode v Sloveniji in tujini. Rezultati ankete so potrdili, da se uporabniki v celoti strinjajo s trditvami o prednostih Scruma, ki jih navaja literatura. Med dejavniki, ki najbolj vplivajo na uspeh po Scrumu vodenih projektov, so posebno izpostavili medsebojno komunikacijo znotraj razvojne skupine in komunikacijo med razvijalci in produktnim vodjo. Rezultati lahko služijo podjetjem, ki nameravajo uvesti Scrum v svoj razvojni proces, pri analizi pričakovanih koristi in identifikaciji najpomembnejših praks, ki jih morajo upoštevati, da bi pri tem uspela.
Ključne besede: Scrum, agilne metode, razvoj programske opreme, vodenje projektov.

Abstract

Evaluation of Scrum Benefits and its Typical Practices

In spite of the fact that Scrum is the most widespread agile software development method in industry, empirical evidence about its use, advantages and disadvantages remains scarce. Most reports are based on anecdotal evidence that is not supported by empirical data. In order to empirically evaluate users' opinions of benefits of Scrum and identify those practices that contribute most to the success of a Scrum project, a survey was conducted among 75 Scrum users in Slovenia and abroad. The survey has confirmed that the users fully agree with all advantages of Scrum reported in the literature. Among factors with the greatest impact on the success of Scrum projects, they highlighted communication within the development team and communication between developers and the Product Owner. Results of the survey can serve companies that are planning to introduce Scrum into their development process in analyzing expected benefits of Scrum implementation and identifying the most important practices that need to be considered in order to succeed.
Keywords: Scrum, agile methods, software development, project management.

1 UVOD

Agilne metode za razvoj programske opreme (Williams, 2010) so se začele pojavljati v devetdesetih letih prejšnjega stoletja kot reakcija na tradicionalni disciplinirani pristop, ki zahteva natančno vnaprejšnje načrtovanje in predpisuje točno določen postopek razvoja. V nasprotju s tradicionalnimi pristopi daje *Manifest agilnega razvoja programske opreme*[1] prednost posameznikom in interakciji med njimi pred procesi in orodji, delujoči programski opremi pred obsežno dokumentacijo, sodelovanju z naročnikom pred pogodbenimi pogajanji in odzivanju na spremembe pred togim sledenjem načrtu. S tem zmanjšuje pomen nekaterih vrednot, ki so bolj značilne za tradicionalne metode.

Kljub temu da med razvijalci programske opreme in informacijskih sistemov ni enotnega mnenja o primernosti agilnih metod, vse več podatkov kaže na njihovo uspešnost in naraščajočo popularnost. Tako Ambler (2008) navaja, da uporaba agilnih metod bistveno poveča produktivnost, kakovost in zadovoljstvo vseh udeležencev v razvojnem procesu. Forrester v svojem poročilu (West & Grant, 2010) ugotavlja, da je leta 2009 uporabljalo agilne metode že 35 odstotkov projektov, po Gartnerjevi napovedi (Murphy idr., 2009) pa naj bi leta 2012 kar 80 odstotkov projektov potekalo na agilen način. Čeprav prevladuje mnenje, da so agilne metode primerne predvsem za manjše projekte (Boehm & Turner, 2004), lahko v literaturi zasledimo poročila o uvajanju teh metod tudi v največjih podjetjih s področja informacijske tehnologije, kot so npr. Microsoft (Begel & Nagappan, 2007), Yahoo (Scotland & Boutin, 2008), Nokia (Laanti idr., 2011) ipd.

Scrum (Schwaber, 2004) je med vsemi agilnimi metodami najbolj razširjen, saj njegov delež po zadnjih podatkih za leto 2011 znaša 52 odstotkov, v kombinaciji z ekstremnim programiranjem pa 66 odstotkov (Version One, 2011). Tudi v Sloveniji se krog uporabnikov te metode vedno bolj širi; skupina Skram.si v omrežju LinkedIn šteje preko tristo članov. Vseeno pa je le malo empiričnih raziskav, ki bi s kvantitativnimi podatki ovrednotile njegove prednosti in tipične prakse. Pregled empiričnih študij o uporabi agilnih metod, ki sta ga izdelala Dybå & Dingsøyr (2008), navaja eno samo študijo, ki obravnava metodo Scrum. Več je bilo narejenega na področju izobraževanja, v okviru katerega tudi v Sloveniji že več let poučujemo to metodo v obliki praktičnega dela na študentskih projektih (Mahnič, 2010; Mahnič, 2012), na podlagi izkušenj pri delu s študenti pa so bila predstavljena tudi priporočila za njeno uvajanje v prakso (Mahnič, 2011).

Da bi natančneje ugotovili, kako uporabniki metode Scrum ocenjujejo njene prednosti in kakšen pomen pripisujejo posameznim tipičnim praksam, ki jih predpisuje Scrum, smo med uporabniki te metode konec leta 2011 izvedli anketo, ki sta jo (poleg uvodnega dela) sestavljala dva vsebinska sklopa vprašanj. Uvodni del je služil za segmentacijo anketirancev in je obsegal vprašanja o njihovi vlogi v projektih, ki so vodeni po metodi Scrum (Scrum Master/Product Owner/Team Member), in izkušnjah (koliko časa že uporabljajo Scrum in na koliko projektih Scrum so že sodelovali). Namen prvega vsebinskega sklopa vprašanj je bil preveriti, v kolikšni meri se anketiranci strinjajo s trditvami o prednostih metode Scrum, ki jih zasledimo v literaturi. V okviru drugega sklopa pa smo želeli ugotoviti, katere tipične prakse, ki jih predpisuje Scrum, po njihovem mnenju največ prispevajo k uspešnosti projektov.

Ideja za izvedbo ankete se je porodila v okviru projekta uvajanja metode Scrum v delo oddelka za spletni razvoj pri časopisnem podjetju Delo (Urevc idr., 2012), za njeno izpolnjevanje pa smo poleg sodelavcev tega oddelka zaprosili še člane slovenske skupine Skram.si, člane skupnosti razvijalcev sistema za upravljanje z vsebinami Drupal[2] (ker razvoj spletnega portala časopisa Slovenske novice temelji na uporabi tega sistema) in člane skupine Scrum Practitioners, ki tudi deluje v omrežju LinkedIn.

Dobili smo 75 odgovorov, od tega 47 (62,7 %) iz Slovenije in 28 (37,3 %) iz tujine. Med tistimi, ki so odgovorili, je bilo 33 (44,0 %) skrbnikov metodologije (angl. *Scrum Master*), 29 (38,7 %) razvijalcev članov razvojne skupine (angl. *Team Member*) in 13 (17,3 %) produktnih vodij (angl. *Product Owner*).

Glede na to, da je vsaj v Sloveniji metoda Scrum razmeroma nova, je večina anketirancev (35 ali 46,7 %) imela manj kot šest mesecev izkušenj z njeno uporabo, vendar je bil tudi delež takih, ki metodo uporabljajo več kot 24 mesecev, razmeroma visok (20 anketirancev ali 26,7 %). Od 7 do 12 mesecev izkušenj je imelo 7 anketirancev (9,3 %), od 13 do 24 mesecev pa 11 (14,7 %). Dva anketiranca (2,7 %) še nista imela praktičnih izkušenj, ampak sta metodo poznala samo iz literature.

Večina anketirancev (48 ali 64,0 %) je doslej delala na enem ali dveh projektih, ki so potekali po metodi Scrum. Takih, ki so sodelovali pri 3–5 projektih je bilo 12 (16,0 %), izkušnje z več kot petimi projekti pa je imelo 13 (17,3 %) anketirancev.

Namen tega prispevka je skozi rezultate ankete izpostaviti glavne prednosti metode Scrum in izluščiti tiste tipične prakse, ki po mnenju njenih uporabnikov najbolj vplivajo na uspešnost projektov, vodenih po tej metodi. Pri tem primerjamo mnenja uporabnikov iz Slovenije z mnenji uporabnikov iz tujine ter analiziramo, kako so mnenja odvisna od njihovih izkušenj in vloge, ki jo imajo v metodi. S tem želimo pomagati tistim podjetjem, ki se odločajo za uvedbo metode Scrum, da bi bolje spoznala njene prednosti in se osredinila na tiste dejavnike, ki so po izkušnjah uporabnikov najpomembnejši. Upamo, da bo to pripomoglo k večji uspešnosti projektov in povečanju zadovoljstva ob uvajanju agilnih metod. Obenem pričakujemo, da bodo podatki koristni tudi za podjetja, ki že delujejo na podlagi Scruma, saj jim bodo omogočili optimizacijo procesov na podlagi izkušenj drugih.

Da bi bralec bolje razumel pomen posameznih anketnih vprašanj, bomo v nadaljevanju najprej na kratko predstavili metodo Scrum. Nato bomo podrobneje opisali odgovore na oba vsebinska sklopa vprašanj, v sklepu pa povzeli najpomembnejše rezultate. Ker uporablja Scrum specifično terminologijo, ki jo je v nekaterih primerih nemogoče dobesedno

[1] http://agilemanifesto.org (angleška verzija); http://agilemanifesto.org/iso/sl (slovenska verzija).
[2] http://drupal.org.

prevesti v slovenščino, navajamo v opisu metode poleg (po našem mnenju najprimernejšega) slovenskega prevoda tudi izvirni izraz v angleščini.

2 METODA SCRUM

Scrum izhaja iz premise, da je razvoj programske opreme preveč zapleten in nepredvidljiv proces, da bi ga lahko natančno načrtovali vnaprej. Namesto tega je treba vzpostaviti empiričen nadzor, ki omogoča sproten vpogled in prilagajanje trenutnemu stanju. To lahko dosežemo z iterativnim in inkrementalnim pristopom.

Vloge v metodi Scrum

Razvoj programske opreme po metodi Scrum predvideva tri vloge: produktni vodja (angl. *Product Owner*), razvojna skupina (angl. *Scrum Team*) in skrbnik metodologije (angl. *Scrum Master*).

Produktni vodja je predstavnik naročnika in zastopa vse, ki so zainteresirani za rezultate projekta. Njegova glavna naloga je upravljanje seznama zahtev (angl. *Product Backlog*), določanje njihove prioritete in grupiranje zahtev v posamezne izdaje (angl. *Release*). Seznam zahtev ni nikoli dokončen, ampak se lahko dopolnjuje ves čas trajanja projekta. Vsaka zahteva je praviloma opisana v obliki uporabniške zgodbe (angl. *User Story*), ki vsebuje besedilo zgodbe in seznam sprejemnih testov (Cohn, 2004). Ker gre za zelo ohlapen opis zahteve, mora biti produktni vodja stalno na voljo razvijalcem za pojasnila glede podrobnosti, povezanih z realizacijo. Po vsaki iteraciji pa mora preveriti, ali je zgodba realizirana ustrezno ali ne. Ta vloga je ključnega pomena za uspešnost projekta, zato mora imeti produktni vodja jasno vizijo o tem, kaj je treba narediti, in skladno s tem usmerjati razvojni proces.

Razvojna skupina je zadolžena za implementacijo zahtevane funkcionalnosti. Sestavljena mora biti tako, da pokriva vsa področja, ki so pomembna za izvedbo projekta. Skupina deluje po načelu samoorganizacije in sama določa, kako na najboljši način realizirati posamezne zahteve. Člani razvojne skupine so kolektivno odgovorni za uspeh posameznih iteracij in projekta kot celote.

Skrbnik metodologije ima na videz enako vlogo, kot jo ima pri tradicionalnem pristopu vodja projekta, vendar so njegove zadolžitve v resnici drugačne. Njegova naloga je skrbeti za nemoten potek dela, tako da vsi, ki delajo na projektu, upoštevajo pravila metode Scrum in ustrezno izvajajo predpisane aktivnosti. Pri tem mora paziti, da se Scrum vklaplja v kulturo organizacije tako, da daje najboljše rezultate. Skrbnik metodologije v veliki meri deluje kot varuh, ki varuje razvojno skupino pred škodljivimi zunanjimi vplivi, odpravlja morebitne ovire in s tem zagotavlja optimalne pogoje za delo.

