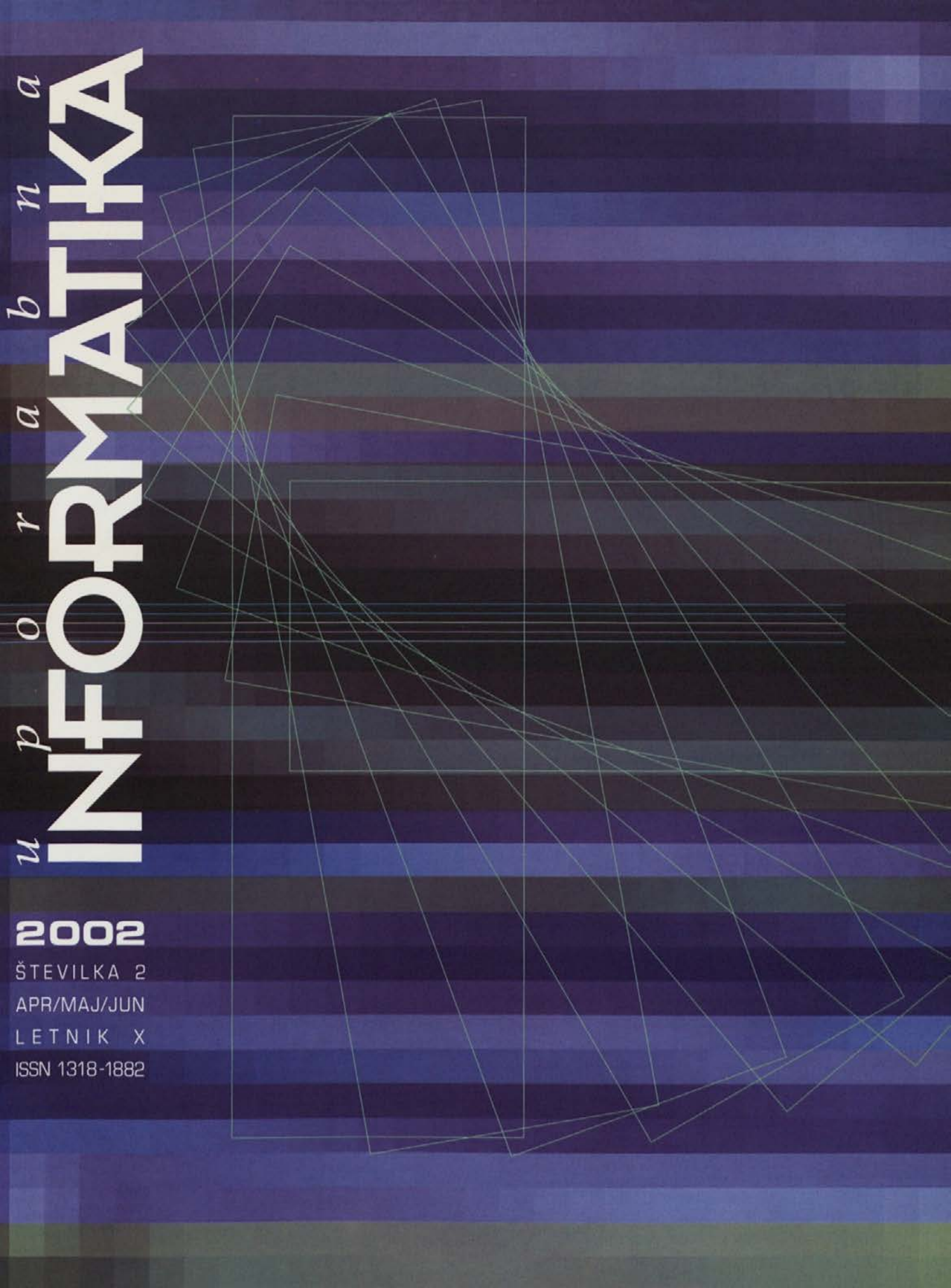
uporabna INFORMATIKA

2002
ŠTEVILKA 2
APR/MAJ/JUN
LETNIK X
ISSN 1318-1882



DONATORJI

ASTER

Nade Ovčakove 1, 1000 Ljubljana
Tel.: +386 01 589 42 00

gambit trade
računalništvo in informatika d.o.o.

Savska c. 3a, 1000 Ljubljana
Tel.: 01 437 63 33

MAOP®
Vaš partner v informatiki
MAOP RAČUNALNIŠKI INŽENIRING D.O.O., WWW.MAOP.SI

MARAND
Napredna računalniška hiša

Cesta v Mestni log 55, 1000 Ljubljana
Tel.: 01 283 33 77

Microsoft®

rrc
RRC Računalniške storitve d.d.
Jadranska 21, Ljubljana
Tel.: 01 / 4778 500, Faks: 01 / 4255 229
www.rrc.si, info@rrc.si

SIQ
ISO 9001
Q-231

SIEMENS
Dunajska 22, 1511 Ljubljana, Slovenija

SMART
COM
d.o.o.
Brnčičeva 45, 1001 Ljubljana, Slovenija
tel: + 386 01 56 11 606

SRC SI
sistemske integracije
Tržaška cesta 116, 1000 Ljubljana
Tel.: 01 242 80 00 • Fax: 01 423 41 73
e-mail: src@src.si • http://www.src.si

433748

vsebina

uporabna INFORMATIKA

Narodna in univerzitetna knjižnica LJUBLJANA

Navodila avtorjem

Revija Uporabna informatika objavlja originalne prispevke domačih in tujih avtorjev na znanstveni, strokovni in informativni ravni. Namenjena je najširši strokovni javnosti, zato je zaželeno, da so tudi znanstveni prispevki napisani čim bolj mogoče poljudno.

Članke objavljamo v slovenskem jeziku, prispevke tujih avtorjev pa tudi v angleškem jeziku. Vsak članek za rubriko Strokovne razprave mora za objavo prejeti dve pozitivni recenziji.

Prispevki naj bodo lektorirani, v uredništvu opravljamo samo korekturo. Po presoji se bomo posvetovali z avtorjem in članek tudi lektorirali.

Polno ime avtorja naj sledi naslovu prispevka. Imenu dodajte naslov organizacije in avtorjev elektronski naslov. Prispevki za rubriko Strokovne razprave naj imajo dolžino cca 30.000 znakov, prispevki za rubrike Rešitve, Poročila, Obvestila itd. pa so lahko krajši.

Članek naj ima v začetku Izvleček v slovenskem jeziku in Abstract v angleškem jeziku. Izvleček naj v 8 do 10 vrsticah opiše vsebino prispevka, dosežene rezultate raziskave.

Abstract naj se začenja s prevodom naslova v angleščino.

Pišite v razmaku ene vrstice, brez posebnih ali poudarjenih črk, za ločilom na koncu stavka napravite samo en prazen prostor, ne uporabljajte zamika pri odstavkih.

Revijo tiskamo v črno beli tehniki s folije, zato barvne slike ali fotografije kot originali niso primerne. Objavljali tudi ne bomo slik zaslonov, razen če so nujno potrebne za razumevanje besedila. Slike, grafikoni, organizacijske sheme itd. naj imajo belo podlago. Po možnosti jih pošiljajte posebej, ne v okviru članka.

Članku dodajte kratek življenjepis avtorja (do 8 vrstic), v katerem poudarite predvsem delovne dosežke.

Z vsa vprašanja se obračajte na tehnično urednico Katarino Puc. Prispevke pošiljajte na disketi in papirju na naslov Katarina Puc, Slovensko društvo informatika, Vožarski pot 12, 1000 Ljubljana, ali samo po elektronski pošti na naslov katarina.puc@drustvo-informatika.si.

Po odločitvi uredniškega odbora, da bo članek objavljen v reviji, bo avtor prejel pogodbo, s katero bo prenesel vse materialne avtorske pravice na društvo INFORMATIKA. Po izidu revije pa bo prejel plačilo avtorskega honorarja po tedaj veljavnem ceniku ali po predlogu glavnega in odgovornega urednika.

Naslov uredništva je:

Slovensko društvo INFORMATIKA, Uredništvo revije Uporabna informatika, Vožarski pot 12, 1000 Ljubljana

www.drustvo-informatika.si/posta

© Slovensko društvo INFORMATIKA, Ljubljana

Spoštovane bralke in bralci,

čeprav smo o e-upravi ne davno izdali posebno številko, se ponovno vračamo k tej temi. Glavni razlog je v tem, da razvoj e-poslovanja v upravi očitno ne napreduje tako, kot si je pred letom in pol zamislila vlada, ko je sprejela Strategijo uvajanja elektronskega poslovanja do leta 2004. Pa ne, da se je uštela samo slovenska vlada, informacije, ki jih dobivamo od drugih držav članic EU, kažejo, da imajo tudi drugod podobne probleme. Čeprav je večina evropskih držav članic EU postavila izjemno napete roke za razvoj e-uprave, večinoma tam nekje do leta 2005, je sedaj že bolj ali manj jasno, da je število problemov, ki jih bo potrebno na tej poti rešiti, preveliko in tako hiter napredek preprosto ne bo mogoč. Prve podrobne študije, ki so bile opravljene v EU koncem lanskega leta v vseh članicah EU in tudi nekaterih kandidatkah, dajejo sedaj že razmeroma jasno sliko, kje smo.

Do sedaj se je pokazalo, da je napredek hiter na področju tako imenovanih informacijskih storitev, saj je večina upravnih institucij že predstavljena na spletu s svojimi spletnimi stranmi, na katerih imajo občani in podjetja na voljo že razmeroma bogato paleto najrazličnejših bolj ali manj uporabnih informacij, ki pridejo prav pri reševanju upravnih zadev. Razvoj te informacijske ponudbe je razmeroma preprost, saj ne zahteva obsežnejših priprav, notranjih reorganizacij in prenove internega poslovanja. Tudi ponudba uradnih dokumentov, ki si jih lahko sami natiskamo, jih izpolnimo in nato posredujemo upravnim organom po klasični poti, je že razmeroma bogata.

Do očitnega zastoja pa prihaja pri razvoju vseh zahtevnejših vrst storitev, kjer gre za dvosmerno komunikacijo med upravo in njenimi uporabniki in je potrebno za izpeljavo postopka po novem, to je po elektronski poti, spremeniti notranje poslovanje. To so tako imenovane transakcijske storitve. Nikjer še niso natančno ugotovili, koliko tovrstnih storitev uprava sploh izvaja. Ocene se sučejo okrog več tisoč oziroma celo tja do deset tisoč različnih storitev, ki jih uprava na vseh mogočih področjih in ravneh opravlja za različne uporabnike. Pri razvoju teh storitev, ki običajno zahtevajo osebno identifikacijo uporabnika, plačilo, neposredni dostop do cele vrste javnih registrov in podatkovnih zbirk, pa smo še povsem na začetku.

Tehnologija je še relativno majhen problem, saj ustrezne rešitve obstajajo. Večina problemov tiči v organizacijski in normativni sferi. Notranje poslovanje organov bo potrebno v ta namen v celoti prenoviti. Upravni postopki in poslovni procesi, ki so se desetletja razvijali, ob predpostavki, da je papir osrednji delovni komunikacijski in arhivski medij, preprosto ne ustrezajo več. Njihovo spreminjanje pa je dolgotrajno, saj zahteva ogromno dela, pa tudi spreminjanje obstoječe zakonodaje in delovnih navad, organizacijskih rešitev, delitev pristojnosti med organi, prilagajanje organizacijskih struktur in še bi lahko naštevali. Zahteva nujno sodelovanje med organi, ki ga v preteklosti pogosto ni bilo, čeprav bi bilo zelo koristno.

Kako to situacijo razrešiti, še ni povsem jasno. Dejstvo je, da je elektronska storitev integracija tehnologije z vsebino. Glavni problem je v tem, da tisti, ki so pristojni za vsebine v upravi, tega še ne razumejo v celoti in mislijo, da je e-uprava problem računalničarjev. To se najbolj očitno pokaže, če si ogledamo slovenski državni portal. Portal je prav lepo oblikovan in zasnovan, vendar se vsebinsko v enem letu praktično ni spremenil, razen informacijskih storitev z njegovo pomočjo niti spremembe naslova ne morete opraviti. Očitno je, da bi vlada morala čimprej nekaj storiti, da se stvari premaknejo z mrtve točke. To nekaj pa je dobro premišljen in kar se da konkreten in realen akcijski načrt, ki bo spravil skupaj glavne akterje in s konkretnimi projekti začel serijsko bruhati nove e-storitve,

Glavni in odgovorni urednik
Mirko Vintar

Povabilo k pripravi prispevkov za tematsko številko revije Uporabna informatika

Tema: Celovite programske rešitve – ERP

Vsebinska izhodišča:

Celovite uporabniške programske rešitve (ERP - Enterprise Resource Planning) lahko opredelimo kot celovito povezan in na poslovnem modelu organizacije temelječ sistem, ki ob uporabi sodobne informacijske tehnologije zagotavlja vsem poslovnim procesom organizacije in z njo povezanim poslovnim partnerjem optimalne možnosti načrtovanja, razporejanja virov in ustvarjanja dodane vrednosti. Uvajanje celovitih rešitev temelji na konceptu prenove poslovanja, ki temelji na prenosu najboljše prakse, zajete v teh rešitvah, v posamezno organizacijo in njeno neposredno okolje.

Za tematsko številko Uporabne informatikeso zlasti pomembni prispevki naslednjih tematskih sklopov:

- ERP in elektronsko poslovanje;
- ERP in prenova poslovanja in poslovnih procesov;
- razvoj ali nakup sistemov ERP;
- stanje (predstavitev) in razvoj na področju domačih rešitev ERP;
- različna vprašanja izbire, nabave in uvajanja sistemov ERP in najboljše prakse (npr.: metodološka, tehnološka, ekonomska, pravna, organizacijska);
- priprava, vodenje in zagotavljanje uspešnosti projektov ERP, predstavitev uspešnih projektov.

Vabimo avtorje, da nam pošljejo še neobjavljene prispevke z vsebino v okviru navedenih izhodišč.

Rok za oddajo prispevkov: 1. september 2002

Recenzija in potrditev: 15. oktober 2002

Predvidena objava: december 2002

Prispevki naj bodo pripravljeni skladno z navodili avtorjem, v slovenščini z angleškim izvlečkom ali v angleščini (tuji avtorji).

Gostujoči urednik in kontaktna oseba:
dr. Andrej Kovačič, andrej.kovacic@uni-lj.si

DEVELOPMENT OF APPLICATION FOR REMOTE DATABASE ACCESS USING JAVA SECURITY FUNCTIONS

Jasmin Malkić, Tatjana Welzer, Boštjan Brumen

Abstract

Business use of Internet represented by on-line banking, shopping and other commercial activities, introduced a few years ago, has made the development of a solid and secure data transfer through the public network largely necessary. The resu lt of the development made in this direction, was a wide spectrum of solutions that provide different levels of security. By rule, all those projects that could tackle high security standards that e-commerce requires, applied cryptographic engines. The paper analyses the history of Internet security in brief, and pays a special attention to the cryptographic properties of the programming language Java, now and in the future. By setting an example of the client-server application for remote authorized access to a database source, it demonstrates the use of Java's basic cryptographic tools.

Izvleček

Poslovna uporaba interneta v spletnem bančništvu, trgovini in drugih poslovnih dejavnostih, ki je bila vpeljana v zadnjih nekaj letih, zahteva razvoj varnega prenosa podatkov preko javnega omrežja. Rezultat programskega razvoja v tej smeri je širok spekter rešitev, ki ponujajo različne stopnje varnosti. Kriptografske algoritme pravilo-ma uporabljajo projekti, ki izpolnjujejo visoke varnostne standarde za varen prenos podatkov. Članek kratko analizira zgodovino varnosti na internetu, s poudarkom na kriptografskih lastnostih programskega jezika java, sedaj in v prihodnosti. Podan je tudi primer aplikacije odjemalec-strežnik za avtorizirani dostop do oddaljenega podatkovnega vira, ki prikazuje uporabo osnovnih javanskih kriptografskih orodij.

■■■

0. Introduction

Since its very beginning, Internet has been dealing with two, at first sight totally opposite tasks - providing public information and hiding its secret content from unauthorized access. Although data transfer for both, public and secret content can use the same lower communication layers, it's obvious that the transfer of secret information requires some extra work in order to keep it secret.

There are many different possible scenarios for data transfer over the Internet when we need to protect data, and they require the implementation of different security levels. Although the universal security solution isn't given, good solutions for different situations do exist. It's hard to unify the Internet security approaches, mainly because of the very nature of Internet itself. Its basic philosophy - to be open to many computers which work on different operating systems and use different Internet clients, implicates a lot of different security approaches. What is also important is the fact that Internet operates in many countries, and security laws can vary. This leads to the conclusion that every Internet security project should be made in accordance with the given local conditions.

1. The Origins of Internet Security

Before HTTP (Hypertext Transfer Protocol, defined in RFC 2616) became the number one format for Internet data, the public need for security didn't really exist. Internet was mainly the occupation and privilege of the academic and military structures; security attacks were possible but rare and much easier to track. Upon emergence of the World Wide Web, which uses HTTP data to present its content, Internet quickly entered the everyday life of many people who could now use it to read the latest news, for personal presentation, and even to do their shopping or bank transactions. It became clear that protecting private data that begun to circulate through the wires was a must.

Web servers and HTTP were not designed to tackle serious security tasks. Although basic authentication is supported by many Web servers, (e.g. *Apache Web Server on UNIX*) where access to a certain directory can be secured by a password stored on the server, the problem arises because nothing protects the password itself while it travels through the Internet in HTTP authentication request. Anybody who cares enough can intercept such a request and read the plain text password in it.

As soon as HTTP became the Internet mainstream, several groups started to work on the methods for securing its transfers. The official project, sponsored by the IETF was named Secure HTTP (*Farrow, 2001*), but also some other works found their way to users. One of those created by Netscape, thanks to the large number of Netscape Navigator users, managed to exceed its initial purpose – to improve Navigator's security. Today, this cryptographic security solution named SSL has its free implementation (OpenSSL) for multiple operating systems, and a large developer community, but still it is not the Internet standard. Which is partly because some serious problems arise with the use of this security method.

The first factor that proved that SSL was far from being perfect, was growing hardware ability to brute force the cryptographic keys. Brute forcing method guesses all the possible combinations of the key, and with 40-bit keys that were used by most Netscape and MS Internet Explorer clients, can be guessed within less than one day with the latest processors involved (*Farrow, 2001*). To avoid this, the use of most recent operational systems and Internet browsers which introduce 128-bit encryption keys is largely recommended. Still, SSL has also some serious problems connected with the fact that its reliability highly depends on the environment, first of all, the client browser being used. Such problems usually result in CERT Advisories or security patches for commonly used Web browsers.

In May 2000, *Kevin Fu* from MIT discovered that Netscape Navigator could be fooled to accept an invalid certificate, or in other words, the right certificate from a false location (*Farrow, 2001*). If a user visited a site that had a certificate where the server name didn't match the common name found within the certificate (usually the case of using the stolen certificate), Navigator failed to detect this and let the user pass secret data to suspicious server. *Mitja Kolšek* from ACROS, Slovenia, discovered another problem with Navigator. Kolšek has discovered that Navigator assumed that subsequent access to the same IP address was part of the same SSL session. Combined with possible DNS spoofing attack between subsequent sessions, this could lead into Navigator visiting a different site (due to failure in checking if server common name matches the one within the certificate), and leaving data on a wrong server. CERT Advisory CA-2000-5 covered this problem. This showed that in spite of the existing certificate verifying system that SSL provides, client's security holes can produce some serious doubt on behalf of a user, to whom a secret content is transferred.