Potek razvojnega procesa po metodi Scrum

Na začetku projekta produktni vodja izdela seznam zahtev, ki vsebuje vse do takrat znane uporabniške zgodbe. Zgodbe razvrsti po prioriteti in jih razdeli v predvidene izdaje. Razvojna skupina oceni zahtevnost vsake zgodbe z ustreznim številom točk (angl. *Story Points*) in določi predvideno hitrost razvoja (angl. *Velocity*), tj. število točk, ki jih lahko realizira v eni iteraciji. V skladu s tem nato izdela načrt izdaje (angl. *Release Plan*), tako da v skladu s prioriteto razporedi zgodbe po posameznih iteracijah. Seštevek točk vseh zgodb v neki iteraciji ne sme preseči predvidene hitrosti razvoja.

Nadaljnji razvoj poteka v iteracijah (angl. *Sprint*), ki v originalni verziji metode Scrum trajajo 30 dni, danes pa največ uporabljajo iteracije, ki trajajo dva ali štiri tedne. Vsaka iteracija se začne s sestankom za načrtovanje iteracije (angl. *Sprint Planning Meeting*), na katerem se produktni vodja in razvojna skupina dogovorita, katere zgodbe bodo realizirane v naslednji iteraciji.

Na podlagi tega razvojna skupina izdela seznam nalog (angl. *Sprint Backlog*), ki vsebuje vse naloge, potrebne za realizacijo dogovorjene funkcionalnosti do konca iteracije. Med iteracijo ta seznam še dopolnjujejo, obsežnejše naloge pa razdelijo na več manjših, tako da vsaka zahteva 4 do 16 ur dela.

Člani razvojne skupine se vsak dan zberejo na petnajstminutnem sestanku (angl. *Daily Scrum Meeting*), na katerem vsak izmed njih odgovori na tri vprašanja: »Kaj si naredil od prejšnjega sestanka? Kaj boš delal do naslednjega sestanka? Ali imaš pri delu kakšne težave?« Namen tega sestanka je sinhronizirati delo vseh članov razvojne skupine in sproti identificirati morebitne probleme.

Na koncu vsake iteracije je predpisan sestanek za pregled rezultatov (angl. *Sprint Review Meeting*), na katerem razvojna skupina predstavi rezultate svojega dela produktnemu vodji in vsem zainteresiranim uporabnikom. Metoda striktno zahteva, da razvijalci spoštujejo koncept »*Done*«, kar pomeni, da mora biti

vsaka zgodba realizirana, testirana in dokumentirana v celoti, tako da jo je moč neposredno predati v produkcijo. Produktni vodja sme sprejeti samo tiste zgodbe, ki v celoti ustrezajo omenjenemu konceptu, in samo seštevek točk teh zgodb se lahko upošteva kot dejanska hitrost razvoja (angl. *Actual Velocity*) razvojne skupine.

Po tem sestanku (in pred začetkom naslednje iteracije) skrbnik metodologije organizira še sestanek za oceno kakovosti razvojnega procesa (angl. *Sprint Retrospective Meeting*), katerega namen je poiskati možnosti za izboljšave razvojnega procesa, tako da bi bil ta še bolj učinkovit v naslednjih iteracijah.

3 OCENA PREDNOSTI METODE SCRUM

V prvem sklopu anketnih vprašanj smo anketirance spraševali, v kolikšni meri po njihovem mnenju držijo trditve o metodi Scrum, objavljene v literaturi. Za izhodišče smo vzeli prednosti, ki jih navajata Rising & Janoff (2000). Anketiranci so veljavnost posameznih trditev ocenjevali s pomočjo petstopenjske Likertove lestvice, pri čemer je ocena 1 (najslabša ocena) pomenila, da je trditev v celoti napačna, ocena 5 (najboljša ocena) pa, da trditev v celoti drži (tj. da se anketiranec z njo v celoti strinja). Spraševali smo o spodnjih trditvah.

1. Izdelek (projekt) lahko razdelimo na zaporedje obvladljivih delov.

Ta trditev izhaja iz iterativne in inkrementalne narave metode Scrum, ki zagotavlja, da smo v določenem trenutku osredinjeni na zahteve trenutne iteracije, ki (zaradi doslednega upoštevanja prioritete posameznih zgodb) prinašajo naročniku največjo korist. To zahteva od produktnega vodje (razvijalci pa mu morajo pomagati pri tem), da načrtuje razvoj v obliki večjega števila izdaj, ki jih je moč postopoma in v čim krajših časovnih intervalih predajati v produkcijo. Ključno vprašanje je, ali je takšna razdelitev mogoča pri vseh projektih, saj se lahko zgodi, da so vse komponente nekega izdelka preveč prepletene med seboj.

2. Projekt napreduje tudi, ko zahteve niso stabilne.

Seznam zahtev lahko dopolnjujemo z novimi uporabniškimi zgodbami ves čas, dokler projekt ni končan. Pri tem je pomembno, da produktni vodja vzdržuje prioriteto posameznih zgodb in s tem zagotavlja, da so zgodbe vedno razvrščene glede na poslovno vrednost, ki jo prinašajo naročniku. To poenostavi načrtovanje iteracij (vsakokrat izberemo za realizacijo tiste zgodbe, ki so trenutno na vrhu seznama), obenem pa zagotavlja, da naročnik vedno dobiva rešitev, ki je zanj v tistem trenutku najbolj koristna. Pri tem je treba poudariti, da se potem, ko je vsebina iteracije že dogovorjena, produktni vodja ne sme vmešavati v delo razvojne skupine. Med samo iteracijo mora imeti razvojna skupina mir, da se lahko v celoti posveti realizaciji dogovorjene funkcionalnosti. Na začetku naslednje iteracije spet sledi dogovarjanje o realizaciji najpomembnejših uporabniških zgodb iz seznama zahtev, ki se je medtem lahko spremenil.

3. Vsi udeleženci imajo popoln vpogled v potek dela.

To je zagotovljeno z rednimi vsakodnevnimi sestanki, na katerih se člani razvojne skupine medsebojno informirajo o poteku dela in morebitnih težavah, na katere naletijo. Uporabniki pa imajo možnost, da na koncu vsake iteracije pregledajo realizirano funkcionalnost in podajo svoje pripombe.

4. Izboljša se komunikacija med člani razvojne skupine.

Ta trditev se neposredno navezuje na prejšnjo. Naj poudarimo, da komunikacijo izboljšuje tudi t. i. pristop »*face-to-face*«. Glede na to, da vse podrobnosti projekta produktni vodja in člani razvojne skupine razčiščujejo sproti, se komunikacija vsekakor poveča v primerjavi z metodami, pri katerih razvijalci delajo na podlagi pisno podanih zahtev.

5. Člani razvojne skupine so zaslužni za uspeh v času razvoja in ob koncu projekta.

Metoda omogoča članom razvojne skupine, da sami ocenjujejo zahtevnost posameznih zgodb in predvideno hitrost razvoja, zato praviloma niso (ali vsaj ne bi smeli biti) izpostavljeni pritiskom zaradi nerazumno postavljenih rokov in nesprejemljivih zahtev produktnega vodje. Pred tem jih ščiti tudi skrbnik metodologije. Obenem delujejo po načelu samoorganizacije in sami izbirajo način, kako bodo realizirali posamezne zahteve. Zato so pri svojem delu bolj sproščeni. Vse to prispeva k večji odgovornosti in zavzetosti razvojne skupine, da tisto, kar je obljubila na začetku vsake iteracije, tudi uresniči. S tem pa se poveča tudi zavedanje o kolektivnih zaslugah za uspešen potek projekta.

6. Naročnik dobiva novo funkcionalnost v dogovorjenih časovnih intervalih.

To je zagotovljeno z iterativnim in inkrementalnim postopkom razvoja. Vsaka iteracija se konča s sestankom, na katerem razvojna skupina predstavi rezultate svojega dela. Rezultat vsake iteracije mora biti nova funkcionalnost, ki je neposredno uporabna. Dolžina iteracij je dogovorjena vnaprej, tako da naročnik točno ve, v kakšnih intervalih bo dobival predvidene rezultate. Na podlagi sprotnega vzdrževanja načrta izdaje pa lahko vedno oceni, kdaj bo izdaja končana v celoti oziroma kolikšen obseg funkcionalnosti bo lahko reliziran do določenega roka.

7. Naročnik (uporabniki) lahko sproti preverja(jo), kako izdelana programska oprema deluje v resnici.

Koncept »*Done*« zahteva, da je rezultat vsake iteracije delujoča programska koda, ki je v celoti preverjena in integrirana v trenutno rešitev. Zato lahko naročnik po vsaki iteraciji preveri, kako bo načrtovani sistem deloval v resnici. Pogosto se dogaja, da so uporabniki šele takrat, ko v živo preizkusijo neko rešitev, sposobni pravilno formulirati svoje zahteve. Sprotno preverjanje izdelane programske opreme tako zagotavlja, da uporabnik dobi tako rešitev, kot jo v resnici potrebuje. Po drugi strani pa odpravlja obsežne integracijske teste na koncu projekta.

8. Metoda prispeva k izgradnji boljših odnosov med naročnikom (uporabniki) in razvijalci: poveča se medsebojno zaupanje in presek skupnega znanja.

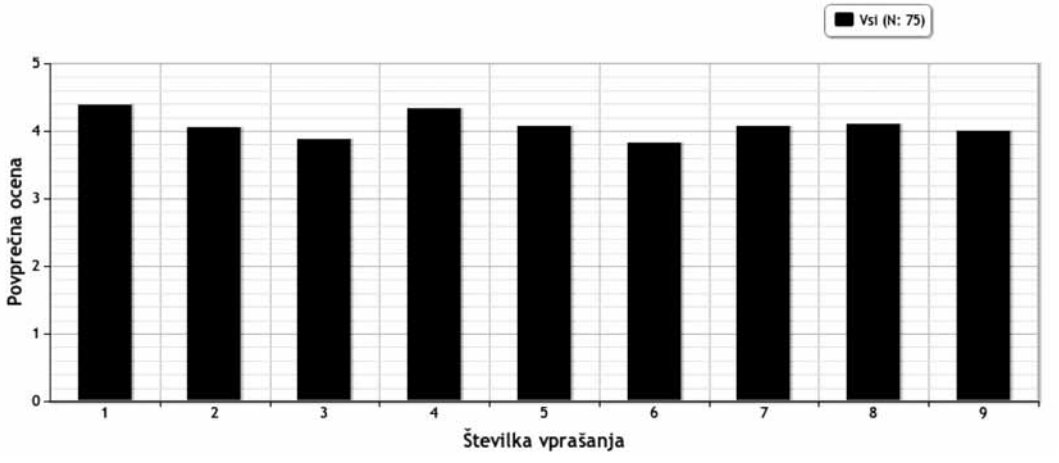
Metoda predvideva sodelovanje predstavnikov naročnika (produktnega vodje) ves čas trajanja projekta, ne samo na začetku in na koncu, kot je praviloma pri tradicionalnem pristopu. To vpliva na boljše medsebojno (s)poznavanje vseh sodelujočih, kar se odraža v izboljšanju odnosov in povečanju zaupanja. Razumeti je treba, da v projektih razvoja programske opreme pogosto sodelujejo tudi strokovnjaki z netehničnih področij. Pogosta težava v tako heterogenih skupinah se izkaže prav v velikih interesnih razlikah in majhnem preseku skupnega znanja. Dobra in sprotna komunikacija odpravlja te težave in povečuje presek skupnega znanja.

9. Metoda prispeva k izgradnji pozitivnega vzdušja, v katerem vsi pričakujejo uspeh projekta.

Ta točka je zelo povezana s predhodno. V okolju, v katerem vladajo pozitivni odnosi in medsebojno zaupanje, bo vladalo tudi pozitivno vzdušje, to pa se bo seveda odražalo tudi v pozitivnih pričakovanjih.