Microsoft's Internet Explorer has also had problems with the implementation of SSL, in versions 4 and 5 the password, part of the basic HTTPS authentication scheme, once sent to server was revealed (sent without encryption) later if a client made a HTTP connection (without SSL) to the same server.

This has been solved by issuing the Microsoft's security advisory and the official patch addressing this problem. Those are only some of the problems that arise from SSL employing the common Internet clients (Netscape Navigator or Internet Explorer) as an important part of a secure connection. However, this appears to be the price SSL pays for its portability, and many clients and operating systems supporting it.

2. The Use of Security Hash Algorithm (SHA-1)

As a US Federal Standard in the category of Computer Security, SHA-1 was established by National Institute of Standards and Technology in April 1995 (*Burrows,1995*). Since this body encourages all private and commercial organizations to use it, SHA is widely used and implemented, whenever any kind of software security is needed, also in many tools other then Java or SSL described here. Besides Internet, SHA-1 also finds its place in data storage, software distribution, and other applications that require data integrity assurance and data origin authentication.

Input for SHA-1 is a binary message with less than 2^{64} bits, out of which algorithm computations produce a 160-bit output called *message digest*. SHA-1 is an improved technical revision of SHA, as described in *Federal Information Processing Standards Publication (FIPS) 180*. SHA-1 is secure due to its properties: it is computationally infeasible to find a message, which corresponds to a given message digest, or to find two different messages which produce the same message digest.

SHA-1 message digest can also be used as an input to the *Digital Signature Algorithm* (DSA) which generates or verifies the signature for the message. If the message integrity is more important than its secrecy, it pays off to digitally sign the message digest rather than the message itself, because the message digest is in most cases much shorter.

SHA-1 "engine" is based on the principles similar to those used by *Prof. Ronald L. Rivest* from MIT when designing the MD4 message digest algorithm (*Springer-Verlag, 1991*). It sequentially accepts packets of 512 bits, which means that a start message has to be padded to contain 512 x n bits. In order to achieve this, a "1" followed by the m "0"s are appended to the end of the original message. At the very end of a padded message, a 64-bit integer which expresses the length of the original message (in bits) is also appended, which concludes padding process (*Burrows,1995*).

3. Java Security Features

The security considerations in Java Cryptography Architecture mainly point at secure authentication and data transfer (*Sun, 2001*). As Java is, after all, an object language designed preferably for network and Internet programming, it's therefore essential to provide data integrity on the way from one machine to another. In order to achieve this, there are three main cryptographic concepts which Java uses: *Message Digests*, *Digital Signatures* and *Certificates*. They assume different levels of security, and their use depends on the security objectives.

Message Digest

The basic security tool, not only in Java, is the *message digest*, and almost all Java security concepts use it in some form. When a situation allows, pure digests of the secret content can be instantiated and used for authentication purposes. The class which is responsible for this is *java.security.MessageDigest*, where all the necessary methods for this process are contained. Cryptography core is provided by Java implementation of ***SHA-1*** or ***MD5*** algorithm, between which a user chooses when creating an instance of MessageDigest class/object:

```
MessageDigest md= MessageDigest.getInstance(String algorithm);
```

The *MessageDigest* object has the ability to be filled with the array of bytes using the method *update(**byte**[ ] data_to_diggest)*, and to digest its content using the method *digest( )*, with no parameters. If chosen algorithm for *MessageDigest* object is SHA-1, loaded array of bytes will be the subject of cryptographic procedure producing a 160-bit output from the *digest( )* method. This can make a good use for the secure transfer of passwords.

If a remote client needs a password to be authenticated to the server, it can digest it for secure transfer through non-trusted network, then transfer it to server which from its side can also digest its own copy of password and compare it with those received from the client for authentication.

Some downs of the message digesting are decreasing the public confidence in MD5, and the possibility of a third-party catching the digested password during the transfer and interpret it later to the server in order to access its secured resources. To avoid this, the improvements can be made by computing the digest together with timestamp and random numbers.

This approach focuses on a situation when the authentication of a client to the server is of greatest importance, and server's identity is unquestioned. As it's also the most obvious way for the demonstration of Java's implementation of SHA-1 algorithm, it has been chosen as an example in the next chapter.

Digital Signatures

A signature is the message digest encrypted with the signer's private key. In Java, signatures are provided by methods defined in *java.security.Signature* class. Factory methods for *Signature* object are *getInstance* with two different constructors:

```
Signature p = Signature.getInstance (String algorithm); and
Signature p = Signature.getInstance (String algorithm, String provider);
```

The second method returns a *Signature* for the given algorithm-provider pair, while the first does the same but with the first provider that supports the given algorithm. A signature uses two algorithms – one to calculate a message digest and one to encrypt the message digest with the issuer public key. The SUN provider shipped with JDK1.1 and further versions supports ***DSA*** encryption of a ***SHA-1*** message digest (*Oaks, 1998*). This is simply referred to by giving a DSA string value to the first *Signature* constructor. Data to *Signature* object are fed on the same way as to *MessageDigest* object, using the *update()* method, however Signature first has to be initiated with the signer's private key by method *initSign(**PrivateKey** myPrivateKey)*. On the server side, received *Signature* object is verified by *initVerify(**PublicKey** myPyblicKey)* method, which returns *boolean* variable to indicate if signature is verified or not.

In the *Signature* class, Java introduces private and public key for authentication, where the private key of a signer is used for making the signature, and public for verifying it on remote side. Signatures still don't provide full confidentiality because of the public key transfer. If a client would always carry a signer's public key e.g. on a floppy disk, and could use it when necessary this method would be much more secure. On the other hand this would be a bit unpractical for use on the Internet. Certificates do solve this problem (*Kundsen, 1998*).

Certificates

To verify a signature, the public key file is needed, but the problem is how to distribute the keys securely over the network. Even if the key can be downloaded, how can a user be sure that the received key is really issued from the server that it refers to. The best solution to these problems, which still maintain all the benefits of Internet, is given in the form of the Certificates.

A certificate is a statement, signed by one person, that the public key of another person has a particular

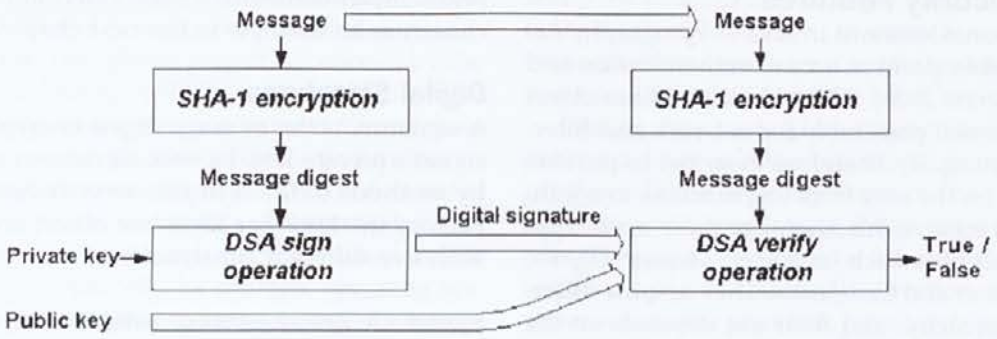


Figure 3.1. **–Common Digital Signature process with use of SHA-1 and DSA**

value. A person who guarantees the validity of the other certificates has to be trustworthy, and is called a *Certificate Authority (CA)*.

Java API from version 1.2. is equipped with necessary methods to recognize the certificates issued from CAs that use ***X.509v3*** certificates. Most of the methods for distributing the certificates are packed into *java.securtiy.cert.Certificate* class.

The certificate object contains information about the issuing identity, its public key, and its signature of the above information. All that information can be recalled using the methods defined in that class, such as *getPublicKey( )*. Support for ***X.509*** certificates is stored in the additional class. With *getInstance( )* method in *java.security.cert.X509Certificate* class, such certificate can be loaded into Java program. The validation on the server side, is implemented through the standard method *validate( )*.

By using the certificates, an additional trusted authority is provided to assure the purity of signature. This feature is expected to develop even more in future versions of Java (JSDK 1.4.1 and further), in order to assure full support for this way of securing the communications over public network.

4. SHA-1 Encryption Within Java – An Example

A simple client-server architecture, designed for the remote access to database source could be an example of Java security basics. Regarding the building of such architecture itself, Java offers two generic classes - *Applet* for client and *HttpServlet* for server side.

Java applets depend on Java Virtual Machine (JVM, shipped with almost every Internet browser available) that has to be implemented on the client side. JVM's runtime security properties let the downloaded *Applet* classes run inside it, while preventing any harm to the local system (*Oaks, 1998*).

The server side needs a web server to serve the applet class to remote clients, and on the same machine an environment for the central application should reside. Such application has to be able to manipulate with the HTTP requests - responses, and to communicate with the database. While every Java application which imports *java.sql* package and proper JDBC database drivers can do database communication, HTTP handling is reserved for the servlets – Java's server-side components in many ways analogous to CGI applications (*Fields, 1999; Darby, 1998*).

Figure 4.1. illustrates the password verification authentication concept. The implementation of the client and server Java classes will be explained here in brief, while the full source code can be seen on http://www.inet.ba/malkic/ird1.

Client Side

The important GUI objects in client class (*class CappletTs extends Applet*) are: *messageField* and *passwordField* (of type *TextField*) that contain the parameter for database query and authorization password respectively; and *responseField* (of type *TextArea*) that receives a response from the server – selected database content or error message. When the GUI event occurs (password and database query sent) method *private void interactWithServlet ( )* is called. To establish a HTTP connection with Java servlet application that resides on the server, an URL object must be created within this method. *servletURL* of type *URL* and *servletConnection* of type *URLConnection* encapsulate the server's address and communication port, while *servletConnection* also handles two data streams for output and input (*out* of type *DataOutputStream* and *in* of type *InputStream*).

The cryptography object in this applet is *md* of type *MessageDigest*. Its constructor receives "SHA" string parameter which means this object will encode its content using SHA-1 algorithm. For better security of the

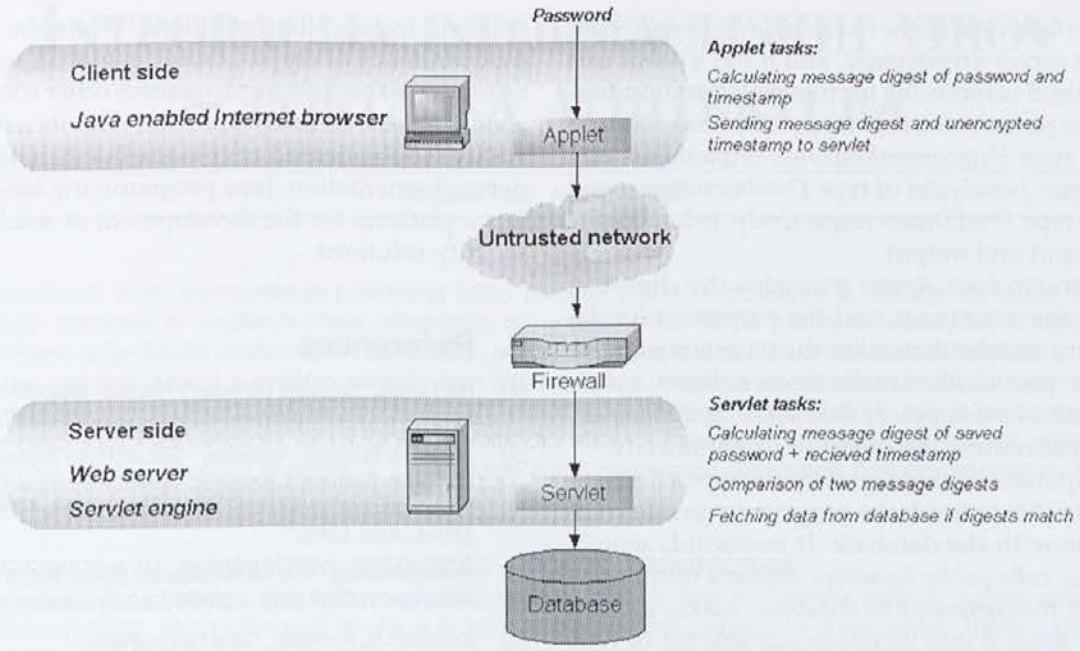


Figure 4.1. – **Simplified architecture of authorized access to remote database source**

transferred data, the password represented by *String passwd* is encrypted together with *timeStamp* of type *long*. This variable is a numerical interpretation of the current value of the local *Date* object. *md*'s *digest()* method output is *byte* array *protect* that is sent to server through *out* together with unencrypted *timeStamp*. Enclosed in *out* are also parameter for database query (from *messageField*), and *int* value *length* which indicates number of *protect* array members. When SHA-1 is used *length* isn't so crucial because it's by SHA-1 definition always 20 (length of SHA-1 output is 160 bits or 20 bytes), but this leaves space for the use of other algorithms too.

Method *public byte[ ] toBytes (long lval)*, has an internal purpose, and serves as a converter from type *long* to *byte[ ]*. *MessageDigest* object can't be updated with *long* type directly, so in this case *timeStamp* variable of type *long*, has to be converted first (server side uses the same method for this conversion). *Figure 4.2.* shows both encrypted and non-encrypted data that applet is sending through the public network.

Server Side

Web server installed on the server machine serves the applet class to remote clients. Together with Java servlet runtime components it also enables working conditions for the servlet class that resides on the server. A suitable combination of these components could be e.g. SUN's JSDK 2.1 with Microsoft's IIS. An important segment of the server is the database – it can be an ODBC data source, but also any other database that has JDBC drivers implemented (besides ODBC standard JDK contains the drivers for Oracle database, and a free driver classes package for MySQL is also available on *http://mysql.sourceforge.net*).

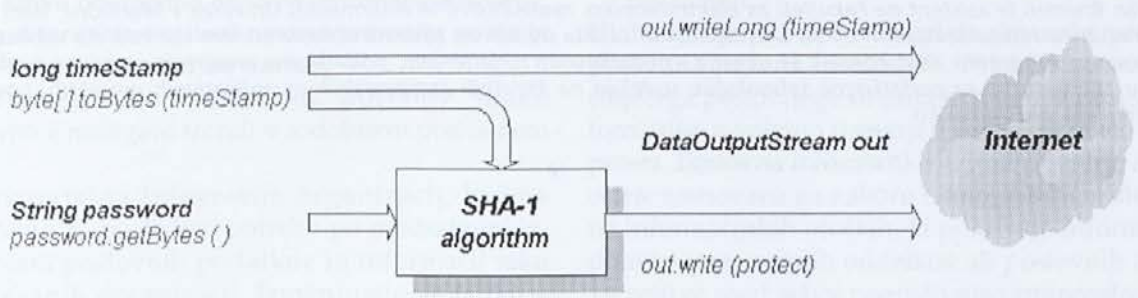


Figure 4.2. – **Secure password transfer from the client side**

Servlet class (*class DBaseTh extends HttpServlet*) concludes this server architecture, and it has a *public void doPost* method responsible for the HTTP communication. Its two parameters, *req* of type *HttpServletRequest*, and *res* of type *HttpServletResponse*, implements two data streams - *fromApplet* of type *DataInputStream* and *toApplet* of type *PrintStream* respectively, to handle the servlet's input and output.

Data stream *from Applet* provides the digest of password, the timestamp, and the parameter for database query. Servlet then takes the timestamp and its copy of the password to make its own digest, by using the method *public byte [] doSHAEncrypt(String sec, long lval)* that returns the digest of its parameters.

By comparison of two digest byte arrays a decision is made whether to finish the program or go into communication with the database. If passwords match, servlet class calls *public ResultSet doQuery (String query)* method that connects to database, makes a given query and feeds it into its return variable *rs1* of type *ResultSet*. The data from *rs1* are extracted by using its method *next( )*, and sent through *out* back to the client.

5. Conclusion

The need for the Internet security in the world of e-commerce grows together with the Internet itself. As in the Internet terms growth also means a variety of software involved, both on client and server side, it seems that secure data transfer is a good basis for the platform independent solutions. While the platform independency of cryptographic engine in such applications must be fully implemented, other components should at least be easily portable. With its native portability, sophisticated cryptographic functions, and network orientation, Java programming language offers a platform for the development of solid Internet security solutions.

6. References

[1] Rik Farrow, *"Network Defense"*, Network Magazine Vol.16, January 2001, http://www.networkmagazine.com.

[2] James H. Burrows - redactor, *"Secure Hash Standard"*, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg USA, FIPS PUB 180-1, April 1995.

[3] Springer-Verlag, *"The MD4 Message Digest Algorithm"*, Advances in Cryptology – CRYPTO 1990 Proceedings, 1991.

[4] Jonathan B. Kundsen, *"Java Cryptography"*, O'Reilly Books May 1998.

[5] Scott Oaks, *"Java Security"*, O'Reilly Books May 1998.

[6] Chad Darby, *"Applet and Servlet Communication"*, Java Developer's Journal, September 1998.

[7] Sun Microsystems, *"Secure Computing With Java: Now and the Future"*, SUN Microsystems White Papers 2001, http://java.sun.com/marketing/collateral/security.htm.

[8] Dyane K. Fields, "Applet-to-Servlet Communication for Enterprise Applications", 1999, http://developer.netscape.com/viewsource.

◆

Jasmin Malkić je diplomiral na Fakulteti za elektrotehniko in računalništvo Univerze v Zagrebu (Hrvaška) in je študent podiplomskega študija računalništva in informatike na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Sodeloval je pri projektih v javi in C++ pri razvoju "GSM Billing and Customer Care" sistema v podjetuju ZIRA Ltd., Sarajevo (Bosna in Hercegovina). Zadnje dve leti se ukvarja z varnostjo na internetu.

◆

Dr. Tatjana Welzer je izredna profesorica na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Predava predmete Podatkovne baze I in II ter Dostopnost in zaščita podatkov, na univerzitetnem in na visokošolskem strokovnem programu, ter predmet Zaščita v raunalniških okoljih na podiplomskem programu. Raziskovalno se ukvarja z načrtovanjem podatkovnih baz, ponovno uporabo, kakovostjo podatkov in varnostjo računalniških sistemov. Kot vodja Laboratorija za podatkovne tehnologije vodi ali sodeluje pri mnogih domačih in mednarodnih projektih, povezanih s problematiko podatkovnih tehnologij.

◆

Boštjan Brumen je asistent na Fakulteti za elektrotehniko, raunalništvo in informatiko Univerze v Mariboru. Vodi vaje pri predmetih Podatkovne baze I in II ter Dostopnost in zaščita podatkov, tako na univerzitetnem kot tudi na visokošolskem strokovnem programu. Raziskovalno se ukvarja s podatkovnim rudarjenjem, podatkovno analizo in varnostjo podatkov. V okviru Laboratorija za podatkovne tehnologije sodeluje na številnih raziskovalnih in aplikativnih projektih, povezani s podatkovno problematiko.