Rezultati

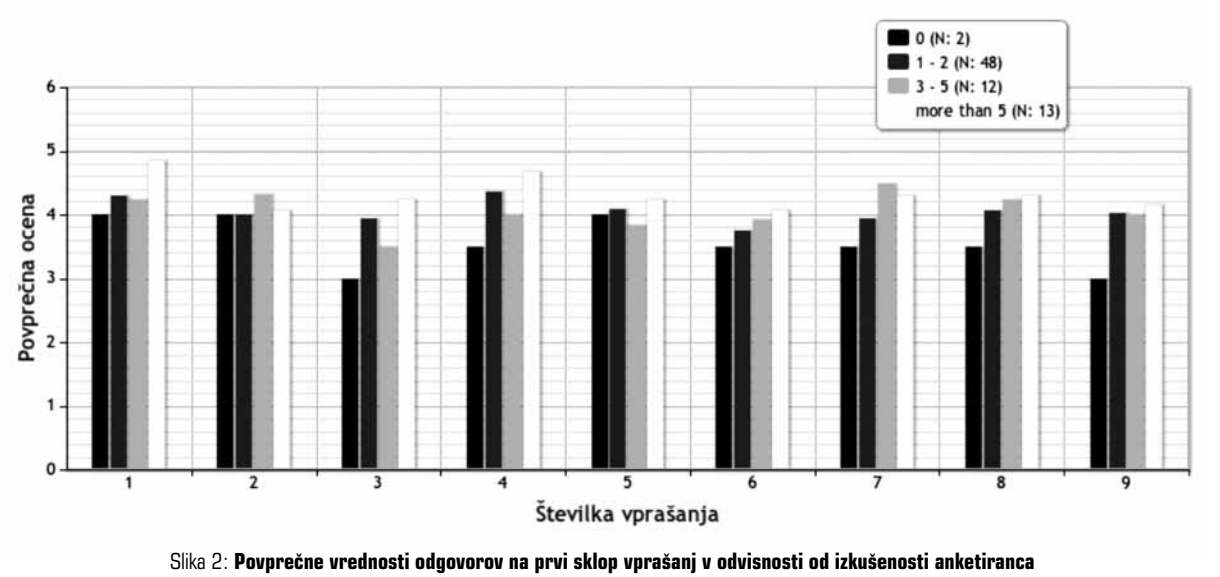
Povprečne ocene, izračunane na podlagi odgovorov vseh 75 anketirancev, so prikazane na sliki 1. Iz slike



Slika 1: **Povprečne vrednosti odgovorov na prvi sklop vprašanj**

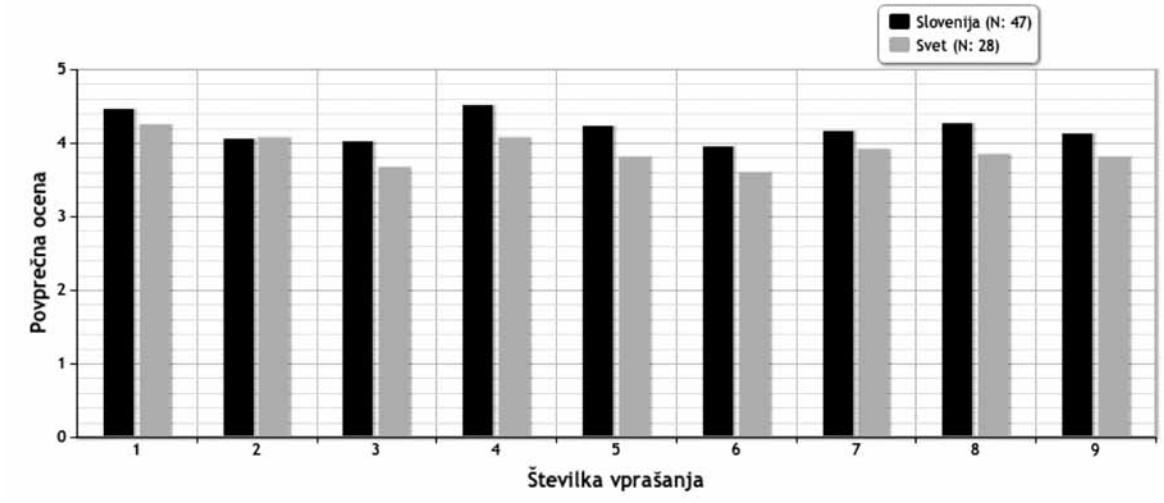
je razvidno, da se uporabniki metode Scrum strinjajo z vsemi trditvami, saj se vse ocene nahajajo na intervalu med 3,8 in 4,4. Tudi standardni odklon, ki smo ga izračunali, je sorazmerno nizek in je za vse trditve manjši od 1,0.

Anketiranci so se najbolj strinjali s trditvama št. 1 (projekt lahko razdelimo na zaporedje obvladljivih delov) in št. 4 (izboljša se komunikacija med člani razvojne skupine). Po drugi strani sta se za najmanj sprejeti trditvi izkazali št. 3 (vsi udeleženci imajo popoln vpogled v potek dela) in št. 6 (naročnik dobiva novo funkcionalnost v dogovorjenih časovnih intervalih).



Slika 2: **Povprečne vrednosti odgovorov na prvi sklop vprašanj v odvisnosti od izkušenosti anketiranca**

Odgovore smo dodatno analizirali glede na izkušnje anketirancev (slika 2), tako da smo anketirance razdelili na štiri skupine, upoštevajoč število projektov, vodenih po metodi Scrum, pri katerih so že sodelovali (0 projektov, 1–2 projekta, 3–5 projektov in več kot 5 projektov). Pokazalo se je, da se s trditvami opazno bolj strinjajo tisti, ki imajo s Scrumom več izkušenj. Še posebno pa je zanimivo, da se pri večini trditev povprečne ocene povečujejo sorazmerno s številom projektov, pri katerih je sodeloval anketiranec. Na podlagi te ugotovitve bi lahko trdili, da z več izkušnjami raste tudi zaupanje v metodo Scrum. Videti je, da so novi uporabniki najprej nekoliko bolj previdni, a se z izkušnjami začnejo bolj zavedati njenih prednosti.



Slika 3: **Povprečne vrednosti odgovorov na prvi sklop vprašanj v odvisnosti od lokacije anketiranca**

Zanimiva je tudi primerjava odgovorov slovenskih in tujih strokovnjakov, ki jo prikazuje slika 3. Prav z vsemi trditvami se slovenski uporabniki metode Scrum strinjajo nekoliko bolj kot njihovi kolegi iz tujine.

4 FAKTORJI, KI VPLIVAJO NA USPEH PROJEKTA

V drugem sklopu vprašanj nas je zanimalo, katere tipične prakse, ki jih predpisuje metoda Scrum, po njihovem mnenju najbolj vplivajo na uspeh projekta. Anketiranci so z ocenami od 1 do 5 ocenjevali spodnje dejavnike.

1. Jasno zastavljen seznam vseh zahtev
Seznam zahtev služi kot podlaga za ocenjevanje zahtevnosti posameznih zgodb, izdelavo načrta izdaje in načrtovanje posameznih iteracij. Sprejemni testi, ki jih mora vsebovati vsaka uporabniška zgodba, omogočajo preverjanje, ali je zgodba realizirana v skladu s potrebami uporabnikov.

2. Sprotna komunikacija s produktnim vodjo
Ker so uporabniške zgodbe samo ohlapen opis vsake zahteve, je za razjasnitev vseh podrobnosti potrebna sprotna komunikacija s produktnim vodjo. Če je ta komunikacija slaba ali ne poteka sproti in se nejasnosti ne razrešijo, lahko pride do situacije, ko razvojna skupina ne more nadaljevati z delom ali pa realizira nekaj, kar ne zadovolji pričakovanj naročnika.

3. Dober skrbnik metodologije
Skrbnik metodologije vsakodnevno bdi nad uporabo metodologije in potekom projekta. Zagotavljati mora take pogoje za delo, da je razvojna skupina čim bolj produktivna. Dober skrbnik metodologije bo zelo zgodaj identificiral odstopanja od začrtanega načrta in nastajajoče težave. Z usklajenim delovanjem vseh sodelujočih bo zagotovil, da bodo takšne okoliščine razrešene čim hitreje.

4. Dobra komunikacija med člani razvojne skupine
Dobra komunikacija med člani razvojne skupine se odraža tudi v dobrih in pozitivnih odnosih, pri čemer vsi člani delujejo kooperativno. Omogoča tudi sproten, celovit in transparenten vpogled v potek dela na projektu ter medsebojno pomoč v primeru težav.

5. Pravilne ocene zahtevnosti posameznih zgodb.
Pravilne ocene zahtevnosti so prvi pogoj za izdelavo zanesljivega načrta izdaje in pravilno načrtovanje posameznih iteracij. Na podlagi teh ocen in predvidene hitrosti razvoja razvrščamo zgodbe v posamezne iteracije. Če ocene niso vsaj približno pravilne, se nam lahko zgodi, da v neko iteracijo uvrstimo preveč ali premalo zgodb.

6. Začetna razporeditev zgodb po iteracijah – načrt izdaje
Načrt izdaje omogoča določitev rokov za izvedbo in časovnih okvirov, v katerih bomo sposobni posamezne dele projekta predati naročniku. Načrt izdaje daje celovit vpogled v časovne okvire projekta kot celote.

7. Pravilna ocena hitrosti razvoja na začetku vsake iteracije
Hitrost razvoja nam pove, kakšno količino dela je razvojna skupina sposobna opraviti v eni iteraciji. Na začetku jo ocenimo glede na različna merila (velikost razvojne skupine, izkušenost itn.), kasneje pa jo prilagajamo glede na dejansko doseženo število točk v predhodnih iteracijah. Ta faktor je zelo povezan s tistim, ki smo ga omenili pod točko 5. Samo kombinacija dobrih ocen zahtevnosti uporabniških zgodb in dobre ocene hitrosti razvoja bo omogočila korektno načrtovanje posameznih iteracij.

8. Dosledno izvajanje sestankov za načrtovanje iteracije
Rezultat tega sestanka je seznam vseh uporabniških zgodb, ki jih moramo realizirati v iteraciji, in dekompozicija teh zgodb na posamezne naloge. Ker ta predstavlja načrt dela posamezne iteracije, bo neustrezno izvajanje tega sestanka lahko pripeljalo do slabo vodene in/ali izpeljane iteracije.

9. Dosledno izvajanje vsakodnevnih sestankov
Vsakodnevni sestanki omogočajo sproten vpogled v stanje projekta. Omogočajo, da morebitne težave odkrijemo takoj, ko se pojavijo, in jih odpravimo sproti. Paziti pa je treba, da ne postanejo predolgi, saj s tem zmanjšamo čas, ki ga lahko izkoristimo za razvoj. Ravno zato so vsakodnevni sestanki časovno omejeni.

10. Dosledno upoštevanje koncepta »Done«
Scrum zahteva, da je na koncu iteracije vsaka funkcionalnost realizirana tako, da jo je moč neposredno predati v uporabo. S tem odpade obsežno testiranje na koncu projekta, ampak imamo vedno na razpolago delujočo verzijo izdelka. Dosledno upoštevanje koncepta »*Done*« omogoča, da naročnik sproti preizkuša novo funkcionalnost in na podlagi tega precizneje oblikuje svoje zahteve.