◆

INTEGRACIJA INFORMACIJSKIH VIROV – PORTALNA ATHITEKTURA

Miroslav Ribič, Andrej Kovačič

Izvleček

V sodobnem svetu elektronskega poslovanja lahko podjetje preživi in prosperira le, če zna učinkovito izrabiti informacije. Potrebno je zagotoviti prave informacije ob pravem času. Tu pa nastopijo težave, saj podjetja ponavadi vzdržujejo večje število medsebojno nepovezanih podatkovnih baz in uporabljajo več poslovnih aplikacij, ki izdelujejo stotine različnih poročil v različnih oblikah, tem informacijskim virom pa je potrebno dodati še razne dokumente, pripravljene v urejevalnikih besedil, preglednicah, elektronsko pošto in druge nestrukturirane informacije. Ta problem rešujejo portalni strežniki, ki omogočajo integracijo različnih informacij, tako strukturiranih kot nestrukturiranih, v enotno spletno stran, to je portal. V nadaljevanju bomo podrobneje obravnavali portalno arhitekturo ter sodobne načine pridobivanja informacij.

Abstract

INTEGRATION OF INFORMATION RESOURCES – PORTAL ARCHITECTURE

The information challenge that many organizations face today is how to organize data to support smart and fast decision-making, which ultimately results in a more responsive and competitive organization. There is plenty of data available, and dozens of ways to collect it: enterprise resource planning (ERP) tools collect data at every possible point along the information chain and generate reports daily, data warehouses and analytic tools slice and dice data in hundreds of ways, not to mention information collected from wireless devices and the Internet. Organizations now face the burden of data overload - and how to turn this data into meaningful information. This problem can be solved by portal server, which is able to render and aggregate information into complex web pages – portals - to provide information to users in a compact and easily consumable form. In this article we present the portal architecture, which can make integration of information resources in such way.

■ ■ ■

1. Potreba po integraciji informacijskih virov

Današnji poslovni svet je vseskozi v stiski s časom. Z vseh strani nas preplavljajo potrebni in nepotrebni podatki. Vendar med vso to maso zelo težko poiščemo za nas dejansko potrebne in koristne informacije. V podjetjih, pa tudi drugod, smo soočeni z veliko količino virov informacij in orodij za dostop do informacij. Zaposleni, predvsem ključni odločevalci, vse težje najdejo čas, da te vire odkrijejo in se naučijo učinkovito uporabljati množico novih orodij. In tukaj se nam kot rešitev na dlani kar sami od sebe ponujajo poslovni portali. Ponujajo nam prilagodljiv, varen, enostaven ter enoten vmesnik za dostop do vseh potrebnih informacij znotraj pa tudi izven podjetja. Uporabniki lahko dostopimo do vseh informacijskih virov, tako strukturiranih kot tudi nestrukturiranih, ne da bi poznali njihovo lokacijo ali format.

Implementacijo poslovnih portalov lahko povežemo z mnogimi trendi v sodobnem poslovnem svetu:

- Horizontalna integracija organizacij, ki ima običajno za posledico potrebo po medsebojni izmenjavi poslovnih podatkov in informacij tako povezanih organizacij. Izmenjujejo se lahko v strukturirani ali nestrukturirani obliki, kar pa postane z uporabo poslovnega portala nepomembno.
- Prenasičenje s podatki zahteva osredotočenje na ustrezne informacije. Tudi to lahko ponudi portal s tako imenovano poosebitvijo, to je z dostavljanjem pravih informacij ob pravem času uporabnikom, ki te informacije potrebujejo oziroma zahtevajo.
- Trend zniževanja stroškov ob hkratnem povečevanju kakovosti proizvodov in storitev in skrajševanju časa oz. poslovnega cikla je ustvaril potrebo po enostavnih, fleksibilnih pa vendar zmogljivih računalniških aplikacijah. To zahteva razvoj računalniških rešitev, kreiranih iz enostavnih sestavnih delov, ki jih je mogoče hitro in sprotno prilagajati.

Globalizacija je z uporabo spleta dejansko prevetrila poslovna pravila in modele. Pri tem prenova obstoječega poslovnega sistema in njegove poslovne informatike v spletno usmerjen sistem pogosto ni preprosta. Poslovna informatika je tradicionalno najpogosteje zasnovana na naboru raznovrstnih rešitev, to je na informacijskih otočkih, ki pokrivajo informacijske potrebe posameznih oddelkov ali poslovnih funkcij. Te rešitve med seboj pogosto niso preprosto združljive. Zato je vzpostavitev neposrednega sodelovanja med heterogenimi informacijskimi sistemi ponavadi

trd oreh. V zadnjih letih se je jezik XML izkazal kot preprosto in učinkovito orodje za reševanje tovrstnih težav, saj omogoča učinkovito upravljanje s podatki.

Upravljanje s podatki na globalnem nivoju pa ni dovolj. Potrebna je tudi njihova učinkovita predstavitev. Portal pri tem igra vlogo povezovalca in združevalca vsebin različnih virov. Pri tem uporablja simbolni jezik XSL, s katerim zbrane podatke XML pretvori v uporabniku prijazen spletni prikaz.

Poslovni portal lahko torej ponudi informacije, ki jih ponujajo internet, intraneti, ekstraneti, podatkovna skladišča in baze ter razni poslovni »upravljavci« dokumentov. Torej omogoča takojšen, neposreden dostop do odločujočih informacij in dokumentov, hkrati pa je poosebljen in zagotavlja ne le iskanje, temveč tudi strukturirano vodenje do informacij. Zagotavlja tudi zelo izpopolnjena orodja za iskanje, filtriranje in analiziranje podatkov in informacij. Portali se lahko uporabljajo tudi kot podpora elektronskemu trgovanju, izboljšanemu upravljanju z znanjem, itd.

Namen portalov lahko torej strnemo v enostavno iskanje in dostop do poslovnih informacij, ki jih potrebujemo za učinkovito opravljanje svojega dela. Informacijske vire nam pomagajo urediti v smiselno celoto, omogočajo boljšo lastno organiziranost in možnost povezovanja s sodelavci. Zaradi svoje povezovalne narave nam omogočajo tudi integracijo poslovnih procesov.

2. Portalna arhitektura

Portali so torej enotna vstopna točka v internet, kjer uporabniki dobimo dostop do informacij. Različne vrste informacij in aplikacij zahtevajo različne selekcijske mehanizme in različno prikazovanje, vse pa mora temeljiti na portalni arhitekturi in podatkih o uporabniških profilih. Zato za uvajanje portalov uporabljamo komponentni model, ki dopušča nadgradnjo portalov s posebnimi vstavljivimi komponentami, ki jim pravimo portleti. Portleti so deli programske logike, ki se ponavadi izvajajo na portalnem strežniku, kjer iz podatkovnih virov pridobivajo informacije in jih pretvorijo v spletno obliko, primerno za prikaz na portalu.

Slika 1 prikazuje poenostavljeno portalno arhitekturo. Razdelili smo jo na štiri osnovne logične komponente in sicer na:

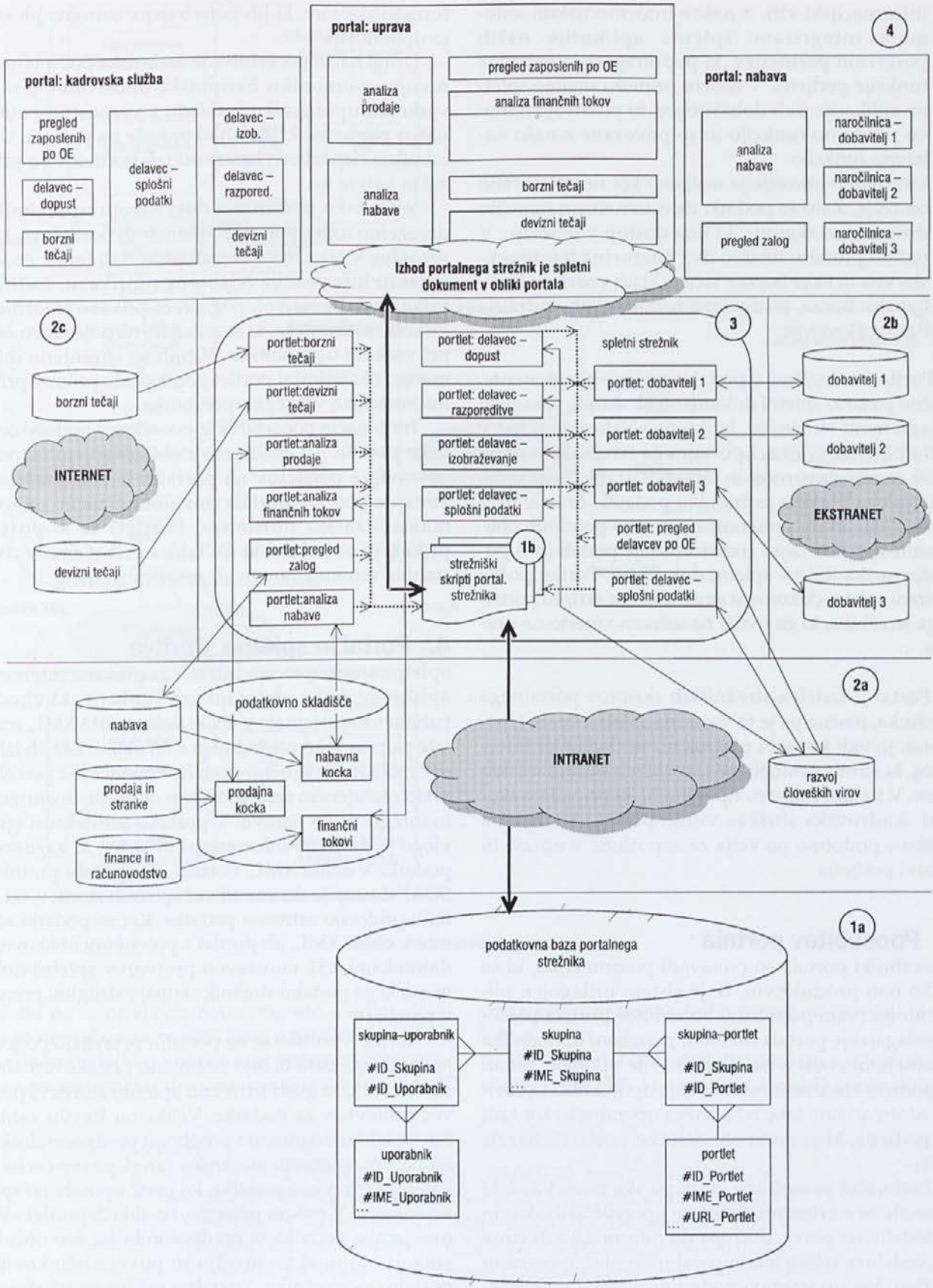
1. portalni strežnik,
2. informacijske vire,
3. portlete in
4. portale.

1. **Portalni strežnik** nam predstavlja generator, s katerim na podlagi vhodnih parametrov kreiramo portale. Fizično je sestavljen iz dveh delov, to je iz podatkovne baze portalnega strežnika, ki se nahaja na podatkovnem strežniku, in strežniških skriptov, ki se nahajajo na spletnem strežniku.

a) V **podatkovni bazi portalnega strežnika** shranjujemo metapodatke, ki jih strežniški skripti potrebujejo, da kreirajo portal. V njej se vzdržujejo trije elementarni šifranti in sicer šifrant uporabnikov, šifrant uporabniških skupin in šifrant portletov. Elementarni šifranti so med seboj povezani prek dveh intersekcijskih tabel in sicer »skupina – uporabnik« in »skupina – portlet«. V tabeli »skupina – uporabnik« določimo z relacijami med šifrantoma uporabnikov in skupin uporabnikov, kateri uporabniki pripadajo določeni skupini. Tako si močno poenostavimo skrbništvo pravic, saj pravice dostopa do informacijskih virov ali portletov določimo skupini uporabnikov, njeni uporabniki pa te pravice podedujejo. Pravice dostopa posamezne skupine do portletov opredelimo v intersekcijski tabeli »skupina – portlet«.
b) Naloga **strežniških skriptov portalnega strežnika** je, da na podlagi zahteve znanega uporabnika (potrebna je predhodna avtorizacija uporabnika) iz podatkovne baze portalnega strežnika na osnovi predhodno opisanih relacij pridobijo podatke o portletih, do katerih ima uporabnik dostop. Predvsem so pomembni podatki o lokacijah portletov (naslov URL) in podatki o poosebljeni podobi portala. Na podlagi pridobljenih podatkov lahko strežniški skripti dosežejo portlete, ki vrnejo dele spletnih vsebin. Tako pridobljene dele spletnih vsebin je potrebno v skladu s prikrojeno podobo uporabnika le še sestaviti v en spletni dokument, to je portal, in ga poslati odjemalcu – brskalniku.

2. Portalni strežnik lahko preko svojih podaljškov, to so portleti, dostopa do poljubnega števila **informacijskih virov**, edina omejitev je, da se podatki pretakajo po protokolu TCP/IP oziroma HTTP. To informacijske vire razporedi na intranetne, ekstranetne in internetne. Dostop do njih poteka povsod na enak način, razlika med njimi je le v stopnji zavarovanosti omrežja.

a) Intranetno omrežje je zavarovano internetno omrežje v okviru enega podjetja, torej so intranetni informacijski viri spletne aplikacije, ki podpirajo delo posameznih funkcij podjetja. V našem primeru, na sliki 1 (območje 2a), imamo štiri spletne aplikacije, in sicer »razvoj človeških virov«, nabava«, »prodaja in stranke« ter »finance in računovodstvo«, ki polnijo podatkovno skladišče oziroma prodajno, nabavno in finančno kocko.
b) Ekstranetno omrežje je zavarovano internetno omrežje v okviru skupine podjetij, torej so ekstranetni



Slika 1: **Poenostavljena portalna arhitektura**

informacijski viri, z našim informacijskim sistemom integrirane spletne aplikacije naših poslovnih partnerjev, ki podpirajo delo nabavne funkcije podjetja. V našem primeru imamo spletne aplikacije treh dobaviteljev, ki pokrivajo njihovo prodajno funkcijo in so povezane z našo nabavno funkcijo.

c) Internetno omrežje je mišljeno kot nezavarovano omrežje. Zato so podatki na internetnem omrežju dostopni vsakomur, ki ima dostop v omrežje. V našem primeru imamo dva internetna informacijska vira in sicer borzne tečaje, ki jih vzdržuje Ljubljanska Borza, in devizne tečaje, ki jih vzdržuje Banka Slovenije.

3. **Portleti** so logične, v portal vstavljive komponente, fizično pa so to spletni dokumenti ali skripti, shranjeni na spletnem strežniku, lahko pa so shranjeni kar v meta podatkovni bazi portalnega strežnika – na ta način se poveča varnost in razširljivost celotnega sistema. Naloga portleta je pridobiti podatke, če se le da v obliki XML, in jih pretvoriti s pomočjo posebnih oblikovnih skriptov (npr. transformacije podatkov s pomočjo jezika XSL) v spletni dokument. Portlet pošlje kreirani spletni dokument strežniškemu skriptu portalnega strežnika, ki ga uvrsti na ustrezno mesto na portalu.

4. **Portal** je izdelek strežniških skriptov portalnega strežnika, fizično pa je to navadna spletna stran. Uporabnik jo vidi kot sebi prikrojeno sestavljanko informacij, ki so bile pridobljene iz različnih informacijskih virov. V našem primeru uporabniki, ki pripadajo skupini »kadrovska služba« vidijo portal »kadrovska služba«, podobno pa velja za zaposlene v upravi in nabavi podjetja.

3. Poosebitev portala

Uporabniki portala so ponavadi posamezniki, ki so lahko bolj produktivni, če je sistem prilagojen njihovim osebnim potrebam. Poosebitev portala pomeni prilagajanje portala osebnim potrebam uporabnika z namenom enostavne in učinkovite uporabe, hkrati pa pomeni zmožnost oskrbovanja uporabnika s pravimi informacijami tako od samega uporabnika kot tudi od podjetja, ki je postavilo določen portal (Schaeck, 2001).

Prvi način poosebitve je skupinska poosebitev, ki se uvede že z ustrezno strukturo uporabniških skupin in dodelitvijo pravic dostopa do informacijskih virov – opredelitev relacij med uporabniškimi skupinami in portleti. Vsi uporabniki podedujejo pravice skupin, katerih člani so, in tako pri delu uporabljajo le tiste informacijske vire, ki jih potrebujejo, oziroma jih smejo uporabljati.

Drugi način poosebitve je vsebinska poosebitev na nivoju uporabnika. Skupinska poosebitev ponudi vsakemu uporabniku v skladu z varnostno politiko nabor portletov, ki jih lahko prikaže na portalu. Uporabnik si sam izbere, katere od teh portletov bo prikazal in katere ne.

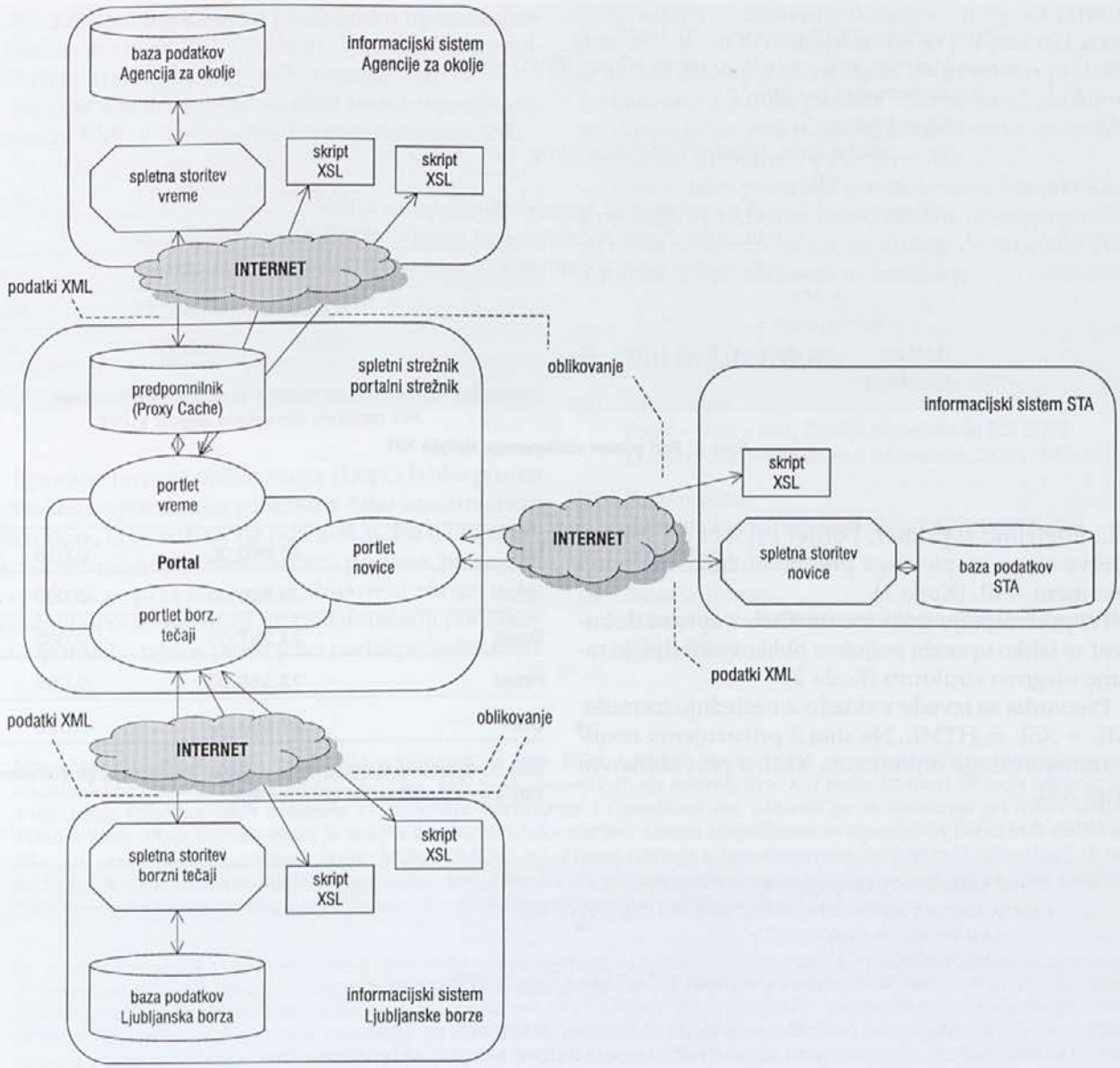
Vsebinsko portalizacijo na nivoju uporabnikov dosežemo tudi s pomočjo »pametnih portletov«, ki so sposobni v tako imenovanih piškotkih (angl. cookie) shraniti informacije o izbranem prikazu, zadnjem prikazu ali pa najbolj pogostem prikazu informacij. Piškotki so datoteke, ki se nahajajo na odjemalcu, torej pri vsakem uporabniku. V njih so shranjene informacije, s katerimi si portlet pomaga, da prikroji prikaz informacij za vsakega uporabnika.

Tretji način poosebitve je poosebitev podobe celotnega portala. To dosežemo z oblikovanjem ustrezne postavitve portletov na portalu, z izbiro ustrezne zunanje podobe portala in z minimizacijo oziroma maksimizacijo portletov. Portleti se v portalu prikažejo kot okna, ki jih lahko prikažemo v dveh stanjih, minimiziranem ali maksimiziranem.

4. Portal in spletne storitve

Spletne storitve so modularno zasnovane internetne aplikacije, ki se obnašajo kot funkcije, ki vhodne parametre sprejemajo v obliki dokumenta XML, rezultate pa prav tako posredujejo v tej obliki. Ker jih lahko uporabljamo izključno prek interneta, to je po celem svetu razširjenem računalniškem omrežju, imajo izrazito integracijsko naravo. V portalni arhitekturi igrajo vlogo podaljškov informacijskih virov, ki zagotovijo podatke v obliki XML. Portleti ob uporabi protokola SOAP dostopijo do ene ali več spletnih storitev, od katerih pridobijo ustrezne podatke. Ker so podatki zapisani v obliki XML, jih portlet s posebnimi oblikovnimi datotekami XSL enostavno pretvori v spletni dokument, ki ga portalni strežnik skupaj z drugimi pretvori v portal.

Nekateri portleti se na portalih pojavljajo pogosteje kot drugi, zato bi bilo normalno pričakovati, da se na strežnik, ki gosti ustrezno spletno storitev, pošlje več zahtevkov za podatke. Velikemu številu zahtevkov se lahko izognemo s posebnim predpomnilnikom na našem spletnem strežniku (angl. proxy cache), v katerega spravimo podatke, ko prvič uporabimo spletno storitev. V našem primeru, na sliki 2, portlet »Vreme« jemlje podatke iz predpomnilnika, kar občutno zmanjša promet v omrežju in poveča učinkovitost portalnega strežnika. Tovrstne rešitve se izkažejo za dobre le v primerih, ko je število zahtevkov veliko,



Slika 2: **Portal in spletne storitve**

podatki pa se ne spreminjajo pogosto. Portlet »Vreme« je uporaben za najširši krog uporabnikov, poleg tega pa se podatki o vremenu ne spreminjajo pogosto, na primer enkrat dnevno, kar pomeni, da je tudi podatke v predpomnilniku potrebno osvežiti le enkrat dnevno.

Portlet pridobljene podatke pretvori v spletni dokument po pravilih, ki so zapisana v oblikovnih skriptih

```
<Root>
    <Tečaj ID='1' Podjetje='Lek' Tečaj='45.060,00' Sprememba='0,018'/>
    <Tečaj ID='2' Podjetje='Krka' Tečaj='28.515,00' Sprememba='0,000'/>
    <Tečaj ID='3' Podjetje='Droga' Tečaj='34.287,00' Sprememba='-0,002'/>
    <Tečaj ID='4' Podjetje='Petrol' Tečaj='22.356,00' Sprememba='0,003'/>
    <Tečaj ID='5' Podjetje='Sava' Tečaj='17.053,00' Sprememba='-0,026'/>
</Root>
```

Koda 1: **Podatki v obliki XML, ki jih portlet pridobi od spletne storitve**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
    <xsl:template match="/">
    <table border="0">
        <xsl:for-each select="Root/Tečaj" order-by="+ @ID">
            <tr>
                <td><xsl:value-of select="@Podjetje"/></td>
                <td align="right"><xsl:value-of select="@Tečaj"/></td>
                <td align="right"><xsl:value-of select="@Sprememba"/></td>
            </tr>
        </xsl:for-each>
    </table>
</xsl:stylesheet>
```

Koda 2: **Prvi primer oblikovnega skripta XSL**

XSL. Poglejmo si primer. Portlet pridobi od spletne storitve ali pa iz spletnega predpomnilnika naslednji dokument XML (Koda 1).

Pri pretvarjanju dokumenta XML v spletni dokument se lahko uporabi poljuben oblikovni skript, ki razume njegovo strukturo (Koda 2).

Pretvorba se izvede v skladu z naslednjo formulo: XML + XSL = HTML. Na sliki 3 prikazujemo rezultat transformacije dokumenta XML s prvi oblikovni skript XSL.

Lek	45.060,00	0,018
Krka	28.515,00	0,000
Droga	34.287,00	-0,002
Petrol	22.356,00	0,003
Sava	17.053,00	-0,026

Slika 3: **Spletni dokument, ki je rezultat transformacije dokumenta XML s prvim oblikovnim skriptom XSL**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
    <xsl:template match="/">
    <table border="1">
        <xsl:for-each select="Root/Tečaj" order-by="+ @Podjetje">
            <tr>
                <td><xsl:value-of select="@Podjetje"/></td>
                <td align="right"><xsl:value-of select="@Tecaj"/></td>
                <td align="right"><xsl:value-of select="@Sprememba"/></td>
                <td align="center">
                    <img>
                        <xsl:choose>
                            <xsl:when test="@Sprememba < 0">
                                <xsl:attribute src="dol.gif">red</xsl:attribute>
                            </xsl:when>
                            <xsl:when test="@Sprememba > 0">
                                <xsl:attribute src="gor.gif">green</xsl:attribute>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:attribute src="enako.gif">white</xsl:attribute>
                            </xsl:otherwise>
                        </xsl:choose>
                    </img>
                </td>
            </tr>
        </xsl:for-each>
    </table>
</xsl:stylesheet>
```

Koda 3: **Drugi primer oblikovnega skripta XSL**

Na zahtevo uporabnika pa se lahko uporabi drugi oblikovni skript XSL (Koda 3). Tega sproži uporabnik na portalu (klik na gumb, kazalec, itd).

Na sliki 4 prikazujemo rezultat transformacije dokumenta XML z drugim oblikovnim skriptom XSL.

Droga	34.287,00	-0,002	↓
Krka	28.515,00	0,000	=
Lek	45.060,00	0,018	↑
Petrol	22.356,00	0,003	↑
Sava	17.053,00	-0,026	↓

Slika 4: **Spletni dokument, ki je rezultat transformacije dokumenta XML z drugim oblikovnim skriptom XSL**

Uporabo novega oblikovnega skripta lahko portlet za vsakega uporabnika zabeleži v tako imenovanem »piškotku«, ki se nahaja na odjemalcu. Ko bo uporabnik naslednjič želel videti vsebino portleta, bo portlet iz »piškotka« prebral, kateri oblikovni skript je bil nazadnje uporabljen in ga pri transformaciji podatkov tudi uporabil – tako si prikrojimo portal po lastni meri.

Uporaba oblikovnih skriptov omogoča izredno fleksibilnost in učinkovitost, saj so isti podatki znova in znova uporabljeni, ne da bi jih ponovno pridobili. To bistveno zmanjša promet v omrežju. Z oblikovnimi skripti lahko iste podatke predstavimo na različne načine, jih filtriramo, sortiramo, itd.

Portali tako postajajo glavni komunikacijski kanal v podjetjih, predvsem v zelo velikih. So enotna vstopna točka za komunikacijo in dostop do različnih virov informacij v organizaciji in izven nje.

5. Viri in Literatura

[1] Fabjan Borut:
Portal – vrata v svet, Zbornik posvetovanja DSI 2001. Ljubljana: Slovensko društvo Informatika, 2001. 551 str.

[2] Poslovni portali
(URL:http://www.ixtlan-team.si/novice/prispevki/portal/default.asp), 15.9.2001.

[3] Schaeck Thomas:
WebSphere Portal Server and Web Services Whitepaper. IBM, 2001. 23 str.

◆

Mag. Miroslav Ribič, rojen 10.7.1974 v Ljubljani, je leta 1993 zaključil Srednjo šolo za računalništvo. Izobraževanje je nadaljeval na Ekonomski fakulteti v Ljubljani, kjer se je usmeril v študij informatike. Kot redni študent EF se je ukvarjal tudi z izgradnjo informacijskih sistemov, ki podpirajo upravljanje s človeškimi viri, aktivno pa je sodeloval pri informatizaciji Skladov dela. Dodiplomski študij je zaključil z delom »Informacijski sistem spremljanja in usmerjanja presežnih delavcev v RS«. Kot imetnik Microsoftovih licenc MCP in MCSD se aktivno ukvarja s proučevanjem internetnih tehnologij. S tega področja je tudi uspešno ubranil magistrsko delo z naslovom »Implementacija elektronskega poslovanja med podjetji«. Trenutno je zaposlen v podjetju IDS Scheer, kjer sodeluje pri uvajanju rešitev podjetja SAP.

◆

Dr. Andrej Kovačič je v zadnjih desetih letih delal kot projektant, razvijalec in svetovalec pri projektih strateške prenove in informatizacije poslovanja. Je izredni profesor s področja poslovne informatike na Ekonomski fakulteti in Visoki upravni šoli ter predstojnik Inštituta za poslovno informatiko pri EF v Ljubljani. Bil je dolgoletni predsednik programskega odbora Dnevov slovenske informatike v Portorožu, je član izvršnega odbora Slovenskega društva Informatika, član uredniškega odbora revije Uporabna informatika, svetovalec in veščak s področja vodenja in upravljanja podjetij (PHARE, Zveza ekonomistov) in pooblaščeni revizor informacijskih sistemov.

◆

TEHNOLOGIJE AGENTOV – UPORABA V ELEKTRONSKEM POSLOVANJU

Denis Trček
Institut "Jožef Stefan", Jamova 39, 1000 Ljubljana
E-pošta: denis.trcek@ijs.si

Povzetek

Elektronsko poslovanje je prineslo precej novih paradigem v poslovanje in ena od njih so tehnologije agentov. Tovrstne tehnologije združujejo več področij, ki segajo od objektnega programiranja kot takega prek komunikacij, umetne inteligence pa do varnosti. V prispevku je podan pregled omenjenega področja ter primer uporabe v nabavni poslovni funkciji.

Ključne besede: elektronsko poslovanje, tehnologije agentov.

Abstract

E-BUSINESS APPLICATIONS OF AGENT TECHNOLOGIES

Many new paradigms have been introduced recently with the proliferation of e-business applications, and agent technologies are among the most important ones. They join complementary areas that include object programming, communications, artificial intelligence and security. In the article an overview of this field is given with the emphasis on use of agent technologies in supply chain management.

■ ■ ■

Uvod

Danes poznamo kar nekaj področij, kjer se srečujemo s tehnologijami agentov, npr. upravljanje omrežij in izobraževanje. Na področju elektronskega poslovanja so agenti postali dejstvo predvsem pri dejavnostih trženja (npr. pri iskanju in določanju ciljnih skupin prek svetovnega spleta), nabave (npr. pri iskanju najugodnejših ponudnikov), vse bolj prodirajoč segment mobilnih komunikacij pa širi uporabo teh tehnologij tudi v sektor turizma. Tak primer je projekt CRUMPET (Creation of User Friendly Mobile Services Personalized for Tourism) [FIP01a], kjer bodo podprte najnovejše prilagodljive nomadske storitve preko različnih vrst infrastrukture (IP, WLAN, GSM, GPRS in UMTS).

Najbolj poznana okolja za upravljanje agentov (OUA) so:

- Grasshopper (podrobnosti so dosegljive na naslovu http://www.ikv.de).
- FIPA-OS (podrobnosti so dosegljive na naslovu http://fipa-os.sourceforge.net).
- JADE (podrobnosti so dosegljive na naslovu http://sharon.cselt.it/projects/jade).
- ZEUS (podrobnosti so dosegljive na naslovu http://www.labs.bt.com).

Koncepti vseh teh agentnih okolij so si podobni. Namen tega članka je podati tipično strukturo in delovanje agentov, nato pregled okolij za upravljanje agentov in možnost njihove uporabe pri modeliranju ter podpori poslovnih procesov.

Tehnologije agentov – lastnosti in arhitekture

Evolucija na področju agentov ima korenine v začetkih objektnega programiranja, ko smo pri programski realizaciji reševanja problemov iz realnega sveta začeli uporabljati koncept objektov, to je zaključenih enot programske kode, katerim so dodeljeni določeni podatki in pa pripadajoče metode. Razvoj globalnega omrežja in mobilnih komunikacij pa je pripeljal do tega, da so objekti postali funkcionalno vse bolj sposobni. Tiste, ki jih opredeljujejo lastnosti, ki so imitacija človeškega obnašanja, imenujemo agenti. Ti se pri realizaciji internih izračunov poleg lastnih metod poslužujejo še metod preostalih objektov, interne izračune pa vršijo na osnovi prepričanj, sposobnosti in izbir ter komunicirajo med sabo z visokonivojskim jezikom, ki ni neposredno povezan z izračuni.

Definicij agentov je več, ena najpogosteje uporabljanih in najbolj generičnih pa je podana v [Gri01]. Ta pravi, da so agenti programske rešitve, ki izkazujejo naslednje lastnosti:

a) Avtonomnost – agent samodejno začne dejavnosti skladno s svojimi cilji, ima lasten nadzor in lahko deluje v imenu uporabnika ter je v znatni meri neodvisen od sporočil preostalih agentov.
b) Prilagodljivost – agent je sposoben prilagajati svoje obnašanje, bodisi na podlagi lastnega učenja, prilagajanja uporabnika ali pa z naknadnim nalaganjem ustrezne kode iz okolja.

c) Mobilnost – agent se je sposoben premestiti iz enega okolja v drugo, naj si bo s prenosom same kode in zagonom na novi lokaciji ali pa s t.i. serializacijo, kjer se preneseta koda in stanje agenta in se nadaljuje izvajanje v novem okolju.
d) Inteligenca – agent je sposoben razmišljati o svojih ciljih, pridobljenih informacijah in ostalih agentih ter uporabnikih.
e) Sposobnost sodelovanja – agent zna komunicirati in sodelovati z drugimi agenti v okolju pri realizaciji določenega cilja, pa naj si bo to v dinamičnih ali pa statičnih okoljih.
f) Obstojnost (persistenca) – agenti so ob podpori infrastrukture sposobni ohranjati stanje in znanje na daljši rok.

Pred nadaljnjo obravnavo podajmo definicijo združbe agentov. Združba agentov je definirana kot skupek agentov, kjer posamezni agenti nimajo vseh potrebnih sposobnosti za rešitev problema, kjer ni nekega centraliziranega nadzora, kjer so podatki decentralizirani in kjer je računanje asinhrono [Flo99].

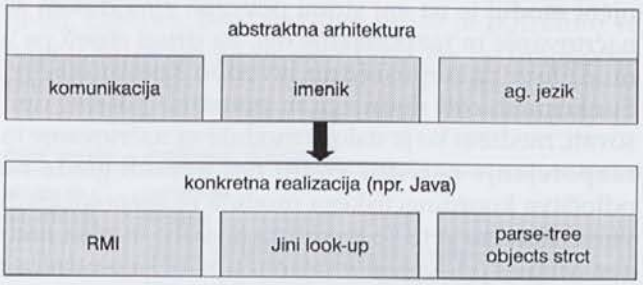
Iz definicije lahko neposredno ugotovimo, da je internet naravno okolje za združbe agentov, ki so opredeljene z naslednjimi lastnostmi:

- Arhitekturo agentov kot takih, izhajajoč iz zgornje definicije, od katerih so eni agenti neposredno opravilni, drugi pa infrastrukturne narave.
- Arhitekturo združbe same, ki zahteva tudi definicijo sledečih elementov:
  - Jezika, ki predstavlja bistvo komunikacije med agenti in je deklarativne, to je pojasnjevalne narave (definicije, predpostavke), kjer sta najpogostejša predstavnika Knowledge Query and Manipulation Language [Fin97] in FIPA ACL [FIP01b].
  - Komunikacijskih protokolov, ki opredeljujejo načine izmenjave sporočil v omenjenem jeziku in format njihove predstavitve, kar je v najpreprostejši obliki lahko izvedba prek vtičnic in sklada TCP/IP ali pa elektronske pošte.
  - Ontologij, ki pomenijo način soglašanja o pomenu konceptov. Definirane so kot sheme za opis konceptov in njihovih relacij v nekem komunikacijskem kontekstu. Običajno pri tem izhajamo iz predikatnega računa prvega reda. Primer ontologije je Ontolingua [Gru93].
  - Izvajanja registracije imen in storitev, s čimer je omogočena identifikacija obstoječih agentov in vzpostavitev komunikacije z njimi ob poznavanju lastnosti (sposobnosti) le-teh.

Ena najbolj uveljavljenih skupin standardov, ki pokriva do sedaj omenjena področja za zagotovitev interoperabilnosti heterogenih agentov, je tista, ki jo sprejema FIPA (The Foundation for Intelligent Physicall Agents). Specifikacije FIPA zajemajo abstraktno arhitekturo, komunikacijo agentov, upravljanje agentov, transport sporočil in pa aplikacije.

Abstraktna arhitektura

Spodnja slika prikazuje abstraktno arhitekturo tehnologij agentov.

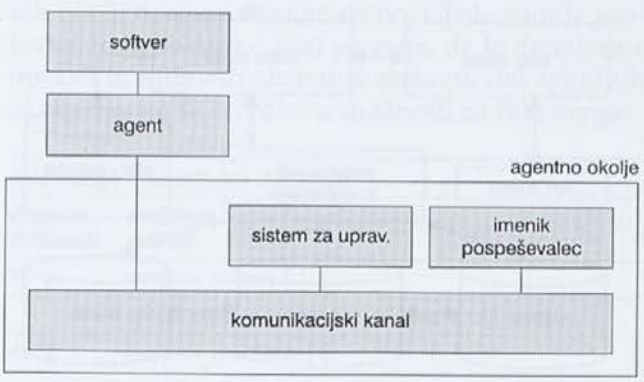


Slika 1: Abstraktna arhitektura tehnologij agentov.