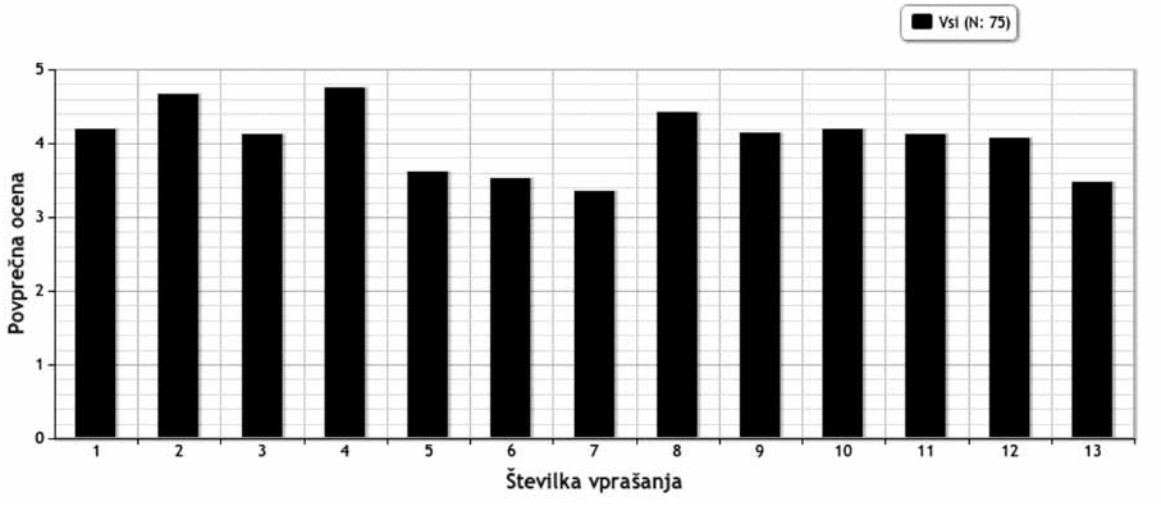
11. Dosledno izvajanje sestankov za predstavitev rezultatov iteracije
V okviru teh sestankov produktnemu vodji in zainteresiranim uporabnikom predstavimo rezultate vsake iteracije. Služijo tako za pregled in potrjevanje realizirane funkcionalnosti kot za razpravo o morebitnih pomanjkljivostih in nadaljnjih usmeritvah. S tem sproti preprečujemo morebitne odklone in zagotavljamo, da izdelana funkcionalnost res ustreza potrebam naročnika.

12. Dosledno izvajanje sestankov za oceno kakovosti razvojnega procesa
Ti sestanki služijo za stalno izboljševanje razvojnega procesa. Razvijalcem dajejo možnost, da na podlagi analize pozitivnih in negativnih izkušenj iz prejšnje iteracije definirajo izboljšave, ki jih bodo uvedli v naslednji iteraciji. Seveda je treba poudariti, da so ti sestanki koristni samo, če se dogovorjeni ukrepi potem tudi izvajajo.

13. Dobro orodje za spremljanje poteka del na projektu
Čeprav Scrum ne zahteva obsežne dokumentacije, je koristno imeti orodje, ki nam olajša spremljanje poteka del na projektu. Tovrstna orodja omogočajo vzdrževanje seznama zahtev, seznamov nalog za posamezne iteracije, beleženje podatkov o vloženem in preostalem delu, grafične prikaze napredka ipd.

Rezultati

Iz rezultatov, ki so prikazani na sliki 4, je razvidno, da anketiranci pripisujejo največji pomen dobri komunikaciji med člani razvojne skupine (vprašanje 4) in dobri komunikaciji s produktnim vodjo (vprašanje 2). To je tudi razumljivo, saj za Scrum (tako kot za vse agilne metode) velja, da stalna in kakovostna komunikacija nadomešča pisanje obsežne in zahtevne dokumentacije.

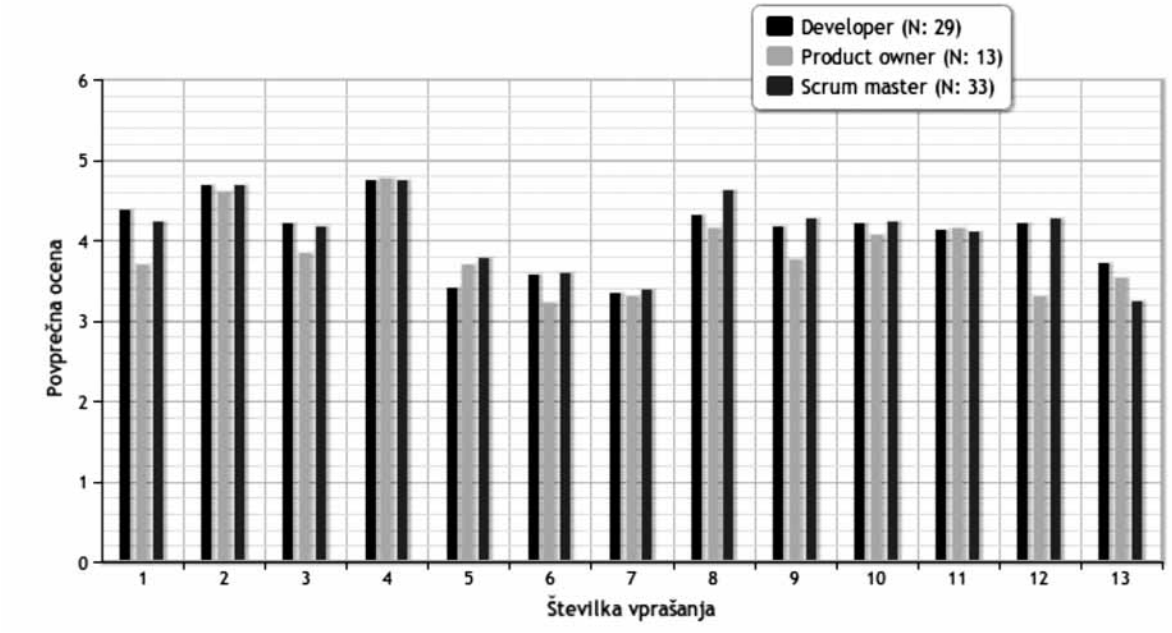


Slika 4: **Povprečne vrednosti odgovorov na drugi sklop vprašanj**

Po drugi strani pa vidimo, da so bile najnižje ocenjene tiste prakse, ki se nanašajo na ocenjevanje hitrosti razvoja (vprašanje 7), načrtovanje izdaje (vprašanje 6) in ocenjevanje zahtevnosti posameznih zgodb (vprašanje 5). V določeni meri je to presenetljivo, še posebno zato, ker so ocena hitrosti razvoja in ocene posameznih zgodb podlaga za načrtovanje posameznih iteracij, načrt izdaje pa zagotavlja pregled nad projektom kot celoto. Po drugi strani pa je takšen rezultat razumljiv, saj Scrum večinoma upora-

bljamo v agilnih okoljih, v katerih je v ospredju prilagajanje spremembam v zahtevah, medtem ko sta natančnost načrtovanja in sledenje načrtu drugotnega pomena. Sorazmerno nizko je bila ocenjena tudi potreba po ustreznem orodju za spremljanje poteka del, kar pa potrjuje, da je metoda Scrum razmeroma preprosta za uporabo, mnogi uporabniki pa namesto računalniške podpore raje uporabljajo fizične kartice in tablo, na kateri s prestavljanjem kartic prikazujejo, kako napreduje delo na projektu.

Analiza odgovorov glede na vlogo anketiranca je prikazana na sliki 5. Čeprav pri večini vprašanj med odgovori ni bistvenih razlik, velja opozoriti na nekatera odstopanja v odgovorih produktnih vodij, ki imajo pri nekaj vprašanjih opazno nižjo vrednost od ostalih. Ta razlika je najbolj očitna pri odgovorih na vprašanje 12, ki govori o doslednem izvajanju sestankov za oceno kakovosti razvojnega procesa (angl. *Sprint Retrospective Meetings*). To se da preprosto razložiti z dejstvom, da pravila metode Scrum ne zahtevajo obvezne prisotnosti produktnega vodje na teh sestankih, saj so izboljšave razvojnega procesa predvsem v domeni razvijalcev, za njihovo uvajanje pa mora skrbeti skrbnik metodologije. Zato so ti sestanki za produktnega vodjo pogosto manj opazni in posledično izpadejo manj pomembni. Seveda pa ne bi bilo nič narobe, če bi se tudi produktni vodja bolj zavedal prave vrednosti teh sestankov. S tem ko bolje teče delovni proces, namreč veliko pridobi tudi on: prej dobi svoj izdelek, ta pa je tudi bolj kakovosten. Zato predlagamo, da se ta vidik projekta še posebno predstavi produktnemu vodji, katerega – če je to mogoče – povabimo k sodelovanju na omenjenih sestankih.



Slika 5: **Povprečne vrednosti odgovorov na drugi sklop vprašanj glede na vlogo anketiranca v metodi**

V primerjavi z ostalimi so produktni vodje nižje ocenili tudi pomen jasno zastavljenega seznama vseh zahtev (vprašanje 1) in načrta izdaje (vprašanje 6). Pričakovali bi, da bo produktni vodja tem vidikom projekta pripisoval večjo vlogo, saj sta tako eden kot drugi tesno povezana z njegovim delom. Prav produktni vodja je odgovoren za izdelavo in upravljanje seznama zahtev, dober seznam zahtev pa je prvi pogoj za izdelavo načrta izdaje, ki produktnemu vodji daje informacijo o časovnem poteku projekta in mu omogoča, da pravilno načrtuje dinamiko uvajanja nove programske opreme v svoji organizaciji.

Nižje vrednosti odgovorov na vprašanje 1 lahko razložimo s predpostavko, da se zdijo zahteve produktnemu vodji jasne in samoumevne, zato upravljanju seznama zahtev ne pripisuje takega pomena kot drugi udeleženci v razvojnem procesu. Pri tem se očitno ne zaveda, da je seznam zahtev gonilo vsega razvoja in vpliva na vse druge aktivnosti. Zato je nujno, da produktnega vodjo na to še posebej opozo-

rimo. Težje pa je razložiti nižje vrednosti odgovorov na vprašanje 6. Razlagi za takšen rezultat bi lahko bili dve. Mogoče je, da pri t. i. projektih »O*n-going*« in »S*tart-up*«, pri katerih zaradi nejasnih in spremenljivih zahtev največ uporabljamo agilne metode, produktni vodje časovnemu vidiku projekta ne dajejo takšnega poudarka, kot to navadno pričakujemo. Pogoste spremembe v zahtevah lahko tudi povzročijo, da postanejo načrti izdaj (če jih ne sproti vzdržujemo) hitro neuporabni, kar še dodatno zniža oceno o njihovi koristnosti. Drugi razlog bi lahko bil v preslabem poznavanju metodologije in nerazumevanju pomena, ki ga ima načrt izdaje za potek projekta. Na to opozarja tudi Cohn (2005), ki pravi, da načrt izdaje služi kot kažipot k cilju, h kateremu naj napreduje projekt, brez njega pa se razvijalci le brezkončno pomikajo od iteracije do iteracije. Vsekakor bi bilo zanimivo, če bi to vprašanje podrobneje raziskali v kateri od prihodnjih podobnih raziskav.

Za razliko od produktnih vodij pa je videti, da skrbniki metodologije dajejo načrtovanju večji poudarek. To se odraža v odgovorih na vprašanji 5 in 8, pri katerih so ocene skrbnikov metodologije višje kot pri drugih vlogah. Omenjeni vprašanji se nanašata na točnost ocen zahtevnosti posameznih zgodb in dosledno izvajanje sestankov za načrtovanje iteracije. Oba dejavnika sta povezana s tekočim potekom projekta in vplivata na vsebino in obseg dela v posameznih iteracijah. Ker so skrbniki metodologije odgovorni za nemoten potek dela, je povsem razumljiva njihova skrb za ta faktorja. Delno pa je lahko povečana skrb za pravilno ocenjevanje zgodb in načrtovanje posameznih iteracij tudi posledica zgodovinske obremenjenosti nekaterih skrbnikov metodologije s klasičnim vodenjem projektov, ki (včasih tudi nehote) enači vlogo skrbnika metodologije z vlogo vodje projekta, čeprav metoda Scrum temu nasprotuje in zahteva, da razvojna skupina deluje po načelu samoorganizacije, skrbnik metodologije pa skrbi za njeno pravilno uporabo in zagotavljanje optimalnih pogojev za delo.

5 SKLEP

Naša raziskava potrjuje, da se uporabniki metode Scrum v celoti strinjajo s trditvami o prednostih te metode, ki jih navaja literatura, še zlasti s tem, da Scrum omogoča razbitje projekta na zaporedje manjših obvladljivih delov in izboljša komunikacijo med člani razvojne skupine. Prav komunikacija pa je po mnenju anketirancev najpomembnejši dejavnik, ki vpliva na uspešnost po Scrumu vodenih projektov. To vključuje tako komunikacijo med razvojno skupino in produktnim vodjo, kot tudi komunikacijo znotraj razvojne skupine in s skrbnikom metodologije. Posebno je pomembno, da stopnja strinjanja s prednostmi metode Scrum narašča sorazmerno z večanjem praktičnih izkušenj njenih uporabnikov.

Druga, nekoliko presenetljiva ugotovitev govori o tem, da uporabniki metode Scrum ne pripisujejo tako velike vloge ocenjevanju in načrtovanju časovnega poteka projekta, kot bi lahko pričakovali. To lahko razložimo z naravo projektov, pri katerih uporabljamo Scrum. Pogosto gre za spletne projekte in projekte »*Start-up*«, pri katerih sledimo principu »*Release early, release often*«. Pri tem principu velikokrat pravzaprav ne moremo govoriti o roku izvedbe, saj se izdelek neprestano izboljšuje in dodeluje, izdaje pa se dogajajo pogosto in inkrementalno. To bi lahko potrdil tudi komentar enega od anketirancev: »*Težko je reči, da naročnik dobi dogovorjeno funkcionalnost v roku – ta je namreč fleksibilna in razširljiva ter se sproti prilagaja razmeram na projektu.*«

6 VIRI IN LITERATURA

[1] Ambler, S. W. (2008). Has agile peaked? Let's look at the numbers. *Dr. Dobb's Journal*. http://www.ddj.com/architect/207600615?pgno=1.

[2] Begel, A. & Nagappan, N. (2007). Usage and preceptions of agile software development in an industrial context: an exploratory study. Zbornik prispevkov: *First International Symposium on Empirical Software Engineering and Measurement*. Madrid, Španija.

[3] Boehm, B. & Turner, R. (2004). *Balancing Agility and Discipline*. Boston, MA: Addison-Wesley.

[4] Cohn, M. (2004). *User stories applied for agile software development*. Boston, MA: Addison-Wesley.

[5] Cohn, M. (2005). *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice Hall PTR.

[6] Dybå, T. & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50, 9–10, 833–859.

[7] Laanti, M., Salo, O. & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology*, 53, 3, 276–290.

[8] Mahnič, V. (2010). Teaching Scrum through team-project work: students' perceptions and teacher's observations. International Journal of Engineering Education, 26, 1, 96–110.

[9] Mahnič, V. (2011). Problemi in rešitve pri uvajanju metode Scrum v proces razvoja programske opreme. Zbornik prispevkov: *18. konferenca Dnevi slovenske informatike*. Portorož, Slovenija.

[10] Mahnič, V. (2012). A Capstone Course on Agile Software Development Using Scrum. IEEE Transactions on Education, 55, 1, 99–106.

[11] Murphy, T., Duggan, J., Norton, D., Prentice, B., Plummer, D. & Landry, S. (2009). Predicts 2010: Agile and Cloud Impact Application Development Directions, Gartner. http://www.gartner.com/DisplayDocument?id=1244514.
[12] Rising, L. & Janoff, N. S. (2000). The Scrum software development process for small teams, *IEEE Software*. 17, 4, 26–32.
[13] Schwaber, K. (2004). *Agile Project Management with Scrum*, Redmond, WA: Microsoft Press.
[14] Scotland, K. & Boutin, A. (2008). Integrating Scrum with the process framework at Yahoo! Europe, Zbornik prispevkov: *Agile 2008*. Toronto, Canada.
[15] Urevc, J., Štebe, R. & Mahnič, V. (2012). Izkušnje z uvajanjem metode Scrum v časopisni hiši Delo, Zbornik prispevkov: *19. konferenca Dnevi slovenske informatike*. Portorož, Slovenija.
[16] Version One (2011). State of Agile Survey. http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf.
[17] West, D. & Grant, T. (2010). Agile Development: Mainstream Adoption Has Changed Agility. Forrester research. http://www.forrester.com/rb/Research/agile_development_ mainstream_adoption_has_changed_ agility/q/id/56100/t/2.
[18] Williams, L. (2010). Agile software development methodologies and practices. *Advances in Computers*. 80, 1–44.

▪

Janez Urevc je diplomiral na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. V svojem diplomskem delu se je ukvarjal z uvedbo agilne metode Scrum v oddelek spletnega razvoja časopisne hiše Delo, pri tem pa posebno pozornost namenil ovrednotenju njenih prednosti, merjenju učinkovitosti razvojne skupine in analizi točnosti ocenjevanja uporabniških zgodb. Poleg agilnih metod ga zanimajo spletne in mobilne tehnologije, predvsem v povezavi s prosto programsko opremo. V času študija je dve leti deloval kot vodja MMC Kiperpipa, sedaj pa deluje kot samostojni razvijalec ter največ časa posveča raziskovanju najaktualnejših tehnologij na spletu in mobilnih platformah. Svoje prispevke objavlja v različnih poljudnih in strokovnih publikacijah ter konferencah.

▪

Viljan Mahnič je izredni profesor in predstojnik Laboratorija za tehnologijo programske opreme na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Pri svojem pedagoškem in raziskovalnem delu namenja posebno pozornost agilnim metodam za razvoj programske opreme in informacijskih sistemov, metrikam v programski opremi ter zagotavljanju učinkovitosti in kakovosti razvojnega procesa. Rezultate svojih raziskav je objavil v več uglednih znanstvenih revijah, redno pa sodeluje tudi na domačih in mednarodnih konferencah. Doslej je vodil več projektov s področja razvoja informacijskih sistemov, še zlasti na področju študijske informatike, pomaga pa tudi pri uvajanju metode Scrum v slovenska podjetja.

Informatika mora pokazati svojo poslovno vrednost

Mitja Cerovšek
TPV – trženje in proizvodnja opreme vozil, d. d., Kandijska cesta 60, 8000 Novo mesto
m.cerovsek@tpv.si

Izvleček

Iskanje vzvodov za povečevanje uspešnosti in učinkovitosti je naloga vodstvene strukture v podjetju. V času velikih sprememb in pretresov v širšem poslovnem okolju podjetja intenzivno iščejo ustrezne rešitve za povečevanje svoje konkurenčnosti in inovativnosti. Informatika v podjetju omogoča nastajanje novih poslovnih modelov in je vir novih priložnosti, ustvarja nove proizvode in privablja nove kupce. Z naraščanjem pomena informatike narašča tudi njena odgovornost. V tej luči se mora informatika izrazito usmerjati v razumevanje in povečevanje svoje poslovne vrednosti, ki se nahaja v ljudeh, poslovnih procesih in informacijski tehnologiji. Usmerjena uporaba informacijske tehnologije, menedžment poslovnih procesov in vključevanje ter razvoj zaposlenih so ključni vzvodi povečevanja poslovne vrednosti informatike. To pa je tudi edina pot, da ta postane in ostane v svoji polni moči bistveni del prihodnje zmagovite organizacijske in procesne strukture delovanja podjetja.

Ključne besede: menedžment poslovnih procesov, poslovna odličnost, poslovna vrednost, prenova in informatizacija poslovanja, strateško načrtovanje.

Abstract

IT Needs to Demonstrate its Business Value

Looking for instruments which will increase effectiveness and efficiency is an important task of the entire management structure of a company. In times of great changes and stresses in broader business environment companies intensively search for appropriate solutions in order to increase their competitiveness and innovativeness. The IT department of the company drives the creation of new business models and at the same time presents a source of new opportunities, designs new products and attracts new customers. While the importance of information technology rises, its responsibility also increases. Consequently, the information technology has to be focused on the understanding and increase of its business value which is located in the people, business processes and information technology. Focused use of information technology, business process management and integration and staff development are the key levers to increase business value of IT. That is the only way it should follow in order to become and remain a powerful as well as crucial part of the future winning organisational and processing structure of the company.

Keywords: business process management, business excellence, business value, process of business reform and informatisation, strategic planning.

1 UVOD

Vloga informatike v podjetju je pomemben pokazatelj usmerjenosti, urejenosti in organiziranosti poslovnega okolja, kar posledično lahko povezujemo z uspešnostjo in učinkovitostjo vsega delovanja podjetja. Uspešnost poslovanja je prvi cilj vsake vodstvene strukture, ki mora v času tehnološkega napredka in obenem v težkih gospodarskih razmerah na trgu poskrbeti za polno vključitev vseh procesov, okolij in orodij, ki lahko prispevajo k povečevanju dodane vrednosti izdelkov in storitev. Vse bolj pomembno postaja delati prave stvari (uspešnost) na pravi način (učinkovitost).

Tudi informatika pri tem ni nobena izjema. Pogosto poudarjamo, da je sodobna, procesno in v poslovanje usmerjena informatika vir novih poslovnih priložnosti, novih poslovnih modelov in živčni sistem, po katerem se pretakajo sposobnosti, informacije, znanja in kompetence podjetja. To seveda pomeni, da je lahko informatika ključno orodje za delovanje v zahtevnem in konkurenčnem poslovnem okolju, lahko pa je tudi ključna zavora v poslovnem okolju podjetja. Če izključimo notranje in zunanje motnje, ki vplivajo na delovanje informatike, in predpostavimo,

da so njeni rezultati odvisni izključno od dela in sposobnosti vseh, ki se ukvarjajo z njo, trčimo na tole vprašanje: kako lahko informatika danes prispeva k uspešnosti in učinkovitosti delovanja podjetja? Kaj lahko informacijska tehnologija danes ponudi podjetju, da bodo rezultati poslovanja zaradi delovanja informatike boljši? Govorimo torej o vprašanju, ki bi moralo skrbeti najprej ves menedžment, potem pa tudi vse druge strukture, ki se zavedajo vloge in pomena informatike v podjetju.

Če se torej zavestno osredinimo le na informatiko in izključimo notranje in zunanje motnje, ki ji pretijo, imamo pred seboj jasen izziv. Čas je, da informatika sporoči in jasno pove podjetju, katera so področja njenega delovanja, zaradi katerih je podjetje uspešnejše in učinkovitejše. Informatika je v tem kontekstu najprej dolžna podjetju jasno predstaviti svoje poslanstvo, potem pa ga je dolžna tudi opraviti in ovrednotiti. To je zahtevna in zelo ambiciozna naloga, ki jo lažje sprejme in opravi sposobna, motivirana in opolnomočena informatika, ki se bolj kot s tehnologijo ukvarja s poslovanjem podjetja.

2 ODLIČNOST JE ODLIČNO IZHODIŠČE ZA ODLIČNO POSLOVANJE

Zavedanje in odločitev o odličnosti delovanja na osebni in poslovni ravni je prvi korak na poti k zavestnemu izboljševanju in stalnemu napredku. Visoki cilji, ki si jih postavijo posamezniki in organizacije, običajno pa jih sprožajo tudi zahteve kupcev, kličejo po celostni odličnosti misli in dejanj. Jasno je, da iz slabega ne more nastati dobro, zato je kakovost ravnanj pomemben dejavnik univerzalne odličnosti. To je odlično izhodišče tudi za informatiko in za izpolnjevanje njenega poslanstva v podjetju. Ključne usmeritve odličnosti (Otoška rastoča strategija univerzalne odličnosti in mojstrstva, 2010), ki jih lahko razvijamo vsi, udejanjamo na področjih a) vzgoje, izobraževanja in učenja, b) sproščanja človekovih ustvarjalnih potencialov, c) družbeno odgovornega sonaravnega delovanja, č) usmerjenosti v presežke ter d) etičnosti.