Abstraktna arhitektura se sestoji iz agentov, opredelitve komunikacije, imenika in jezika, ki ga agenti uporabljajo za komunikacijo. Na sliki vidimo primer realizacije v javanskem okolju, kjer komunikacija poteka prek javi lastnega protokola RMI, za imenik pa služi storitev Jini look-up.

Agentno okolje

Drug zorni kot na tehnologije agentov nudi arhitektura OUA, ki je podana na spodnji sliki. Agent komunicira z aplikacijskim softverom ali pa neposredno z uporabnikom. Prek komunikacijskega kanala je povezan s sklopom za upravljanje in pa dvema vrstama infrastrukturnih agentov. Prva opravlja funkcijo klasičnega imenika, ki omogoča identifikacijo agentov in njihovo lokacijo, druga pa hrani podatke o funkcionalnosti (lastnostih, sposobnostih) posameznih agentov.



Slika 2: Okolje za upravljanje agentov.

Poleg teh ključnih komponent OUA vsebujejo še knjižnice z ustreznimi algoritmi, protokoli in podatkovnimi strukturami ter orodja za prikazovanje (npr. raznih statističnih veličin).

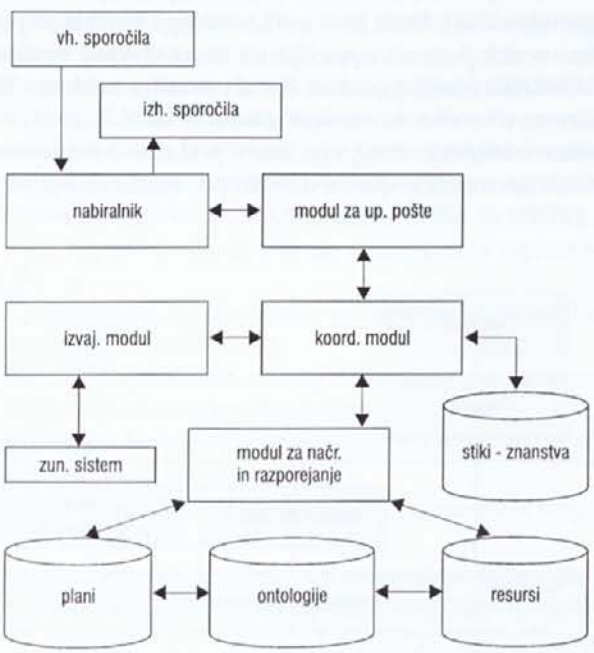
Predstavitev tipične arhitektura samega agenta povzemamo po [Col99]. V osrčju agenta je koordinacijski modul. Ta sprejema odločitve glede ciljev agenta, npr. kdaj neki cilj realizirati ali opustiti, in pa koordinira komunikacijo s preostalimi agenti. Omenjeni modul je na eni strani povezan z modulom za načrtovanje in razporejanje del, na drugi strani pa z modulom za neposredno izvedbo komunikacije. Funkcionalnosti slednjega ni potrebno posebej opisovati, medtem ko je naloga modula za načrtovanje in razporejanje narediti vrstni red opravil glede na odločitve koordinacijskega modula in razpoložljivih virov. Končno je tu še modul za izvedbo in njen nadzor. Moduli pri svojem izvajanju uporabljajo ustrezne notranje in zunanje zbirke podatkov, kot je to razvidno s slike 3. Med osrednjimi je zbirka ontologij.

Modeliranje poslovnih procesov

V tem razdelku bomo podali primer uporabe tehnologij agentov za reševanje problemov na področju poslovanja. Konkretno bo to modeliranje obvladovanja verige dobav.

Ko s pomočjo tehnologij agentov rešujemo določen problem, je pristop analogen pristopom, ki jih poznamo s področja sistemske analize in načrtovanja. Po analizi začnemo z izdelavo konceptualnega modela, ki ga vse bolj razgrajujemo in formaliziramo, tako da dobimo logični model. Tega kodiramo v ustreznem programskem jeziku in ga končno preslikamo v fizični / izvedbeni model. Torej:

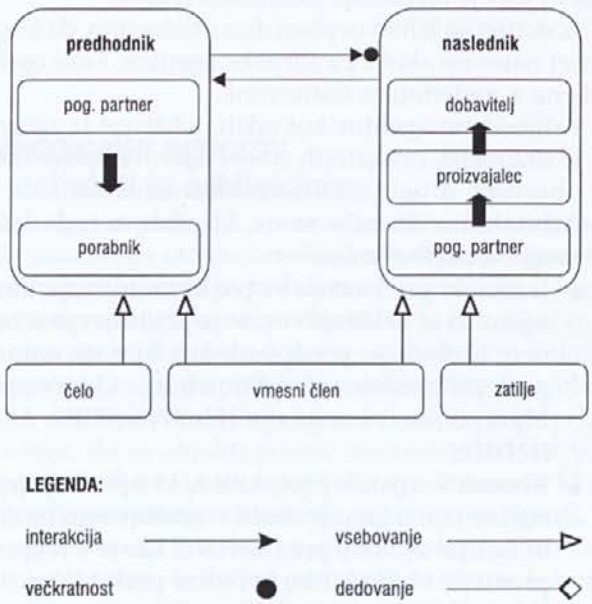
- V določeni domeni (poslovnem okolju) opravimo analizo poslovnega procesa, kjer opredelimo probleme, želene rešitve ter koncept vlog udeležencev.
- V drugem koraku identificiramo potrebne agente, storitve in ontologije, kar predstavlja dizajniranje.
- Sledi realizacija v okviru agentnega okolja, kjer kreiramo agente, jih ustrezno organiziramo in koordiniramo.



Slika 3: **Primer arhitekture agenta.**

Modeliranje vlog

Za naš primer bomo vzeli enostaven primer verige dobav pri izdelavi pametnih kartic. Dobavitelja v verigi sta izdelovalec mikroprocesorjev (IMP) in dobavitelj plastičnih kartic (DPK), ki dostavljata komponente izdelovalcu pametnih kartic (IPK). Slednji vstavi mikroprocesor s pomočjo epoksi rezine v ustrezen utor na kartici in opravi poosebitev le-te.



Slika 4:
Vloge v verigi dobav (veriga je gledana s strani končnega porabnika).

Model vsebuje dve vlogi: predhodnik in naslednik. Modelirane vloge prikažemo z diagramom, ki izhaja iz jezika UML (Universal Modelling Language [Alh98]). Zgornja slika je osnovni gradnik poljubne verige dobav, ki prikazuje ključne vloge in relacije. Gledana je s strani končnega uporabnika, ki je torej v interakciji s čelom verige. Rečeno drugače, čelo verige se imenuje tista točka, ki je v interakciji s končnim uporabnikom. Na spodnji sliki je tako prikazan poteka interakcije s končnim porabnikom.

Slika 5: **Potek interakcije.**

Zgornji potek interakcije predstavlja prav tako osnovni vzorec vzdolž verige dobav. Na tej osnovi lahko podrobno definiramo vsako vlogo, kar je nujno pred dokončno implementacijo. Definicija vsake vloge poleg splošnega opisa zahteva opredelitev relacij le-te do ostalih vlog, odgovornosti, zunanje vmesnike in predpogoje. Očitno v našem primeru IPK nastopa na čelu verige, vmesnih členov nimamo, zatilje pa predstavljata IMP in DPK. Naš model vlog podaja spodnja tabela:

agenti	vloge
IPK	čelo verige – začetnik pogajanj, porabnik
IMP	zatilje verige – pogajalski partner, dobavitelj, proizvajalec
DPK	zatilje verige – pogajalski partner, dobavitelj, proizvajalec

V naslednjem koraku je potrebno doreči, kako bodo agenti realizirali zadane vloge. Vsaka vloga potegne za sabo določene dolžnosti, ki so po naravi družbeno (socialno) in neposredno (izvedbeno) naravnane. Navedimo obe skupini odgovornosti najprej za udeleženca, ki je na čelu verige.

čelo verige – družbene dolžnosti	
vloge	**dolžnosti**
1) začetnik pogajanj	a) Agent mora poznati in se zavedati lastnosti / sposobnosti preostalih agentov.
	b) Agent mora znati izvesti povpraševanje za dobavo materiala.
	c) Agent se mora znati pogajati glede cene materiala.
	d) Agent mora znati posredovati informacije o naročilih lastni vlogi porabnika.
2) porabnik	a) Agent mora znati prejeti material.

čelo verige – izvedbene dolžnosti	
vloge	**dolžnosti**
3) začetnik pogajanj	a) Agent mora podati zahtevo za nov material.
4) porabnik	a) Agent mora obdelati material.

Podobno opredelimo dolžnosti za zatilje verige. Da pa ne bi presegli okvirov tega članka, smo primer napravili samo za čelo verige.

Snovanje aplikacije

S konceptualnega nivoja se v tej fazi selimo proti logičnemu nivoju. Dolžnosti, povezane s posameznimi vlogami, moramo preslikati na način, kot jih zahteva primer, ki ga proučujemo in OUA, v okviru katerega bo potekala realizacija. Rečeno poenostavljeno, človeku predstavljive opise je potrebno še naprej razgrajevati in sicer tako, da bo omogočen prenos v OUA.

čelo verige – družbene dolžnosti	
vloge in dolžnosti	**problemi rešitve**
1) a)	znanje [dobavitelj, zahtevan_material]
	pridobivanje podatkov o zmožnostih ostalih agentov
1) b)	začetek_dialoga [dobavitelj, zahtevan_material]
	omogočiti agentu uporabo ustreznega koordinacijskega protokola
1) c)	vrednotenje [material, cena]
	omogočiti agentu uporabo pogajalskih strategij
1) d)	pošiljanje_sporočila [vloga_porabnika, podrobnosti_dobave]
	omogočiti agentu pošiljanje sporočil
2) a)	prevzem_materiala [ ]
	definiranje izvedbe opravila in njegovih atributov

Komentirajmo najprej postavko 1) a). Da agent pridobi ustrezno znanje, je moč definirati statične povezave pri generiranju agentov, lahko pa agenti uporabijo infrastrukturne možnosti agentnega okolja in sami vzpostavijo ustrezne povezave. Glede postavke 1) b) – agentu moramo omogočiti dostop do ustreznega protokola za začetek in izvrševanje dialoga, npr. Contract Net [Smi80]. Podobno velja glede pogajalskega protokola, kjer lahko uporabimo npr. protokol GrowthStrategy [Col99]. Predzadnja postavka (to je postavka 1) d), pošiljanje sporočila) je v večini agentnih okolij dostopna avtomatično. Glede zadnje postavke, to je postavke 2) a) pa velja, da je definiranje opravil in njihovih atributov sestavni del agentnih okolij. Sledijo še izvedbene dolžnosti za čelo verige:

čelo verige – izvedbene dolžnosti	
vloge in dolžnosti	**problemi rešitve**
3) a)	zahteva_dobave [ ]
	sproži zahtevo za dobavo skladno s cilji agenta
4) a)	izdelava_proizvoda [ ]
	porabi material in naredi proizvode skladno s cilji agenta

V naslednjem koraku moramo pri pretvarjanju v logični model modelirati koncepte dane domene (ontologije), kar izvedemo prek poimenovanja vsakega koncepta, pripadajočih atributov in ustreznih omejitev. V našem primeru obvladovanja verige dobav je ontologija naslednja:

koncepti	atributi (tipi)	omejitve
procesor	proizvajalec (alfanumeric)	[intel | motorola]
	model (alfanumeric)	
plastična_kartica	material (character)	[ABS | PETP]
	standard (alfanumeric)	

V zgornji tabeli atribut proizvajalec označuje proizvajalca mikroprocesorjev, atribut model konkreten model mikroprocesorjev, material označuje vrsto plastike za izdelavo pametnih kartic, standard pa skladnost kartic z določenimi standardi.

Realizacija aplikacije

Na tej točki lahko pristopimo k realizaciji aplikacije. Problem je obdelan do nivoja, ki omogoča neposredno uporabo algoritmov. Teh večinoma ni potrebno pisati samemu razvijalcu, ker so vgrajeni v OUA. Z ustreznimi orodji v okviru konkretnega OUA naredimo naslednje:

- kreiramo ontologijo;
- kreiramo osnovne agente, opis opravil ter organiziramo in koordiniramo agente;
- kreiramo infrastrukturne agente (agenta, ki opravlja funkcijo imenika in agenta, ki vodi evidenco o sposobnostih preostalih agentov).

Ta opravila opravimo kar prek ustreznih grafičnih vmesnikov, podobno kot to počnemo pri programiranju zbirk podatkov (npr. v MS Access).

Sedaj poženemo aplikacijo, s čimer združba agentov "oživi". Dogajanja tekom življenjskega ciklusa združbe agentov lahko opazujemo s pomočjo ustreznega orodja, t.i. vizualizatorja, ki je pravzaprav specializiran agent. Predmet opazovanja je lahko posamezen agent, nadalje dinamika vzpostavljanja odnosov med agenti v realnem času, kaj se dogaja v verigi dobav s surovinami itd. Vizualizator skratka omogoča opazovanje in zajem raznih statistik.

Zaključek

V pričujočem prispevku smo se osredotočili na možnost realizacije poslovnih procesov s pomočjo tehnologij agentov. Resnici na ljubo smo ostali na nivoju simulacije, vendar že danes tovrstna uporaba agentnih tehnologij v poslovnem svetu predstavlja le enega od načinov uporabe.

Če pa komentiramo samo simulacijo, je glavna prednost modeliranja poslovnih procesov s pomočjo teh tehnologij ta, da pristop k modeliranju dobro odseva realni svet, ki se sestoji iz entitet, ki v medsebojnih interakcijah skladno s svojimi cilji in motivi poskušajo realizirati določene cilje. To je bistveno drugače, kot če uporabimo matematične modele in proces podamo z ustreznimi enačbami, odstopanja pa potem naknadno vnašamo npr. s stohastičnimi elementi. V primeru tehnologij agentov bi lahko rekli, da imamo opravka z reševanjem od dna k vrhu, v klasičnem primeru pa z reševanjem od vrha k dnu. S spremembami poljubnih karakteristik vzpostavljenega sistema lahko opazujemo učinke in poiščemo ustrezne rešitve za naše poslovno okolje. S tovrstnimi orodji lahko razvijemo realen občutek za upravljanje proizvodnih in storitvenih sistemov, kar je s klasičnimi analitičnimi pristopi težje.

Je pa seveda na področju tehnologij agentov še precej odprtih vprašanj. Eno glavnih je vprašanje varnosti. Namreč agenti, še zlasti mobilni, so izpostavljeni novim vrstam napadov (tako metode kot njihovi podatki). Do sedaj je bil koncept varovanja v omrežju tak, da smo morali ščititi kodo, vezano na nek procesor. Ščitili smo "matično kodo", to je tisto kodo, ki je imela domicil v določenem strojnem okolju. Pri mobilnih agentih ni več moč govoriti o matičnem procesorju in jih je pravzaprav potrebno ščititi pred okoljem, kjer gostujejo. Klasična kriptografija v primeru mobilnih agentov odpove. Precej je obetal pristop s t.i. mobilno kriptografijo [San98], vendar pa kaj več od grobe teoretične osnove na tem področju ni bilo narejenega. Bi bila pa to generična rešitev za probleme, povezane z varnostjo agentov.

Ne glede na to, da so trenutne rešitve še okorne in je marsikaj nedorečenega (zlasti na področju varnosti), so tehnologije agentov obetavno področje s široko namembnostjo uporabe. V bližnji bodočnosti bodo verjetno botrovale tudi novim poslovnim modelom in ne samo izboljševanju obstoječih.

Literatura

[Gri01] Griss L.M., Accelerating Development with Agent Components, Computer, vol. 34, št. 5, IEEE, maj 2001, str. 37-43.

[Tec98] Tecuci G., Building Intelligent Agents, Acedemic Press, New York 1998.

[Gue99] Guessoum Z., Briot J.P., From Active Objects to Autonomous Agents, IEEE Concurrency, vol. 7, št. 3, marec 1999, str. 68-81.

[Bie98]
Bieszcad A., White T., Pagurek B., Mobile Agents for Network Management, IEEE Comm. Surveys, vol. 1, št. 1, 4th quarter 1998.

[San98]
Sander T., Tschudin C.F., Protecting Mobile Agents Against Malicious Hosts, Mobile Agent Security, LNCS 1419, Vigna G. editor, Springer Verlag, februar 1998.

[Flo99]
Florez M.R.A., Towards a Standardisation of Multi-Agent System Frameworks, Crossroads, ACM, avgust 1999.

[Fin97]
Finin T., Laboru Y., Mayfield J., KQML as an Agent Communication Language, Software Agents, AAAI Press, Menlo Park 1997.

[Gru93]
Gruber T.R., A Translation Approach to Portable Ontology Specifications, Proc. Of KAW '93, Banf 1993, str. 199-200.

[FIP01a]
Makelainen M., CRUMPET – Tourism and Open Source for Small Devices, Inform!, FIPA, vol. 2, št. 1, marec 2001.

[FIP01b]
FIPA, ACL Message Structure Specification, FIPA XC00016E, Geneva, avgust 2001.

[Col99]
Collis J., Ndumu D., The Zeus Agent Building Toolkit, Parts 1 thru 4, BT, november 1999.

[Alh98]
Alhir S.S, UML in a Nutshell, O' Reilly, 1998.

[Smi80]
Smith G.R., The contract net protocol: High-level communication and control in distribuited problem solver, IEEE Transactions on Computers, Vol. 29, št. 12, december 1980, str.1104—1113.

◆

Denis Trček se ukvarja s področjem e-poslovanja s poudarkom na varnosti že dobrih deset let. S tega področja ima prek šestdeset objav, v glavnem v tujini, reference pa vključujejo tudi sodelovanje na aplikativnih projektih doma: od vzpostavitve informacijske infrastrukture Narodne galerije v Ljubljani do arhitekture varovanja komunikacij pri projektu Kartice zdravstvenega zavarovanja ZZZS. Avtor je docent za področje informatike, je zaposlen na Institutu "Jožef Stefan" in je sodelavec Fakultete za računalništvo in informatiko Univerze v Ljubljani.

◆

UPORABA INTELIGENTNIH SISTEMOV PRI NAPOVEDOVANJU GIBANJA TEČAJEV VREDOSTNIH PAPIRJEV

Alojz Tapajner, Peter Kokol
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Smetanova 17, 2000 Maribor
alojz.tapajner{kokol}@uni-mb.si

Povzetek

V zadnjem času dosega trgovanje z delnicami v tujini vse večji razmah, predvsem med malimi delničarji. Trgovanje postaja hkrati šport in igra na srečo, ki je po obsegu že zdavnaj prekosila tradicionalne oblike, kot sta na primer loto ali stave. Vsakdo poskuša v čim krajšem času zaslužiti čim več. Pri natančnejšem opazovanju gibanja tečajev je razvidno, da lahko največ zaslužimo s kratkoročnimi transakcijami. Vprašali smo se, kakšne rezultate bi lahko pri napovedovanju gibanja tečajev dosegli z inteligentnimi sistemi in predvsem kakšna je ponovljivost dogodkov na borzi, ali je te dogodke možno napovedati? Odločili smo se, da analiziramo gibanje tečajev v preteklosti in na podlagi iskanja sorodnosti med trenutnim in preteklim stanjem napovemo, kaj se bo zgodilo v bližnji prihodnosti. Kot osnovo inteligentnega sistema smo uporabili odločitvena drevesa.

Abstract

USE OF INTELLIGENT SYSTEMS FOR STOCK MARKET PREDICTION

In economicaly developed countries stock trading has been growing recently to even larger dimensions, particularly among small investors. Everybody tries to make a fortune in a short time. When observing the stock charts it would seen that short term trading is the best solution to earn money. The question is whether it is possible to predict those occurrences with intelligent systems? We decided to analyse stock performance in the past and based on the research of relationship between past and present to predict what will happen in near future. To accomplish this task we have built several intelligent systems based on decision trees.

■ ■ ■

1. Uvod

V zadnjem času dosega trgovanje z vrednostnimi papirji (delnicami) v gospodarsko razvitih državah vse večji razmah, predvsem med malimi delničarji. Trgovanje postaja hkrati nacionalni šport in igra na srečo, ki je po prometu že zdavnaj prekosila tradicionalne oblike, kot sta npr. loto ali stave. Vsakdo poskuša v kratkem času zaslužiti čim več. Pri nakupih delnic pa poslovanje in položaj podjetij na trgu že dolgo nista edino merilo. Pri natančnejšem opazovanju trga je namreč razvidno, da se zasluži največ s kratkoročnimi investicijami. Analizirali smo gibanje tečajev v preteklosti in na podlagi iskanja sorodnosti med današnjim in preteklim stanjem napovedali, kaj se bo zgodilo v bližnji prihodnosti z uporabo inteligentnih sistemov (IS) in sicer odločitvenih dreves.

Pri pregledu literature smo zasledili številne matematične modele, ki napovedujejo gibanje tečajev vrednostnih papirjev, ki imajo dokaj ustaljeno nihanje (vrednostni papir počasi raste ali oscilira). Številni inteligentni sistemi napovedujejo kar gibanje centralnega borznega indeksa (Dow Jones, DAX), kjer pa gre v glavnem le za napovedovanje borznega trenda [9-11]. Rezultati teh IS so zelo dobri, saj dosegajo tudi verjetnosti zadetka nad 70 %, v primeru turbulenc (večje dinamike gibanja) pa rezultati niso več uporabni. Naš cilj je napovedovanje tečajev podjetij tako imenovanih novih trgov (NASDAQ, Neuer Markt), saj dinamika gibanja tečajev pomeni večji izziv za analizo, z ekonomskega vidika pa tudi največjo možnost zaslužka. Kljub temu, da delnice nove ekonomije doživljajo v zadnjem letu in pol velike zlome, centralni indeks Nasdaq je izgubil od marca 2000, ko je dosegel rekord, pa do danes okoli 60% in Nemax več kot 80%, se situacija glede možnosti velikega zaslužka ni spremenila, povečala se je le stopnja tveganja.

Za analizo vrednostnih papirjev smo izdelali številne matematične modele, s katerimi smo obdelali tečaje in promet ter izračunali najrazličnejše kazalnike (indikatorje) in jih v obliki vhodnih atributov uporabili za generiranje odločitvenih dreves. Analiza, ki smo jo opravili, je po našem mnenju dokaj sorodna tehnični analizi vrednostnih papirjev, saj smo za analizo uporabili le tečaje in promet vrednostnih papirjev.

2. Odločitvena drevesa

Odločitvena drevesa spadajo med metode induktivnega učenja, torej avtomatskega učenja iz rešenih primerov [2, 3]. Cilj je odločitvena struktura - drevo, ki temelji na učnih primerih in je sposobno čim bolj

uspešno »rešiti« še neznani primer. Sam proces učenja je zajet v oblikovanju dreves, saj odločitvena drevesa hranijo izluščeno znanje prav v svoji obliki. Samo drevo sestoji iz dveh vrst vozlišč – atributna (testna) vozlišča so testi vhodnih vzorcev, končni listi pa so kategorije testiranih vzorcev. Drevo vhodni vzorec klasificira s sprehodom od začetnega vozlišča do končnega lista s sprotnim odločanjem v vsakem vozlišču o nadaljnji smeri sprehoda. Sama drevesa se ločijo po številu testov, ki jih lahko opravimo v enem vozlišču, po številu možnih odgovorov in tudi po številu razredov, kamor lahko drevo uvrsti neki vzorec. Sami testi v notranjih vozliščih se lahko izvajajo nad numeričnimi ali diskretnimi vrednostmi in so torej ustrezne oblike, vsak list pa predstavlja možni odgovor na zastavljeno vprašanje. Glavni odliki dreves sta njihova preprostost za uporabo in izraznost rešitve ter naučenega znanja – zgradba drevesa je osnova za napoved, v kateri razred spada neki vzorec in katera vprašanja so pri tej odločitvi pomembna.

3. Tehnična analiza vrednostnih papirjev

Tehnična analiza delnic daje nakupni ali prodajni signal glede na določene grafične formacije gibanja tečajev delnic ali njihovega prometa v preteklosti [6]. Obstaja še temeljna analiza podjetja, kjer gre za oceno položaja podjetja danes in v prihodnosti, vendar lahko takšno oceno poda le strokovnjak s področja, v katerem podjetje posluje, tako da natančno opredeli vse možnosti in tveganja, ki jih panoga vsebuje. Uspešen investitor mora upoštevati obe analizi, temeljno za izbiro ustreznega podjetja (panoge) in tehnično za pravočasen nakup in prodajo [5].

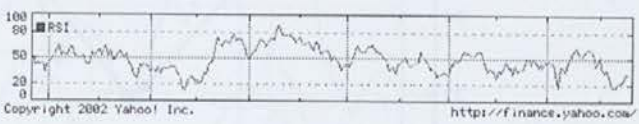
Značilnost tehnične analize je, da na vhodu (vhodni podatek) sprejme veliko količino podatkov o preteklem trgovanju. Investitorja po drugi strani zanima le odgovor na vprašanje ali izvesti transakcijo z določenim vrednostnim papirjem ali ne. Tehnična analiza omogoča enostaven grafičen prikaz nekaterih dejstev in trendov, na podlagi katerih se investitor lažje odloči za nadaljne akcije. Odgovori na vprašanja so prikazani s pomočjo grafičnih krivulj, ki so izračunane na osnovi določenih matematičnih formul. Številne tuje finančne ustanove in razni finančni portali nudijo storitev tehnične analize brezplačno na internetu. Mi smo za spodnje primere uporabili spletno aplikacijo, ki jo nudi Yahoo (http://finance.yahoo.com).

Obstaja mnogo kazalnikov tehnične analize vrednostnih papirjev, omenili bomo le najznačilnejše:

1. **Drseče povprečje** (Moving Average)
Drseče povprečje prikazuje zglajeno različico grafa gibanja tečajev delnice. Z drsečim povprečjem izločimo moteča nihanja, tako da lažje prepoznamo trend gibanja tečajev. Obdobja, ko so tečaji nad krivuljo drsečega povprečja, označujemo kot *pozitivna* (v borznem žargonu *bikovska*). Obratno velja, da kadar so tečaji pod krivuljo drsečega povprečja, je trend *negativen* (*medvedji*). Kazalnik pove tudi, kdaj je primeren trenutek za nakup ali prodajo. To se zgodi v dveh primerih: ko graf tečajev preseka drseče povprečje od spodaj navzgor, je trenutek za nakup, ko ga preseka v obratni smeri, je čas za prodajo (slika 1.a).

Slika 1.a:
Zglajena krivulja na grafu predstavlja 200-dnevno drseče povprečje

2. **Relativna moč** (RSI – Relative Strength Index)
Kazalnik se giblje v mejah od 0 od 100. Kadar se kazalnik povzpne nad zgornjo signalno črto (vrednost 70), daje znak, da je vrednostni papir precenjen. Če tako stanje sovpada z obratom v trendu gibanja tečajev, to pomeni priložnost za prodajo. Obratno, kadar se indeks spusti pod spodnjo signalno črto (vrednost 30), pomeni da je vrednostni papir podcenjen. Skupaj z obratom gibanja tečajev je to signal za nakup delnice. Od svojega nastanka 1978 se je ta kazalnik potrdil v dobri predvidljivosti in postal eden izmed najbolj uporabljenih orodij tehnične analize (slika 1.b).



Slika 1.b: **Indeks relativne moči za graf gibanja delnice na sliki 1.a.**

4. Analiza vrednostnih papirjev z uporabo odločitvenih dreves

Naš pristop analize vrednostnih papirjev temelji na preučevanju preteklega gibanja tečajev in iskanju sorodnih ciklov s pomočjo odločitvenih dreves. Za atribute smo uporabili tečaje delnic in njihov promet v obdobju zadnjih šestih let, od januarja 1995 do decembra 2001. Za odločitveni atribut smo izbrali razreda: *kupi* in

čakaj. Na podlagi razredov odločitvenega atributa je naš cilj določiti dober trenutek za nakup. Določeni pomembni podatki, ki vplivajo na tečaje delnic in hkrati na časovni termin nakupa delnic, kot so npr. stopnja inflacije, medbančne obrestne mere, potrošniški indeksi, gospodarska rast itd, pri tehnični analizi niso zajeti.

Tehnična analiza delnic je v večini primerov kratkoročna analiza gibanja tečajev, kjer ustrezne grafične formacije obetajo določen kratkoročni trend rasti ali padanja delnice. Tako smo se tudi mi odločili za oceno tečaja po preteku pet trgovalnih dni (kratkoročno). Žal pri večini delnic tudi pet trgovalnih dni za večje spremembe tečajev ne zadostuje. Posledica je, da je takšna oblika analize smiselna le za novo ustanovljena, delno pa tudi uveljavljena tehnološka podjetja (nova ekonomija - new economy, v ZDA tečajnica NASDAQ, v Nemčiji Neuer Markt), katerih delnice precej bolj nihajo. Za analizo smo izbrali podjetje Intel s tečajnice NASDAQ (vsebovan je tudi v indeksu Dow Jones Industrial), ki simbolizira dinamično gibanje tečajev. Izračunali smo, da tečaj delnic podjetja Intel v omenjenem petdnevnem roku poraste za več kot 2,5 %, v 35 % učnih primerov (objektov). Na podlagi tega rezultata smo izbrali sledečo lestvico razredov odločitev (preglednica 1):

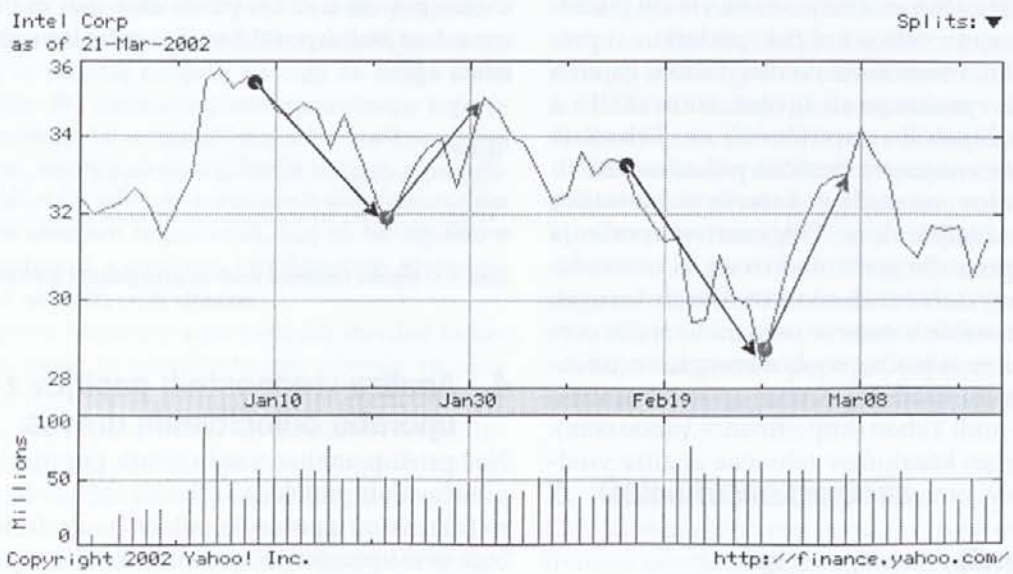
Preglednica 1: Razredi odločitev

kupi	–	porast tečaja > 2,5 %,
čakaj	–	porast tečaja <= 2,5 %

Glede na to, da je provizija pri nakupu in prodaji delnic vse manjša (zaenkrat žal le v tujini, npr. pri nemškem Comdirectu znaša ta pri nakupni vrednosti 10.000 € okoli 0,2 %), predstavlja danes porast delnice za 2,5 % minimalen zaslužek, tudi če upoštevamo najvišji razred davčne stopnje (50 %). Na primer, če delnica v roku petih dni pri vložku 10.000 € poraste za 2,5 %, odbijemo 0,4 % za provizijo in še 1,25 % za davek na dobiček, potem nam ostane 0,85 % oziroma 85 € čistega dobička. Temu sledi, da se nam vrednost 2,5 % z ekonomskega vidika ne zdi prenizka.

Na sliki 2 je lepo razvidno nihanje vrednostnega papirja Intel v prvem kvartalu 2002. Na sliki smo s puščicami označili dve podobni grafični formaciji, ki jih morda lahko napovemo. Opazimo lahko, da so trendi časovno gledano približno enako dolgi in s tega vidika je neka stopnja ponovljivosti prisotna. Težava je v tem, da gibanje tečajev nima vedno tako očitnih enakomernih trendov, kar znatno oteži analizo in napovedovanje. Z ekonomskega vidika je kratkoročno trgovanje tehnoloških delnic povsem upravičeno, saj delnica v roku 3 mesecev ni pridobila na vrednosti, z uspešnimi kratkoročnimi nakupi in prodajami pa bi lahko zaslužili okoli 20 % in več.

Podjetje Intel je eden izmed glavnih nosilcev indeksa nove ekonomije NASDAQ Composite, ki ga sestavlja 500 podjetij nove ekonomije, vsako podjetje (delnica) ima v njem določeno utež., ki je odvisna od tržne kapitalizacije (število delnic pomnožimo s trenutnim tečajem) in povprečnega dnevnega prometa na borzi. Kotira tudi v indeksu polprevodnikov SOXX (Semiconductor index), ki je tako rekoč »pojem« nove ekonomije. SOXX v bistvu združuje vse polprevodnike znotraj tečajnice NASDAQ in odločilno vpliva na njegov trend. Če primerjamo grafa delnice Intel in



Slika 2: Dinamično gibanje tečajev delnice Intel (puščice prikazujejo trende)

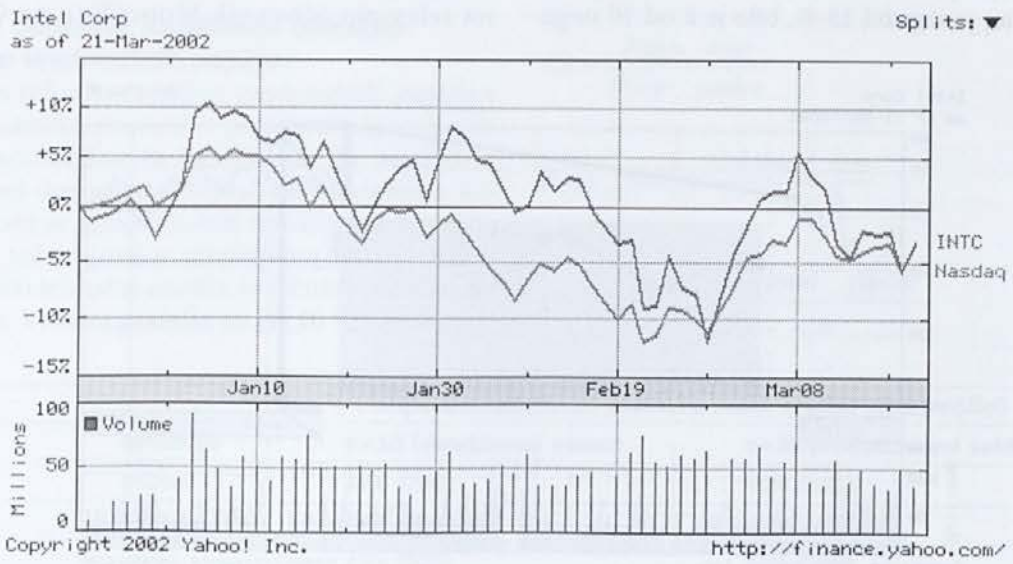
indeksa NASDAQ, vidimo, da je gibanje dokaj primerljivo, z minimalnimi odstopanji (slika 3). Američani imajo pregovor: »Nasdaq goes like Intel goes«.

Če spremljamo poslovne rezultate tehnoloških podjetij po četrtletjih, ugotovimo, da ti močno nihajo, kar seveda neposredno povzroči volatilnost vrednostnih papirjev. Naslednji aspekt, ki pripomore k volatilnosti, pa so še velika odstopanja od pričakovanih poslovnih rezultatov, tako v pozitivni kot tudi negativni obliki. Leto 1999 in 2000 so tehnološka podjetja doživljala senzacionalno rast, saj je vsakdo, npr. želel imeti računalnik, mobilni telefon, priključek na internet itd., kar se je zrcalilo v poslovnih rezultatih. Vrednostni papirji so temu primerno poskočili za nekaj 100% v pričakovanju še nadaljnje večje prodaje in rasti (npr. intel je od poletja 1998 do poletja 2000 zrasel kakšnih 300%). Da bi podjetja lahko obvladovala večja povpraševanja, so masivno večali proizvodne kapacitete in zaposlovali novo delovno silo. Čisto potiho in neopazno pa se je hkrati večala konkurenca, kar dolgoročno pomeni pritisk na stopnjo zaslužka. Konec leta 2000 je povpraševanje po produktih in storitvah upadlo, konkurenca je opravila svoje in sledil je poslovni zlom in tudi zlom borze vrednostnih papirjev (Nasdaq je izgubil 60 %, Neuer Markt pa kar 80%). Podjetja so imela in še danes imajo prevelike kapacitete in seveda preveč delovne sile, ki so jo predvsem 2001 masovno odpuščali in še dodatno bremenili že tako slabe poslovne rezultate v zadnjih dveh, treh četrtletjih.

Tehnične analize in tudi fundamentalna analiza sta bili s stališča dogodkov leta 1999 in prve polovica leta 2000 moteč faktor. Preprosto povedano, kupil si lahko poljuben vrednostni papir in počakal, da si obogatel. To prikazuje tudi statistični podatek, da je v letu 2000 v Nemčiji masivno poraslo število milijonarjev. Če pa vzamemo za primer gibanje tečajev od tega trenutka naprej, lahko ugotovimo, da brez fundamentalne in tehnične analize na borzi vrednostnih ni več mogoče uspešno poslovati. Le še nakup in prodaja v pravem trenutku prinašata dobiček in tako bo v bližnji prihodnosti tudi ostalo. Tako mora sedaj tudi mali delničar pozorno spremljati gibanje vrednostnih papirjev in poslovanje podjetij. Iz teh razlogov in trenutne situacije na trgu vrednostnih papirjev bi bila smiselna tudi uporaba inteligentnih sistemov pri kratkoročnem napovedovanju gibanja tečajev.