Evropski model poslovne odličnosti EFQM opredeljuje področja dejavnikov za doseganje poslovne odličnosti ter usmerja in spodbuja organizacije k doseganju najvišjih standardov delovanja v družbi. Pospešuje zavedanje o evropskem modelu upravljanja odličnosti poslovanja, spodbuja procese samoocenjevanja in primerjave med organizacijami v mednarodnem okolju, priznava in nagrajuje dosežke na področju odličnosti in prenaša dobre prakse za doseganje trajno odličnih rezultatov. Gre za orodje, ki prepozna pet dejavnikov (voditeljstvo, zaposleni, politika in strategija, partnerstva in viri, procesi), ki vplivajo na rezultate poslovanja (slika 1).

Izjemno pomemben dejavnik znotraj modela poslovne odličnosti so procesi. Prav to pa je področje, na katerega s svojim delovanjem neposredno posega informacijsko okolje. Za podjetje se tu odpira izjemna priložnost, da izkoristi možnosti, ki jih pri razvoju poslovnih procesov in novih poslovnih modelov omogoča informacijsko okolje. Poslovni procesi so namreč središčna os podjetja, zato si zaslužijo izjemno pozorno obravnavo. Storitve in izdelke proizvajamo v poslovnih procesih. Pogosto jih je težko prepoznati v njihovi raznolikosti in zapletenosti, še posebno ker so pogosto skriti za funkcijskimi organizacijskimi strukturami. Izjemno težko pa jih prepoznamo in nanje pogledamo na način procesne

Slika 1: **Evropski model poslovne odličnosti**

usmerjenosti organizacijskih struktur. Tu vstopamo na področje prenove in informatizacije poslovanja ter menedžmenta poslovnih procesov, ki prepoznava ves sociološko-tehnični okvir delovanja organizacije. Gre torej za širok spekter medsebojno odvisnih področij, ki zahtevajo obvladovanje menedžmenta sprememb. Uspešnost poslovanja in povečevanje konkurenčnosti je (Groznik idr., 2005: 216) pogosto povezana s prilagajanjem ali tudi s korenito spremembo poslovne strategije, ki se udejanja v spremembi poslovnega modela in poslovnih procesov podjetja. To lahko opravijo le ustrezno usposobljeni in zelo motivirani kadri ob uporabi sodobne informacijske tehnologije.

Informatika torej lahko močno vpliva na kakovost delovanja in poslovanja podjetja. Načelo odličnosti delovanja na področju informatike je torej izvrstno izhodišče za iskanje odgovora na vprašanje, kako lahko informatika povečuje uspešnost in učinkovitost podjetja. Če se del tega skriva v razumevanju in uporabi njene poslovne vrednosti, potem jo je vredno poiskati in opredeliti ter se ji posvetiti bolj podrobno.

3 POSLOVNA VREDNOST INFORMATIKE

Sorazmerno s povečevanjem pomena informatike narašča tudi njena odgovornost. To je dober razlog, da se izrazito posvetimo razumevanju in obvladovanju njene poslovne vrednosti. Informacijsko okolje je pogosto predmet različnih obravnav in analiz na temo njegove vloge in pomena v podjetju. Prva skupna značilnost ugotovitev je največkrat ta, da so te silno različne. Informacijska industrija ter ponudniki opreme in storitev se prerivajo in so glasni v prepričevanju svojih kupcev, da prav oni tlakujejo pot v poslovno uspešnost in da prinaša uporaba zadnjih tehnoloških izdelkov vozovnico v obetavno poslovno prihodnost. Na drugi strani pa stoji z realnostjo in iz nje izhajajočimi izzivi soočeno gospodarstvo, ki – tudi zaradi novih razmer na svetovnih trgih – vse bolj razumno in ciljno razporeja svoje vire. Jasen odgovor na vprašanje, kako lahko informatika prispeva k povečevanju poslovne uspešnosti, bi koristil obema. Žal so redka okolja, v katerih je to vprašanje središče razmišljanja in delovanja menedžerja informatike in menedžerja podjetja.

Ključna naloga menedžerja informatike je prav gotovo strateško načrtovanje razvoja informatike v podjetju. Splošno na strateški ravni načrtujemo in obravnavamo vsebine, ki so bistvenega pomena za uspešno in učinkovito delovanje sistemov. Iz tega sledi, da so v tovrstno obravnavo vključene poslovnoinformacijske vsebine, ki pomembno prispevajo k ustvarjanju in povečevanju poslovne vrednosti informatike. Informacijska tehnologija pa poslovno vrednost ustvarja skupaj z drugimi področji delovanja podjetja. Predvsem tu mislimo na zaposlene z njihovimi znanji in ustrezno produktivnostjo, ki jo dosežejo tudi z uporabo sodobnih informacijskih okolij. Ključni dejavniki strateškega načrtovanja informatike in posledično glavni nosilci poslovne vrednosti informatike (Groznik in Vičič, 2005a: 200) so:

- kadri in znanja,
- poslovni procesi,
- informacijska tehnologija.

Informacijsko okolje samo po sebi (brez jasnih dodatnih utemeljitev) ne pomeni konkurenčne prednosti. Za mnoge je to žal še vedno le neizogiben strošek. Tako pogosto razmišljajo predvsem v okoljih, v katerih na informatiko še vedno gledajo pretežno kot na tehnologijo, povsem pa zanemarijo druge (bistvene) elemente informacijskega okolja, kot so poslovni procesi, izdelki in storitve z novo vsebino in večjo dodano vrednostjo ter novi poslovni modeli. Bistveno je, da na poslovno vrednost informatike gledamo skozi izboljšave, ki jih informatizacija prinese v poslovne procese (Groznik idr., 2005: 213). To pa je področje, ki bi moralo še kako zanimati oba: menedžerja informatike in menedžerja podjetja.

Neprestane rasti naložb v informatiko v časih hitre gospodarske rasti ni mogoče neposredno povezovati tudi s povečevanjem poslovne uspešnosti, ki bi posledično nastala na tej podlagi. Ob tem naletimo na glavno zadrego informatike: pokazati in dokazati, da lahko informatika deluje tudi (ali pa predvsem) kot generator poslovnih priložnosti in da je nikakor ne gre razumeti kot generatorja stroškov. Pri tem se je treba problema lotiti z dveh vidikov (Groznik idr., 2005: 214–217): najprej z vidika izbire ustreznih meril merjenja učinkov informatike, nato pa tudi z vidika izbiranja pravih naložb. Ob tem seveda ni najbolj pomembno, koliko vlagamo, pač pa predvsem, kam vlagamo in kakšni so vplivi naših naložb. Ti so lahko merljivi (višja produktivnost, nižji operativni stroški, višja dodana vrednost, nižji prodajni stroški, nižji stroški administracije itn.), zaradi posebnosti in specifične vloge informatike v podjetju pa so lahko tudi nemerljivi (večje zadovoljstvo kupcev, večja prilagodljivost poslovanja, višja kakovost informacij,

izboljšanje procesa načrtovanja, povečanje zadovoljstva zaposlenih itn.). Naloga menedžerja informatike je, da razume te vidike in jih upošteva pri načrtovanju svojih prihodnjih usmeritev in investicij. Predvsem pa je pomembno, da potencialne priložnosti razumljivo predstavi menedžerju podjetja in z njim vzpostavi kakovosten način komunikacije in hoje k istim skupnim ciljem.

Poslovna vrednost informatike se zanesljivo kaže tudi v njenem strateškem delovanju in njeni strateški vlogi v podjetju. Vse bistvene usmeritve in projekte informatike je treba trdno povezati s ključno smerjo delovanja vsega podjetja, ki je določena v njegovem strateškem načrtu. Iz teh izhodišč lahko izhaja strateški načrt razvoja informatike, ki je potreben, ne pa tudi zadosten pogoj za uspešno in učinkovito delovanje informatike. Zdi se, da so danes pričakovanja vseh menedžerskih struktur, ki naj bi jih izpolnila informatika, vezana predvsem na njeno tehnološko vlogo (kar je pomembno, ne pa najbolj bistveno), manj pa so vezana na njeno poslovno vlogo in umestitev v središče znanj, organizacije in poslovnih procesov. Prav to pa je ključna in najbistvenejša vloga informatike, iz katere podjetje lahko črpa kakovostno in konkretno poslovno vrednost.

4 VZVODI INFORMATIKE ZA USPEŠNO IN UČINKOVITO POSLOVANJE

Za informatiko velja, da težko ovrednotimo njen prispevek k rezultatom poslovanja. Poleg merljivih koristi namreč obstaja tudi velik nabor nemerljivih koristi, ki so za podjetje zelo dragocene. Težko jih je ovrednotiti, vsi pa vemo za njihov posredni vpliv na uspešnost poslovanja. Realno moč informatike lahko v kontekstu našega izhodiščnega vprašanja (Kaj mora informatika dati podjetju?) gradimo na treh stebrih:

- na usmerjeni uporabi informacijske tehnologije,
- menedžmentu poslovnih procesov,
- vključevanju ter razvoju zaposlenih.

Vzvodi usmerjene uporabe informacijske tehnologije

Strateški načrt razvoja informatike, v katerem so zapisani konkretni cilji, ki jih zasledujemo, usmerja energijo, znanja in finančna sredstva k pravim vsebinam. Te morajo slediti izzivu povečevanja konkurenčnosti in dodane vrednosti izdelkov in storitev. Zato je, ko govorimo o dolgu informatike do podjetja, izjemno pomembno, na katera področja menedžer informatike usmerja svoje delovanje, za katere projekte se »bori«, katere strokovne in poslovne cilje zasleduje. Sredstva, ki so na voljo v omejenem obsegu, je treba učinkovito in modro usmerjati v projekte, ki vodijo do povečevanja vrednosti za podjetje. Manfreda in Štemberger (2011) ugotavljata, da kljub znanemu dejstvu o dveh različnih svetovih (poslovno okolje, tehnološko okolje) še obstaja precejšen razkorak med poslovnim in informacijskim svetom podjetja, kar posledično vodi v številne neuspešne in nekoristne zaključke informacijskih projektov.

Informatika mora s svojim delovanjem povečevati uspešnost in učinkovitost vsega podjetja. Investirati ne sme brezciljno (po načelu malo tu, malo tam), pač pa predvsem strateško. Negativne posledice nepremišljenega investiranja se lahko posledično kažejo (Ward, 2002) v izdelkih, ki ne podpirajo poslovnih usmeritev podjetja, v nepovezanih aplikacijah, nejasnih prioritetah pri projektih, slabem vrednotenju projektov, pogostih spremembah planov in nezadostnih infrastrukturnih investicijah. Strateško planiranje (Rožanec in Krisper, 2009: 123–127) spada med ključne procese v informatiki, saj je potrebno za upravljanje vseh informacijskih virov in za zagotavljanje njihove skladnosti s poslovno strategijo, kar pripomore k doseganju dolgoročne uspešnosti organizacije. Prispevek k uspešnosti poslovnega sistema, ki ga dosežemo s skrbnim in usmerjenim načrtovanjem razvoja informatike, lahko merimo s stopnjo povečanja donosnosti naložb, zvišanja tržnega deleža izdelkov in storitev, povečanja učinkovitosti notranjih operacij, povečanja prihodkov od prodaje, povečanja zadovoljstva strank in usklajenosti informatizacije s poslovnimi potrebami.

Vzvodi tako opredeljene usmerjene uporabe informacijske tehnologije, ki temeljijo na dojemanju informatike kot poslovne (in ne tehnološke) priložnosti, so zato lahko tile:

- povezovanje ciljev informatike s cilji podjetja,
- prehod informatike od tehnologije k poslovanju,
- preoblikovanje informatike v vir ustvarjanja konkurenčne prednosti,
- usmerjanje informatike v razvijanje novih poslovnih modelov,
- gradnja korporativnih načel poenotene in standardizirane odličnosti delovanja informatike.

Usmerjena uporaba informacijske tehnologije podjetju pomaga najoptimalneje razporediti razpoložljive vire in z njimi učinkovito (usmerjeno)

gospodariti. Tako delovanje, v nadaljevanju podprto z rezultati, sproža pri sodelujočih pozitivne odzive in pripravljenost za sodelovanje.

Vzvodi menedžmenta poslovnih procesov

Kot svoje obetavno polje delovanja lahko informatika sprejme razumevanje, razvoj in obvladovanje poslovnih procesov, v čemer se skriva bistvo uspešnega in učinkovitega delovanja organizacije. Menedžment poslovnih procesov (angl. Business Process Management – BPM) je razširjeni pogled na prenovo poslovanja in vključuje širok spekter dejavnikov celovitega obvladovanja poslovnih procesov. Podjetje se pri tem pristopu (Kovačič in Bosilj Vukšič, 2005: 45–48) sooča s sociološko-tehničnim okvirom delovanja organizacije, ki ga sestavljajo kulturni, strukturni, kadrovski, tehnološki in procesni vidik. Gre torej za kompleksno in medsebojno povezano sestavo številnih dejavnikov procesnega delovanja podjetja, ki pa jih obravnavamo celovito, na podlagi interdisciplinarnega pristopa in ustreznih metodoloških izhodišč.

Ko informatika aktivno posega na področje menedžmenta poslovnih procesov, v svojem jedru spreminja način razmišljanja in delovanja ter sebi, predvsem pa podjetju, prinaša nove dimenzije razvoja. V središče pozornosti se torej pomikata osredinjenost na kupca in osredinjenost na verigo dodane vrednosti (Spanyi, 2007: 27 in 110–113). Spanyi procese v organizaciji poimenuje kar tretja dimenzija menedžmenta v kompleksnih organizacijskih strukturah. Podjetja kot prva dimenzija opredeljujejo, kje delamo, funkcije kot naslednja določajo, kaj delamo, procesna usmeritev kot tretja dimenzija menedžmenta pa se osredinja na vprašanje, kako delamo. Nenehni proces učenja, ki se ob tem sproža v podjetju, vodi k spremembam v smeri procesnega delovanja in uresničevanja procesne organiziranosti s poudarkom na področjih vodenja, menedžmentu procesne organiziranosti, večjem opolnomočenju zaposlenih, procesu komunikacije, participativne strategije in prilagodljive kulture (Daft in Marcic, 2004).

Pripravljenost organizacije, da v realnem gospodarstvu deluje procesno, je prvi korak na izjemno zahtevni poti, da iz funkcijske miselnosti preklopimo na procesno. Namesto hierarhičnih struktur gre za poudarjanje procesov, njihovih rezultatov ter zadovoljstva strank. Vsi zaposleni, ki so dodeljeni določenemu procesu, neposredno sodelujejo med seboj in koordinirajo svoje delovanje ter neposredno ustvarjajo vrednost za kupca (Dimovski idr., 2003: 145–146). Meje med oddelki so presežene, lastniki procesov pa so zadolženi za poslovni proces v celoti. Zaposleni znotraj posameznega tima so kompetentni, usposobljeni in motivirani za doseganje vrednosti za kupca. Timi so svobodni v razmišljanju, kreativni in prožni. Učinkovitost se meri z zadovoljstvom kupcev in zaposlenih. Cenijo se odprtost, zaupanje, sodelovanje in stalne izboljšave. Povečata se moč odločanja zaposlenih in tudi njihova odgovornost.

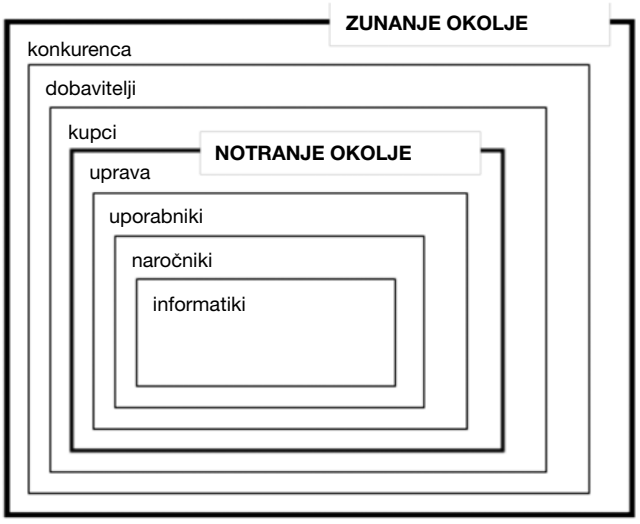
Podjetje tega ne more narediti čez noč. To ni mogoče in tudi ni primerno. Zanesljivo je to stvar premišljene odločitve, da vsi v podjetju želimo stopiti na pot, ki postopno sproža preobrazbo v procesno delovanje z vsemi posledicami tega dejanja. Aktivno sodelovanje informatike na področju menedžmenta poslovnih procesov je naravno širjenje polja njenega delovanja in neposreden pozitiven preskok v kakovosti in vsebini njenega delovanja. Informatika je lahko pobudnik in močno orodje za postopno izvedbo česa takega, saj ima pri tem velikanski motiv: v tem se namreč skriva njena priložnost, tudi prihodnost in predvsem njena odgovornost do podjetja.

Vzvodi vključevanja in razvoja zaposlenih

Organizacijski in kadrovski vidik problema informatike v vlogi pospeševalca uspešnosti in učinkovitosti podjetja je področje, ki pogosto močno zaostaja za ravnjo obvladovanja tehnološke plati tega problema. Kadrovska zasedba, ki sodeluje pri dejanjih načrtovanja, izvedbe, preverjanja in ukrepanja (koraki kroga PDCA) pri procesih obvladovanja informacijskega okolja, je zelo široka (slika 2) in presega le zaposlene znotraj posamezne organizacije.

Informatiki so pri tem le središčna točka vseh deležnikov, okrog katere se zbirajo še naročniki informacijskih rešitev in storitev, uporabniki in uprava podjetja. Iz notranjega poslovnega okolja se okvir deležnikov širi na kupce, dobavitelje in konkurenco, torej v t. i. zunanje poslovno okolje. Odgovornost za obvladovanje poslovne vrednosti informatike je velika in v smislu učinkov presega okvire podjetja, kaj šele informatike.

Konkurenčnost in tekmovalnost kličeta po učinkoviti uporabi razpoložljivih virov. Izobraženi zaposleni, izobraženi kupci, globalni trg in razpoložljivost informacijske tehnologije spreminjajo moč poslovanja organizacije in jo še bolj močno usmerjajo k zaposlenim z ustreznimi znanji. Priča smo premiku proti dodani vrednosti, ki jo narekuje in določa kupec. Po-



Slika 2: **Kadrovski model informacijskega okolja podjetja**

večuje in razvija se vrednost človeškega kapitala kot skupno dobro podjetja in družbe. Prav zaradi tega je razvoju zaposlenih smiselno in potrebno posvečati izjemno pozornost, tudi zaradi njihove čvrste vpetosti v mrežo informacijskega okolja in obvladovanja poslovnih procesov podjetja. So namreč pomembni gradniki spreminjanja organizacije in doseganja kritične mase energij in znanj, ki so potrebna za doseganje dovolj velike stopnje konkurenčnosti na trgu. Zato je izjemno pomembno, da v podjetju vzpostavimo pregledne sisteme sodelovanja pri zaznavanju in izvedbi informacijskoprocesnih potreb. Notranja izmenjava znanj ter način dela v »zračnem delovnem okolju« brez zaprtih vrat in funkcijskih vrtičkov mora postati priznana vrednota in kakovost, zapisana v organizacijskih strukturah podjetja.

Na izzive razvoja zaposlenih in vzpostavitve okolja organizacijskega učenja odgovarja tudi t. i. »učeča se organizacija«, ki v svojem bistvu povečuje komunikacijo in sodelovanje znotraj podjetja, kar organizaciji omogoča neprestano eksperimentiranje, izboljševanje, povečevanje njenih nematerialnih zmogljivosti ter stalno učenje (Dimovski in Penger, 2004). Učečo organizacijo v tem smislu povezujemo s pridobivanjem znanja, z distribucijo informacij, interpretacijo informacij in organizacijskim spominom. Na podlagi povečevanja znanja se povečujejo konkurenčne sposobnosti podjetja.

Z ustrezno vodeno komunikacijo v vseh strukturah organizacije ter z načrtovanim in usmerjenim prenosom strategij, vzorcev delovanja, kulture organizacije, znanj in kompetenc lahko dosežemo (Cerovšek, 2008:105–106) bistveno izboljšanje poznavanja informacijskih rešitev, zagotovimo njihovo kakovostnejšo uporabo, povečamo obseg izkoriščenih poslovnih priložnosti in povečamo razumevanje in poznavanje poslovnih procesov. To seveda pomeni, da moramo vzvode vključevanja in razvoja zaposlenih pomikati tudi onkraj meja organizacije same, kar ponovno krepi in potrjuje misel o veliki obveznosti in odgovornosti, ki jo v razmerju do podjetja nosi informatika.

5 SKLEP

Usmerjenost, urejenost in organiziranost poslovnega okolja se kaže tudi v načinu dojemanja informatike v podjetju in v rezultatih, ki jih ta dosega. Vprašanje povečevanja konkurenčnosti poslovnega okolja v sodelovanju z informatiko je za podjetje pomemben izziv, ki neposredno vpliva na njegovo prihodnost. Vpliva pa tudi na prihodnost informatike same. Določa namreč njeno vlogo v okviru prihodnje organizacijske in procesne strukture delovanja podjetja. Informatika lahko prevzame pobudo in podjetju jasno predstavi in z rezultati podkrepi svojo poslovno vrednost in številne priložnosti, ki jih omogoča v smeri povečevanja konkurenčnosti in dodane vrednosti izdelkov in storitev. Pri tem so vzvodi usmerjene uporabe informacijske tehnologije, vzvodi menedžmenta poslovnih procesov ter vzvodi vključevanja in razvoja zaposlenih mogoče oblike razvoja konkretnih prispevkov informatike k povečevanju uspešnosti in učinkovitosti delovanja podjetja.

6 VIRI IN LITERATURA

[1] Cerovšek, M. (2008). Upravljanje in vloga informatike po formalno končanem obdobju prenove poslovanja, *Uporabna informatika*, 16(2): 103–107.

[2] Daft, R.L. & Marcic, D. (2004). *Understanding Management*, Fort Worth: Thomson Learning, Fourth Edition.

[3] Dimovski, V., Penger, S. & Žnidaršič, J. (2003). *Sodobni management*, Ljubljana: Ekonomska fakulteta.

[4] Dimovski, V. & Penger, S. (2004). Učeča se organizacija: transformacija k horizontalni organizacijski strukturi v dobi ekonomije znanja, *Teorija in praksa*, XLI(5–6): 808–822.

[5] Groznik, A., Štemberger I., M. & Kovačič, A. (2005). Vloga menedžmenta pri zagotavljanju poslovne vrednosti informatike, *Uporabna informatika*, 13(4): 213–222.

[6] Groznik, A. & Vičič, D. (2005a). Vrednost in pomen informatike v podjetju, *Organizacija*, 38(4): 198–202.
[7] Kovačič, A. & Bosilj V., V. (2005). *Management poslovnih procesov: prenova in informatizacija poslovanja s praktičnimi primeri*, Ljubljana: GV Založba.
[8] Manfreda, A. & Štemberger I., M. (2011). Partnership between top management and IT personel – is it really beyond the reach?, European, Meditteranean & Middle Eastern Conference of Information Systems 2011 (EMOIS2011), Athens, Greece, 30.–31. maj 2011, str. 523–530.
[9] *Otoška rastoča strategija univerzalne odličnosti in mojstrstva*. [URL:http://www.fos.unm.si/si/dejavnosti/forum_odlicnosti/strategija/], 20. 1. 2012.
[10] Rožanec, A. & Krisper, M. (2009). Kako meriti uspešnost procesa strateškega planiranja informatike in kako povečati njegovo uspešnost, *Uporabna informatika*, 17(3): 123–136.
[11] Spanyi, A. (2007). *More for Less: The Power of Process Management*, Tampa: Meghan – Kiffer Press.
[12] Ward, J. & Peppard, J. (2002). *Strategic Planning for Information Systems*, New York: John Wiley & Sons, Third Edition.