Izdelali smo nekaj matematičnih modelov za analizo delnic s pomočjo odločitvenih dreves, obetavnejše bomo v nadaljevanju podrobneje opisali. Vsak izmed modelov ima svoje posebnosti, poskusili pa smo pokriti vsaj osnovni del spektra tehnične analize vrednostnih papirjev. Za učno množico smo izbrali tečaje vrednostnih papirjev od januarja 1995 do decembra 1999, kar pomeni okoli 1400 tečajev (učnih objektov), za testiranje pa smo uporabili tečaje od januarja 2000 do decembra 2001, skupaj okoli 500 testnih objektov. Zaradi ustreznega preoblikovanja glede na izbrani matematični model se je število objektov spreminjalo. Tečajnica v ZDA, za razliko od ljubljanske borze, ne pozna srednjega tečaja, zato smo za analizo ter izdelavo učne in testne množice uporabili zaključni tečaj (preglednica 2).

Za testiranje postavljenih matematičnih modelov smo uporabili naslednje aplikacije za izdelavo odločitvenih dreves:



Slika 3: **Primerjava gibanja delnice Intel (INTC) in indeksa »nove ekonomije« NASDAQ**

Datum	Začetni	Najvišji	Najnižji	Zaključni	Promet
31-Dec-01	32.15	32.41	31.41	**31.45**	27,975,400
28-Dec-01	32.94	33.31	32.12	**32.24**	27,584,200
27-Dec-01	32.41	32.98	32.36	**32.67**	22,443,900
26-Dec-01	32.05	33.12	32.02	**32.29**	23,231,600
24-Dec-01	32.16	32.4	32	**32.02**	8,673,800
21-Dec-01	32.22	32.74	31.97	**32.41**	57,817,500
20-Dec-01	32.85	33.2	31.96	**31.98**	49,423,400
19-Dec-01	33.33	33.96	33.04	**33.05**	39,961,400

Preglednica 2:
Podatki o dnevnih tečajih, ki smo jih z različnimi matematičnimi modeli preoblikovali in pripravili učne in testne množice za analizo z odločitvenimi drevesi.

1. **MtDeciT** – klasično generiranje odločitvenih dreves [8]
2. **DecRain** – uporablja genetske algoritme [4]
3. **C5.0** – statistični pristop gradnje odločitvenih dreves [1,2]

MtDeciT in **DecRain** sta aplikaciji, ki sta bili v laboratoriju za načrtovanje sistemov razviti v okviru različnih projektov, **C5.0** pa je referenčna aplikacija na področju odločitvenih dreves.

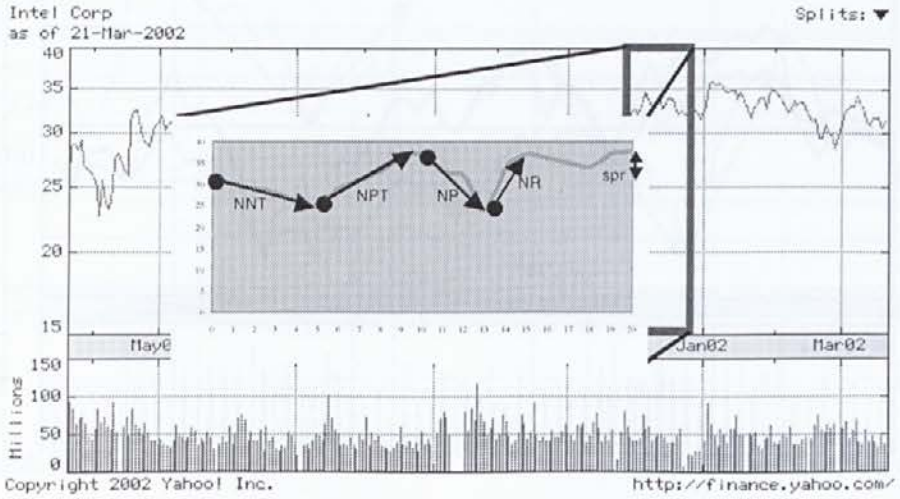
5. Matematični modeli za napovedovanje gibanja tečajev vrednostnih papirjev

5.1 Matematični model razdelitve sektorja na neformalne kazalnike

Ko poskušamo opisati dogajanje na borzi v nekem obdobju, uporabljamo določene numerične atribute. Na primer, dogajanje v dveh tednih bi lahko opisali takole: Nasdaq je izgubil 15 %, bilo je 8 od 10 negativnih trgovalnih dni in največji enkratni padec je znašal 5 %. V takšnem scenariju mnogi investitorji vidijo priložnost poceni nakupa, nekateri zadnjo priložnost za umik s trga, mi pa z zanimanjem pričakujemo, kako bo položaj ocenil naš IS. Gibanje tečajev vrednostnih papirjev smo razdelili na sektorje, za primer smo vzeli 10 in 20-dnevne intervale (prikazano na sliki 4), in za vsak sektor izračunali naslednje atribute:

- spr ... sprememba tečaja v odstotkih med prvim in zadnjim tečajem v sektorju
- NPT ... najdaljši pozitivni trend v dneh
- NNT najdaljši negativni trend v dneh
- DR število dni rasti tečajev
- NR največja neprekinjena rast v odstotkih
- NP največji neprekinjeni padec v odstotkih

Izbrali smo kazalnike, ki so enostavno izračunljivi in po našem mnenju dobro opisujejo dogajanje v določenem obdobju. To so kazalniki, ki so redno prisotni tudi pri poročilih o borznem dogajanju v časopisih ali na televiziji (dnevnik Moneyline na CNN) in so



Slika 4: **Razdelitev grafa delnice na 20-dnevne sektorje**

nekakšna osnova za »ohlapni« opis situacije na trgu vrednostnih papirjev. Razred odločitve smo podali za pet trgovalnih dni v prihodnost.

Primer 1: Gradnja objekta učne množice, pri številu vhodnih atributov n=10:

Novi desetdnevni sektor se prične z n=1 in konča pri n=10, tečaj pri n=0 sodi v predhodni sektor (predstavlja zadnji tečaj predhodnega sektorja). Potrebujemo ga, da lahko izračunamo, ali delnica prvi dan v novem sektorju raste ali pada in za spremembo v odstotkih med zadnjima tečajema v zaporednih sektorjih.

N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Tečaj	4	6	5	6	7	8	10	8	9	6	5	5	6	7	8	8

spr = tečaj(10)/tečaj(0) = 1.25
NPT = tečaj(2) do tečaj(6) = 4
NNT = tečaj (8) do tečaj (10) = 2
DR = 6
NR = tečaj(6)/tečaj(2) = 2
NP = tečaj(10)/tečaj(8) = 0.55
razred = tečaj(15)/tečaj(10) = 1.6

=> učni objekt:

spr	NPT	NNT	DR	NR	NP	razred
1.25	4	2	6	2	0.55	kupi (1.6)

Rezultati analize

Verjetnost zadetka je pri 10-dnevnem nekoliko prekašala verjetnost pri 20-dnevnem sektorju (preglednica 3), vendar rezultati v nobenem primeru ne izpolnjujejo pričakovanj. Model ne preseže verjetnosti zadetka 50 %.

5.2 Model obdelave kazalnikov tehnične analize vrednostnih papirjev

Kazalnikom tehnične analize vrednostnih papirjev smo se že nekoliko posvetili v poglavju 3, kjer smo že omenili najznačilnejša kazalnika. Za analizo z odločitvenimi drevesi smo izbrali tiste kazalnike, katerih vrednosti se gibajo znotraj omejenih intervalov, ne glede na tečaje delnice, imenujemo jih tudi oscilatorji. Kazalniki tehnične analize se računajo za vsak trgovalni dan. Vhodni podatki so pri 90 % kazalnikov le tečaji, promet z delnicami praktično ni uporabljen. Pri vsakem kazalniku moramo podati tudi obdobje za izračun. Ker nas zanimajo kratkoročni trendi, smo izbrali desetdnevni interval (n=10). Pogosto uporabljena značilna izračuna zajemata tudi 200 (dolgoročno) in 38 (srednjeročno) dnevno obdobje. Ta podatek navajamo le kot zanimivost, saj je pogosto prisoten pri borznih poročilih in je dobro poznati njegov pomen.

Ker se so si različni kazalniki med seboj zelo podobni, smo poskušali izbrati skupino kazalnikov, ki so med seboj dokaj neodvisni. Med številnimi kazalniki smo tako izbrali naslednje štiri [7]:

- kazalnik relativne moči (RSI - Relative Strength Index)
- moment (Momentum)
- Williams %R
- kazalnik pretoka denarja (MFI - Money Flow Index)

Primer 2: Gradnja objekta učne množice z izračunom kazalnikov:

RSI (1):

Kot smo že omenili v poglavju 3, se kazalnik giblje v mejah od 0 od 100. Kadar se kazalnik povzpne nad zgornjo signalno črto (vrednost 70), daje znak, da je vrednostni papir precenjen. Če tako stanje sovpada z obratom v trendu gibanja tečajev, to pomeni priložnost za prodajo. Obratno, kadar se indeks spusti pod spodnjo signalno črto (vrednost 30), pomeni da je vrednostni papir podcenjen. Skupaj z obratom gibanja tečajev je to signal za nakup delnice.

$$RSI = 100 - \frac{100}{1 + RS} \quad (1)$$

$$RS = \frac{Povp\_rast}{Povp\_padec}$$

$$Povp\_rast = \frac{Celotna\_rast}{n}$$

$$Povp\_padec = \frac{Celotni\_padec}{n}$$

		verjetnost zadetka pri analizi na testni množici (učni množici)	
delnica	**aplikacija**	**n=10 (desetdnevni sektor)**	**n=20 (dvajsetdnevni sektor)**
Intel	MtDeciT	47% (79%)	40% (70%)
	DecRain	48% (65%)	43% (65%)
	C5.0	49% (90%)	43% (86%)

Preglednica 3: **Metoda razdelitve signala na kazalnike**

Zadnji	Sprememba	Porast	Padec	Povprečna_rast		Povprečni_padec	RS	RSI
	46,125							
1	47,125	1	1	0				
2	46,4375	-0,6875	0	0,6875				
3	46,9375	0,5	0,5	0				
4	44,9375	-2	0	2				
5	44,25	-0,6875	0	0,6875				
6	44,625	0,375	0,375	0				
7	45,75	1,125	1,125	0				
8	47,8125	2,0625	2,0625	0				
9	47,5625	-0,25	0	0,25				
10	47	-0,5625	0	0,5625	0,5063	0,4188	0,8272	45,2703
11	44,5625	-2,4375	0	2,4375	0,4063	0,6625	1,6308	61,9883
12	46,3125	1,75	1,75	0	0,5813	0,5938	1,0215	50,5319
13	47,6875	1,375	1,375	0	0,6688	0,5938	0,8879	47,0297
14	46,6875	-1	0	1	0,6688	0,4938	0,7383	42,4731
15	45,6875	-1	0	1	0,6688	0,5250	0,7850	43,9791

Preglednica 4: **Zgled za izračun RSI pri n=10**

Izračun kazalnika RSI je nekoliko zapleten, zato podajamo še primer izračuna in sicer za desetdnevni časovni interval (preglednica 4).

Moment

Moment (2) poskuša ugotoviti trend, še preden ta postane očiten. Vzpenjajoč moment daje znak bikovskega, padajoč pa medvedjega trenda.

$$Moment(x,n) = \frac{zadnji(x)}{zadnji(x-n)} * 100 \qquad (2)$$

Williams %R

Williamsov kazalnik (3) kaže mesto, v intervalu med maksimalnim in minimalnim tečajem, kjer je trenutno tečaj delnice. Kazalnik služi kot pomoč pri ocenjevanju trenda, saj nam pove, kako se sklepajo posli. Če je vrednost blizu spodnjega roba, pomeni, da so se posli sklepali blizu minimalnega tečaja. Če se krivulja prične obračati proti sredini pomeni, da prodajni pritisk slabi in obstaja možnost preobrata. Obratno velja za rast tečajev.

$$will(x) = -100 * \frac{max\_v\_času\_n - zadnji(x)}{max\_v\_času\_n - min\_v\_času\_n} \qquad (3)$$

MFI

Kazalnik pretoka denarja meri moč pretoka denarja v in iz delnic. Po izračunu je podoben kazalniku RSI, z razliko da upoštevamo promet. MFI temelji na ideji, da se v vrednostni papir investira, če je zadnji tečaj med najvišjimi tečaji ta dan, v primeru, ko je zadnji tečaj v bližini dnevnega minimuma, se prodaja. Če je zadnji tečaj tekočega trgovalnega dne višji od tečaja predhodnega trgovalnega dneva, govorimo o pozitivnem, sicer pa o negativnem pretoku denarja.

$$Povp\_cena(x) = \frac{max(x) + min(x) + zadnji(x)}{3}$$

$$Pretok\_denarja = Povp\_cena * Promet$$

Sedaj je potrebno ločiti dneve z pozitivnim in negativnim pretokom denarja v določeni časovni periodi. Nato za pozitivne in negativne dneve sumiramo pretok denarja in izračunamo MFI.

$$MFI = 100 - \frac{100}{1 + MR}$$

$$MR = \frac{Pozitivni\_pretok\_denarja}{Negativni\_pretok\_denarja} \qquad (4)$$

Izračun kazalnika MFI je pravtako malce zapleten, zato podajamo še primer izračuna in sicer za desetdnevni časovni interval (preglednica 5).

	Min	Max	Zadnji	Promet	Povprečna cena	Pretok denarja	Porast	Padec
	45	46.5	46.125	30003	45.875	1376387.625		
1	47.25	48.125	47.125	20000	47.5	950000	950000	0
2	46	47	46.4375	32424	46.47916667	1507040.5	0	1507040.5
3	46.5	47.9375	46.9375	64343	47.125	3032163.875	3032163.875	0
4	44.6875	45	44.9375	24242	44.875	1087859.75	0	1087859.75
5	44	44.5625	44.25	57575	44.27083333	2548893.229	0	2548893.229
6	44.5	45.25	44.625	42342	44.79166667	1896568.75	1896568.75	0
7	45.5	46	45.75	43242	45.75	1978321.5	1978321.5	0
8	47	48	47.8125	34343	47.60416667	1634869.896	1634869.896	0
9	47.5	48.25	47.5625	45632	47.77083333	2179878.667	2179878.667	0
10	47	47.75	47	34355	47.25	1623273.75	0	1623273.75
11	44	45	44.5625	56576	44.52083333	2518810.667	0	2518810.667
12	46	47.25	46.3125	47699	46.52083333	2218997.229	2218997.229	0
13	47.5	48	47.6875	45457	47.72916667	2169624.729	2169624.729	0
14	46.5	47	46.6875	34541	46.72916667	1614072.146	0	1614072.146
15	45	46	45.6875	27999	45.5625	1275704.438	0	1275704.438

Preglednica 5: **Zgled za izračun MFI pri n=10**

Povprečna_rast	Povprečni_padec	MR	MFI
11671802.6875	6767067.2292	0.5798	36.7000
10721802.6875	9285877.8958	0.8661	46.4116
12940799.9167	7778837.3958	0.6011	37.5433
12078260.7708	7778837.3958	0.6440	39.1741
12078260.7708	8305049.7917	0.6876	40.7444
12078260.7708	7031861.0000	0.5822	36.7965

Rezultati analize

Tudi pri tem modelu nismo prišli do boljših rezultatov glede na predhodni model (preglednica 6). Za izračun kazalnikov smo uporabili le kratkoročno desetdnevno časovno obdobje. Metoda ne preseže verjetnost zadetka 50 %, eden razlogov je pravgotovo, da je vhodnih atributov premalo.

		verjetnost zadetka pri analizi na testni množici (učni množici)
Delnica	**Aplikacija**	**n=10(desetdnevno obdobje)**
Intel	MtDeciT	44% (70%)
	DecRain	44% (69%)
	C5.0	42% (88%)

Preglednica 6: **Metoda izračuna kazalnikov tehnične analize**

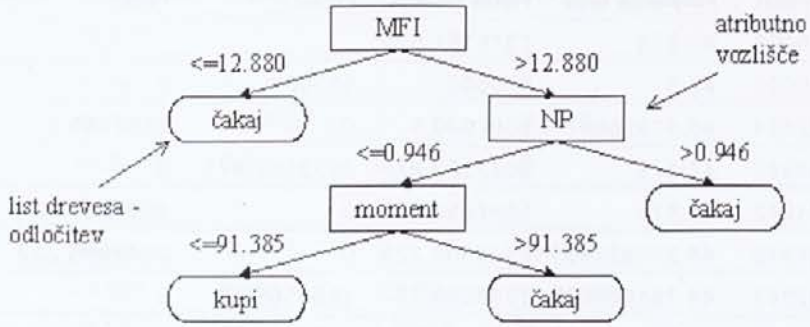
5.3 Model združitve kazalnikov

Matematična modela izračuna neformalnih kazalnikov in kazalnikov tehnične analize imata vsak posebej premalo vhodnih atributov za kvalitetno izdelavo odločitvenega drevesa, zato smo se odločili atribute obeh modelov združiti in ponovno generirati odločitveno drevo. Sedaj smo dobili precej boljše rezultate, verjetnost zadetka je prvič dosegla in presegla stopnjo 60 %. Še posebej pa je izstopala verjetnost zadetka odločitve **kupi** s 75 % (preglednica 7).

		verjetnost zadetka pri analizi na testni množici (učni množici)
Delnica	**Aplikacija**	**n=10(desetdnevno obdobje)**
Intel	MtDeciT	57% (69%)
	DecRain	62% (65%)
	C5.0	53% (73%)

Preglednica 7:
Združeni model neformalnih kazalnikov in kazalnikov tehnične analize

Na sliki 5 je predstavljeno odločitveno drevo, ki je dalo najboljše rezultate, poleg tega je tudi zelo enostavno saj ima le 3 atributna vozlišča in 4 liste z odločitvami. Odločitveno drevo sestavljata dva atributa kazalnikov tehnične analize (MFI in moment) in atribut iz skupine neformalnih kazalnikov (NP). Glede na število vhodnih atributov in učnih objektov



Slika 5: **Odločitveno drevo pri modelu združenih kazalnikov**

vidimo, da je zgrajeno odločitveno drevo izredno majhno. Če imamo na voljo dve odločitveni drevesi z enako kvaliteto odločanja, je ponavadi boljše tisto, ki je manjše (pravilo Ockhamove britve). Malo število atributnih vozlišč in listov v drevesu pomeni večjo splošnost in s tem posledično boljše rezultate odločanja na še novih nepoznanih primerih [12].

Atributa MFI in NP imata za pogoje pretežno robne vrednosti (MFI le redko pade pod 12 in NP redko pade pod 0.94), tako da odločitev o nakupu predstavlja predvsem atribut moment. Če je vrednost momenta okoli 90, se delnico Intela običajno splača kupiti. Na sliki 6 vidimo, da je odločitev o nakupu pri dani vrednosti momenta dokaj utemeljena.