▪

Mitja Cerovšek je diplomiral na Fakulteti za elektrotehniko Univerze v Ljubljani na smeri avtomatika – procesna informatika. Magistrski študijski program ekonomije je opravil na Ekonomski fakulteti Univerze v Ljubljani. Zaposlen je v Skupini TPV, kjer je direktor za informacijske in komunikacijske tehnologije. Strokovno področje njegovega delovanja obsega strateško načrtovanje razvoja informatike v podjetju, menedžment poslovnih procesov, prenovo in informatizacijo poslovanja ter načrtovanje in vodenje strateških in medorganizacijskih projektov. S svojimi prispevki sodeluje na znanstvenih in strokovnih konferencah, predvsem s poudarkom na koristnem in potrebnem povezovanju akademskih znanj z znanji iz gospodarstva, kar se uspešno uresničuje v praksi avtomobilske industrije.

Iz Islovarja

Islovar je spletni računalniški program, ki ga ureja jezikovna sekcija Slovenskega društva INFORMATIKA. Islovar je odprt za prispevke uporabnikov, njegova prednost je v tem, da ga uredniki Islovarja sproti pregledujejo, posodabljajo in popravljajo. Tudi sestavke, ki so bili pred časom že urejeni, slovaropisna skupina ponovno pretrese in če se ne ujemajo s sorodnimi izrazi, ali so zastareli, spremeni izraze ali njihovo razlago.

Zbirko, ki jo predstavljamo tokrat, smo zbrali pod naslovom »odnosi«. Vključili smo novejše izraze, ki se uporabljajo pri odnosih v informacijski družbi. Večkrat smo se morali odločiti, kateremu izrazu dati prednost, kajti uporaba se še ni ustalila in se za en pojem uporablja več izrazov, npr. socialni, družbeni, družabni.

Vabimo vas, da v Islovar www.islovar.org prispevate svoje pripombe ali predloge.

družábna mréža -e -e ž (*angl. social network*) neustr.
gl. družabno omrežje in socialno omrežje

družábno omréžje -ega -a s (*angl. social network*)
gl. socialno omrežje in družbeno omrežje

drúžbena mréža -e -e ž (*angl. social network*) neustr.
gl. družbeno omrežje in socialno omrežje

drúžbena povezáva -e -e ž (*angl. social link*)
stiki med posamezniki v interesni skupini; sin. socialna povezava

drúžbeni médij -ega -a m (*angl. social media*)
medij (3), ki se uporablja za izmenjavo novic, mnenj, izkušenj med uporabniki; sin. socialni medij

drúžbeno mréženje -ega -a s (*angl. social netting, social networking*)
udejstvovanje v družbenem omrežju; sin. socialno mreženje

drúžbeno omréžje -ega -a s (*angl. social network*)
skupina ljudi, ki jih povezujejo skupni interesi in aktivnosti; sin. socialno omrežje, družabno omrežje

drúžbeno označeváje -ega -a s (*angl. social indexing, social tagging, folksonomy*)
označevanje spletnih dokumentov, ki ga opravi neformalna skupina; sin. folksonomija

drúžbeno prográmje -ega -a s (*angl. social software*)
programje za izvedbo storitev družbenega omrežja, npr. Facebook; sin. socialno programje

elektrónsko zalezovánje -ega -a s (*angl. cyberstalking*)
prikrito zalezovanje z uporabo elektronskih storitev, npr. družbenih medijev

emotívno račnálništvo -ega -a s (*angl. affective computing*)
področje računalništva, ki obravnava sisteme, naprave, ki lahko zaznavajo interpretirajo, obdelujejo in simulirajo človekova občutja

folksonomíja -e ž (*angl. social indexing, social tagging, folksonomy*)
označevanje spletnih dokumentov, ki ga opravi neformalna skupina; sin. družbeno označevanje

kibernétsko nasílje -ega -a s (*angl. cyberbullying*)
pošiljanje grozilnih sporočil, poniževanje, opravljanje ali drugačno sramotenje z uporabo informacijske tehnologije

kričáč -a m (*angl. shoutbox, chatterbox*)
vnosno polje na spletni strani, kjer lahko vsakdo komentira vsebino

médij -a m (*angl. medium*)
1. kar omogoča shranjevanje, predstavitev in prenos podatkov; sin. nosilec podatkov (1), nosilec podatkov (2)
2. način, oblika predstavitve sporočila, npr. besedilo, slika, zvok, video; sin. predstavnost
3. sredstvo javnega obveščanja, npr. novičarsko spletišče, internetna televizija, blog

nadíranje -a s (*angl. flaming*)
gl. žaljenje

nadírčno sporočílo -ega -a s (*angl. flame*)
elektronsko sporočilo z elementi verbalnega nasilja

netikéta -e ž (*angl. netiquette, net etiquette*)
pravila obnašanja na internetu; sin. omrežni bonton

nosílec podátkov -lca -- [u̯c] m (*angl. data medium, data carrier*)
1. priprava ali sredstvo, ki omogoča zapisovanje in/ali shranjevanje podatkov, npr. silicijeva ploščica, magnetni trak; sin. medij (1); prim. pomnilnik (1), pomnilniška naprava
2. naprava z vgrajeno pripravo ali sredstvom za zapisovanje in/ali shranjevanje podatkov, npr. tračna kaseta, USB-ključ; sin. medij (1)

omréžje -a s (*angl. network*)
sistemi, naprave, ki so med seboj povezani zaradi izmenjevanja podatkov, informacij; prim. mreža[1]

omréžni bontón -ega -a m (*angl. netiquette, net etiquette*)
pravila obnašanja na internetu; sin. netiketa, spletni bonton

opljúvanje -a s (*angl. flame war*)
gl. spletni spopad

poslôvna informátika -e -e ž (*angl. management information systems, business informatics*)
področje informatike, ki proučuje poslovne informacijske sisteme

predstávnost -i ž (*angl. medium*)
način, oblika predstavitve sporočila, npr. besedilo, slika, zvok, video; sin. medij (2)

sociálna mréža -e -e ž (*angl. social network*)
gl. družbeno omrežje in socialno omrežje

sociálna povezáva -e -e ž (*angl. social link*)
gl. družbena povezava

sociálni médij -ega -a m (*angl. social media*)
gl. družbeni medij

sociálni zaznámek -ega -mka m (*angl. social bookmark*)
spletni zaznamek, vključen na spletno stran interesne skupine, ki članom omogoča preprostejši dostop do izbranih spletnih vsebin

sociálno mréženje -ega -a s (*angl. social networking, social netting*)
udejstvovanje v socialnem omrežju; sin. družbeno mreženje

sociálno omréžje -ega -a s (*angl. social network*)
skupina ljudi, ki jih povezujejo skupni interesi in aktivnosti; sin. družbeno omrežje, družabno omrežje

sociálno prográmje -ega -a s (*angl. social software*)
programje za izvedbo storitev socialnega omrežja, npr. Facebook; sin. družbeno programje

splétni bontón -ega -a m (*angl. netikette*)
pravila obnašanja na internetu; sin. netiketa, omrežni bonton

splétni spopàd -ega -áda m (*angl. flame war*)
večstransko pošiljanje elektronskih sporočil z elementi verbalnega nasilja; sin. opljuvanje

splétni zaznámek -ega -mka m (*angl. web bookmark*)
beležka s spletnim naslovom, ki omogoča vnovičen obisk spletnega mesta

sporočílo -a s (*angl. message*)
informacija, ki jo pošiljatelj usmeri k naslovniku, oddajnik k sprejemniku na določen način in z določenim namenom

stóritev drúžbenega omréžja -tve -- -- ž (*angl. social networking service*)
storitev (1), ki omogoča komuniciranje v družbenem omrežju; sin. storitev socialnega omrežja

storítev sociálnega omréžja -tve -- -- ž (*angl. social networking service*)
storitev (1), ki omogoča komuniciranje v socialnem omrežju; sin. storitev družbenega omrežja

taksonomíja -e ž (*angl. taxonomy*)
1. nauk o razvrščanju v skupine po določenem kriteriju
2. hierarhična razvrstitev po določenem kriteriju, npr. periodni sistem elementov

vnôsno pólje -ega -a s (*angl. input field*)
polje[1](1) za vpisovanje besedila, podatkov; prim. besedilno polje

zaznámek -mka m (*angl. bookmark*)
označitev izbranega mesta v kaki strukturi

žáljenje -a s (*angl. flaming*)
pošiljanje žaljivih sporočil; sin. nadiranje

žaljívec -vca m (*angl. flamer*)
pošiljatelj žaljivih sporočil

Izbor pripravlja in ureja
Katarina Puc s sodelavci Islovarja

Koledar prireditev

Living bits and things 2012	27. november 2012	Bled, Slovenija	http://www.livingbitsandthings.com/
Practical Seminar How to Prevent Data Leakage in Public Institutions and Organisations	29.–30. november 2012	Berlin, Nemčija	http://www.euroacad.eu
The 3rd Annual European Data Protection and Privacy Conference	4. december 2012	Bruselj, Belgija	www.dataprotection2012.eu
20. konferenca Dnevi slovenske informatike	15.–17. april 2013	Portorož, Slovenija	www.dsi2013.si

Pomembni spletni naslovi

→ IFIP News: **http://www.ifip.org/images/stories/ifip/public/Newsletter/news ali www.ifip.org → Newsletter**

→ IT Star Newsletter: **www.itstar.eu**

→ ECDL: **www.ecdl.com**

→ CEPIS: **www.cepis.com**

Dostop do dveh tujih strokovnih revij

→ Revija **Upgrade** (CEPIS) v angleščini (ISSN 1684-5285) je dostopna na spletnem naslovu: **http://www.upgrade-cepis.org/issues/2008/4/upgrade-vol-IX-4.html**.

→ Revija **Novática** (CEPIS) v španščini (ISSN 0211-2124) je dostopna na spletnem naslovu: **http://www.ati.es/novatica/**.

Včlanite se v Slovensko društvo INFORMATIKA

Pristopna izjava

za članstvo v Slovenskem društvu INFORMATIKA

Pravne osebe izpolnijo samo drugi del razpredelnice

Ime in priimek	
Datum rojstva	
Stopnja izobrazbe	srednja, višja, visoka
Naziv	prof., doc., spec., mag., dr.
Domači naslov	
Poštna št. in kraj	
Ulica in hišna številka	
Telefon (stacionarni/mobilni)	

Zaposlitev člana oz. člana - pravna oseba	
Podjetje, organizacija	
Kontaktna oseba	
Davčna številka	
Poštna št. in kraj	
Ulica in hišna številka**	
Telefon	
Faks	
E-pošta	

Zanimajo me naslednja področja/sekcije*

- ☐ jezik
- ☐ informacijski sistemi
- ☐ operacijske raziskave
- ☐ seniorji
- ☐ zgodovina informatike
- ☐ poslovna informatika
- ☐ poslovne storitve
- ☐ informacijske storitve
- ☐ komunikacije in omrežja
- ☐ softver
- ☐ hardver
- ☐ upravna informatika
- ☐ geoinformatika
- ☐ izobraževanje

podpis

kraj, datum

Pošto društva želim prejemati na domači naslov / v službo.

Članarina znaša: 18,00 € - redna
7,20 € - za dodiplomske študente in seniorje (ob predložitvi dokazila o statusu)
120,00 € - za pravne osebe

Članarino, ki vključuje glasilo društva – revijo **Uporabna informatika**, bom poravnal sam / jo bo poravnal delodajalec.

DDV je vključen v članarino.

Naročilnica na revijo UPORABNA INFORMATIKA

Naročnina znaša: 35,00 € za fizične osebe
85,00 € za pravne osebe – prvi izvod
60,00 € za pravne osebe – vsak naslednji izvod
15,00 € za študente in seniorje (ob predložitvi dokazila o statusu)

DDV je vključen v naročnino.

ime in priimek ali naziv pravne osebe in ime kontaktne osebe

davčna številka, transakcijski račun

naslov plačnika

naslov, na katerega želite prejemati revijo (če je drugačen od naslova plačnika)

telefon/telefaks

elektronska pošta

Podpis

Datum

ISSN 1318-1882
9 771318 188001