5.4 Kaj bi nastalo iz 100.000$?

Da bi v praksi preizkusili v prejšnjem poglavju zapisane trditve in preverili uspešnost odločitvenega drevesa (slika 5), smo se odločili, da preizkusimo, kaj bi nastalo iz našega denarja v obdobju trgovanja od 3.1.2000 do 31.12.2001 z delnico Intel. V tem obdobju je Intel izgubil 27 % na vrednosti delnice, startal je s tečajem 43,33$, končal pa na 31,45$, maksimum je znašal 74,50$, minimum pa 19,28$. Z dolgoročnim nakupom delnice bi v tej časovni periodi ustvarili precejšnjo izgubo, preglednica 8 pa prikazuje kako bi se odrezal naš IS.

To obdobje je bilo glede na matematični model združenih kazalnikov razdeljeno na 49 desetdnevnih sektorjev, od tega je bilo 12 napovedi kupi. V preglednici 8 predstavljamo le rezultate napovedi **kupi** v tem obdobju, kakšen je bil dejanski rezultat in koliko denarja smo pridobili oziroma izgubili ter datum napovedi.

Slika 8:
Gibanje delnic Intel in momenta v obdobju treh let (1999 - 2001)

datum napovedi nakupa	nakupna cena	prodajna cena	pravilni rezultatv odstotkih		stanje
07.08.2000	62,75$	66,74$	kupi	6,3%	106.300$
19.09.2000	60,19$	43,18$	čakaj	-28,3%	76.217$
03.10.2000	40,19$	37,45$	čakaj	-6,8%	71.034$
17.10.2000	36,08$	41,87$	kupi	16,1%	82.470$
14.11.2000	40,83$	42,52$	kupi	4,1%	85.851$
13.12.2000	35,41$	31,86$	čakaj	-10,0%	77.265$
28.12.2000	30,86$	31,98$	kupi	3,6%	80.046$
27.02.2001	28.94$	31,44$	kupi	8,6%	86.930$
10.04.2001	24,72$	31,22$	kupi	26,3%	109.792$
21.06.2001	27.23$	29,60$	kupi	8,7%	119.343$
17.08.2001	28,05$	29,06$	kupi	3,6%	123.639$
21.09.2001	20,42$	21,94$	kupi	5,9%	**130.933$**

Preglednica 8: **Kaj bi nastalo iz 100.000$ v obdobju med 1.1.2000 in 31.12.2001**

Med 12 napovedanimi nakupi je bilo 9 pravilnih oziroma verjetnost zadetka v primeru napovedi **kupi** znaša 75 %. Znesek, ki smo ga imeli 31.12.2001 je znašal **130.933$** ali okoli 31% dobička glede na 1.1.2000.

6. Razprava

Za vhodne atribute matematičnega modela smo izbrali neformalne kazalnike, s katerimi se vsaj pretežno opisuje dogajanje na borzi in kazalnike tehnične analize. Vsak kazalnik ima svoj pomen in območje vrednosti, če sedaj pogledamo izdelano odločitveno drevo, lahko sami ocenimo ali je potek odločitve glede na vhodne atribute smiseln. Primerjava učnih in testnih objektov je pokazala, da med objekti ni bilo velike stopnje ponovljivosti, a kljub vsemu veljajo določene tržne zakonitosti (npr. da po večjem padcu sledi rast in obratno ali pa nizke vrednosti kazalnikov prinesejo preobrat).

V praktičnem primeru, kaj bi nastalo iz 100.000$, se je izkazalo, da IS pri nadpovprečnem nihanju tečajev dosega dobre rezultate. V primeru daljšega obdobja (leto dni) rasti bi gotovo iztržili dobiček, v primeru daljšega obdobja padanja pa bi bila izguba manjša. IS se je izkazal kot podpora pri svetovanju nakupa v pravem trenutku, čeprav verjetnost naključja nikoli ne bo mogoče povsem izključiti.

7. Sklep

Naš IS pri napovedovanju gibanja tečajev vrednostnih papirjev sicer ne nudi konstantno dobrih rezultatov nad 60 %, a je kljub vsemu uporaben pri odločanju o pravem trenutku nakupa. V praksi se je izkazalo, da bi IS trgoval bolje od povprečnega investitorja v obdobju, ko se tečaji vrednostnih papirjev niso spremenili oziroma so padali.

Pri analizi vrednostnih papirjev nismo sodelovali z strokovnjaki neposredno, uporabili smo le priporočila in izkušnje ljudi, ki na tem področju delajo in raziskujejo. Upoštevali smo njihova mnenja in priporočila in uporabili kazalnike, ki imajo pri analizi vrednostnih papirjev določen pomen. Kar se tiče trgovanja z vrednostnimi papirji in tudi same analize, prevzemata internet in televizija poglavitno vlogo. Težava je v tem, da se mnenja strokovnjakov med seboj močno razlikujejo, vsak izmed njih ravna in se opredeljuje po lastnih preferenčnih ekonomskih kazalnikih, glede na izkušnje ter dogajanje v preteklosti.

Naše naslednje dejanje pri poskusu izboljšave kvalitete odločanja bo vpeljava teorije kaosa. Na primer, uporaba drsečih povprečij za zgladitev ostrin in šumov, ki ovirajo razpoznavanja trendov. Krivuljo drsečega povprečja bomo nadomestili z realnimi tečaji vrednostnega papirja. Vhod matematičnega modela tako ne bodo tečaji, ampak funkcijske vrednosti krivulje drsečega povprečja.

Literatura

[1] J. R. Quinlan. Induction of Decision Trees, Machine learning, No. 1, pp. 81-106, 1986.

[2] J. R. Quinlan J R, C4.5: Programs for Machine Learning, Morgan Kaufmann publishers, San Mateo, CA, 1993.

[3] S. Hleb Babic, P. Kokol, V. Podgorelec, M. Zorman, M. Sprogar, M. Molan Stiglic. The Art of Building Decision Trees, Journal of Medical Systems, Plenum Press, Volume 24, No. 1, pp. 43, 2000.

[4] M. Šprogar, V. Podgorelec, P. Kokol. Odločitvena drevesa in sistemi z večdimenzionalnimi rešitvami, Uporabna informatika, letnik 8, st. 2, str. 79-86, 2000.

[5] M. Čas, T. Kotar. Borzni izrazi, Kapital, Maribor, 1994.

[6] A. Jerovšek. Delnice: analize, strategije, špekulacije, davki, Edicija FIRST, 1991.

[7] http://www.equis.com/free/taaz/intindicators.html

[8] M. Zorman, Š. Hleb Babič, M. Šprogar. Advanced tool for building decision trees MtDeciT 2.0. V: KOKOL, Peter (ur.), Welzer-Družovec, Tatjana (ur.), Arabnia, Hamid R. (ur.). International conference on artificial intelligence, June 28 - July 1, 1999, Las Vegas, Nevada, USA. Las Vegas: CSREA, 1999, zv. 1, str. 315-318.

[9] N. Bada. Predictions of the Tokyo Stock Exchange Prices Indexes (TOPIX) by Techniqus of Computational Intelligence, Proceeding of the International ICSC Congress, CIMA'99, Rochester, N. Y. USA, June 22-25, 1999.

[10] R. Suchar, I. Ciocoiu, A. Brezulianu, C. Bonciu. Stock Market Predictions Using a Hybrid Neuro-Genetic Approach, Proceeding of the International ICSC Congress, CIMA'99, Rochester, N. Y. USA, June 22-25, 1999.

[11] R. A. Pearson. How to gain?/lose? on the stock market – datamining the ASX, AISAT'2000, Hobart, Tasmania, Australia, 17-20 December, 2000.

[12] Vili Podgorelec, Peter Kokol (mentor), Bruno Stiglic (somentor): Oblikovanje inteligentnih sistemov in odkrivanje znanja z avtomatskim programiranjem, doktorska disertacija, Univerza v Mariboru, oktober, 2001.

Alojz Tapajner je diplomiral s področja računalništva in informatike na Univerzi v Mariboru, Fakulteti za elektrotehniko, računalništvo in informatiko. Njegova raziskovalna področja so inteligentni sistemi in posebno njihova uporaba na finančnih področjih. Sodeluje pri raziskovalnih projektih Laboratorija za načrtovanje sistemov in ima objavljenih več člankov na domačih in mednarodnih konferencah. Bil je tudi član organizacijskih odborov konferenc MIE'99 in CBMS'02.

Dr. Peter Kokol je diplomiral s področja elektrotehnike in doktoriral s področja računalništva, oboje na Univerzi v Mariboru. Njegova raziskovalna področja so inteligentni sistemi, teorija sistemov, teorija kaosa in kvaliteta programske opreme. Je vodja nacionalnih in mednarodnih projektov iz imenovanih področij. Njegova bibliografija obsega več kot 400 enot, od tega več kot 50 originalnih znanstvenih člankov. Bil je predsednik organizacijskih in programskih odborov več svetovnih konferenc. Je tudi svetovalec pri projektih svetovne banke.

GRADNJA HIERARHIČNE STRUKTURE KONCEPTOV IZ NESTRUKTURIRANIH TEKSTOVNIH DOKUMENTOV

Robert T. Leskovar, József Györkös, Ivan Rozman
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Inštitut za informatiko, Smetanova ulica 17, Maribor
RLeskovar@uni-mb.si

Povzetek

Metode za iskanje po obsežnih bazah dokumentov in za klasificiranje dokumentov pogosto uporabljajo za učinkovitejše delovanje vnaprej pripravljene opise področnih kategorij. V prispevku je predstavljena izvirna metoda za samodejno gradnjo hierarhične strukture vsebinskih konceptov, ki jih določi z analizo množice dokumentov. Struktura omogoča učinkovito brskanje in iskanje po vsebinskih konceptih ter zelo poenostavi oblikovanje opisa področnih kategorij za poljubno množico dokumentov. Metoda je zasnovana neodvisno od jezika, v katerem je zapisana vsebina dokumentov. Predstavljeni sta tudi razširitvi za semantično povezovanje konceptov in za pospešitev delovanja metode.

Abstract

CONSTRUCTION OF A HIERARCHICAL CONCEPT STRUCTURE FROM UNSTRUCTURED TEXT DOCUMENTS

Document retrieval methods, implemented for large document collections and document classification methods often use predefined descriptions of subject categories to improve their efficiency. In the article the original method for automated construction of a hierarchical concept structure is presented. The structure is built through an analysis of the documents in a collection. Such a structure makes efficient concept browsing and searching feasible and simplifies defining of subject category descriptions for arbitrary document collections. The method is independent of language used in documents. Two improvements are introduced as well, aiming at connecting the concepts semantically and improving a processing capacity of the method.

■ ■ ■

1. Uvod

1.1 Pridobivanje informacij

Količina informacij, zapisanih v nestrukturirani ali delno strukturirani tekstovni elektronski obliki (nadalje: informacij), s katerimi se kot informacijska družba srečujemo v okoljih delovnih organizacij in v spletu, je vse manj obvladljiva in zahteva nove metode iskanja in predstavitve želenih informacij. Znanje, zapisano s temi informacijami, ima visoko vrednost in njegovo učinkovito upravljanje je ključnega pomena za kakovost delovnih procesov ter posledično konkurenčnost organizacij, tako gospodarskih kot negospodarskih.

Metode pridobivanja informacij (ang. 'information retrieval') so se v zadnjih dvajsetih letih razvile od preprostega iskanja dokumentov do zahtevnejšega iskanja, klasificiranja, filtriranja in grupiranja dokumentov. Področje pridobivanja informacij ne zajema le sintaktičnega vidika dokumentov, ki je predmet preprostega iskanja, ampak tudi na semantični in pragmatični vidik dokumentov. *Preprosto iskanje* (v literaturi je običajno omenjeno kot 'prosto iskanje po tekstu') na podlagi uporabnikovega povpraševanja vrne seznam dokumentov, ki vsebujejo natančno takšne izraze, kot so bili zapisani v povpraševanju. Pomanjkljivosti, ki jih vsebuje ta postopek, so neprepoznavanje večpomenskih besed, sinonimov, izvorov besed, sklanjatev in spregatev. Učinkovitost iskanja je temu primerno nizka.

Osnovne veje področja pridobivanja informacij so danes:

- *Iskanje dokumentov* (najpogosteje ga imenujemo kar 'pridobivanje informacij') temelji predvsem na enem od štirih modelov ter njihovih variantah – verjetnostnem, Boolovem modelu na podlagi mehke logike in vektorskem. Kot rezultat procesa dobimo iz baze dokumentov seznam dokumentov, ki so urejeni po stopnji ustreznosti glede na povpraševanje. Več o posameznih modelih je na voljo v [Yates, 99].
- *Klasificiranje dokumentov* (v literaturi tudi kategoriziranje, ang. 'classification' oz. 'categorization') izvaja razporejanje dokumentov, ki se nahajajo v skupni bazi, v vnaprej določeno množico področnih kategorij. Razporejanje temelji na vsebini dokumentov in ne na metainformacijah, ki jih lahko dokumenti vsebujejo. Zato je razporejanje nedeterminističen proces.

- *Filtriranje dokumentov* je oblika klasificiranja, ki razporeja dinamični tok dokumentov v vnaprej določene kategorije, najpogosteje glede na profil uporabnika, v katerem so v obliki pravil zapisani njegovi izbori. Primeri so sistemi za filtriranje novic, reklamnih sporočil ali Usenet novic.
- *Grupiranje dokumentov* (v literaturi tudi grozdenje ali združevanje, ang. 'clustering') združuje dokumente v grupe na podlagi določenih skupnih značilnosti.

Rezultati naše raziskave predstavljajo doprinos iskanju in klasificiranju dokumentov. V nadaljevanju bomo dodatno utemeljili potrebo po tovrstnih raziskavah.

1.2 Ovire za učinkovito pridobivanje želenih informacij in motivacija za raziskavo

Pri interakciji med človekom in informacijskim sistemom prihaja do več ovir, ki otežujejo učinkovito pridobivanje želenih informacij. Ena od ovir je nezmožnost uporabnikov, da bi oblikovali *ustrezno* povpraševanje. Pri iskanju v spletu je posledica tega ogromna množica vrnjenih dokumentov, med katerimi uporabniki ne najdejo želenih. Študija rabe spletnega iskalnika Excite je pokazala, da so povpraševanja v povprečju krajša od treh besed [Jansen, 98]. Osnovna razloga za to sta, da obstajajo med različnimi avtorji dokumentov razlike v rabi jezika, ki so pogojene med drugim z njihovo izobrazbo, kulturo, razgledanostjo in namenom pisanja, in da uporabniki, ki niso strokovnjaki računalniške stroke, ne poznajo principov, po katerih delujejo metode za iskanje informacij. K reševanju tega problema so se usmerile tehnike za samodejno razširjanje povpraševanj z ustreznejšimi izrazi in za usmerjanje povpraševanj na podlagi vnaprej zgrajenih seznamov izrazov, ki jih uporablja določeno vsebinsko področje (imenujemo jih vertikalni tezavri), vendar povzroči večje število izrazov pogosto tudi večji obseg pridobljenih dokumentov, tako ustreznih kot neustreznih.

Druga ovira za uspešno iskanje informacij se pojavi, če naša informacijska potreba ni jasno definirana. V takšnih primerih se raje lotimo *brskanja* po katalogih informacij (če so na voljo), kjer so dokumenti razvrščeni v področne kategorije. To nam omogoča, da pridemo prek kategorij do želenih informacij, ali da odkrijemo informacije, ki bi nas utegnile zanimati.

Za klasificiranje dokumentov v kategorije skrbi klasifikator – sistem, ki skuša s pomočjo različnih metod ugotoviti, v katero kategorijo (ali več kategorij) je potrebno dokument uvrstiti glede na njegovo vsebino. Najpogosteje so v rabi metode strojnega učenja, s katerimi klasifikator z 'opazovanjem' značilnosti množice dokumentov, ki jih je poprej klasificiral v vnaprej določeno množico kategorij strokovnjak, določajo uvrstitev v te kategorije tudi za ostale dokumente. Na mnogih področjih so se zato oblikovale taksonomije kategorij. V spletu so to splošnonamenski katalogi, kot sta Yahoo!-jev in Infoseek-ov, ki so osnova večini iskalnikov, ameriška Kongresna knjižnica ima sistem oznak Library of Congress Subject Headings (LCSH), v mnogih knjižnicah se uporablja Deweyeva decimalna klasifikacija (DDC), v medicinski domeni uporabljajo množico medicinskih oznak Medical Subject Headings (MeSH) itn..

Najnatančneje lahko klasificiramo dokumente, če so v njih navedeni metapodatki - klasifikacijske oznake ali ključne besede. Kadar niso, jih lahko doda urednik ali katalogizator, ki pozna oziroma preuči vsebino, vendar je to drag in dolgotrajen proces. Na človeško izbiro oznak in ključnih besed vpliva tudi več dejavnikov - predvsem izkušnje, razgledanost, izobrazba in konsistentnost pri uporabi oznak. Raziskave konsistentnosti pri izbiri oznak za opis vsebine določenega dokumenta so pokazale, da se oznake, ki jih izbereta dva različna profesionalna katalogizatorja, razlikujejo v povprečju za 20 odstotkov [Harman, 95].

Našo raziskavo smo usmerili v razvoj metode, ki omogoča, da lahko dobi uporabnik – iskalec informacij – hiter, strukturiran pregled vsebine množice dokumentov, ko ni na voljo vnaprej pripravljena taksonomija kategorij, v katere bi jih lahko klasificirali, in ko dokumenti niso opremljeni z metapodatki. V ta namen pretvori metoda vsebino dokumentov v hierarhično strukturo, v kateri se nahajajo med seboj povezani vsebinski koncepti, kar omogoča učinkovito brskanje in iskanje želenih informacij in predstavlja osnovo za gradnjo taksonomije kategorij. Metodo smo zasnovali neodvisno od jezika, v katerem so dokumenti. Z razširitvami, prilagojenimi določenemu jeziku, lahko še dodatno povečamo njeno učinkovitost.

1.3 Definicije

Najprej bomo natančneje definirali osnovne termine, ki jih uporabljamo v prispevku. Termina 'dokument' in 'izraz' sta privzeta termina iz področja pridobivanja informacij.

Dokument pomeni določeno tekstovno enoto, npr. naslov, povzetek, odstavek, prispevek, poglavje. Množica dokumentov je množica tekstovnih enot.

Izraz je ključna beseda (lahko besedna zveza), ki sodeluje pri opisu vsebine dokumentov. Za ključne besede izberemo predvsem tiste, ki nosijo večji pomen (predvsem samostalniške besedne oblike).

Koncept imenujemo množico izrazov, ki soodvisno nastopajo pri opisu določenih vsebinskih delov. Samodejno prepoznavanje konceptov temelji na dejstvu, da pri opisovanju podobnih stvari uporabljamo podobne besede. Koncepte lahko natančneje določimo ob primerjavi več vsebinsko sorodnih dokumentov.

2. Sorodna dela

Forsyth in Rada sta v svojem delu določala splošnost in specifičnost izrazov na podlagi števila dokumentov, v katerih se pojavljajo [Forsyth, 86]. Uporabila sta predpostavko, da je izraz bolj splošen, če se pojavi v več dokumentih. Na podlagi tega sta zgradila manjši večnivojski graf, ki je predstavljal relacije med izrazi.

Woods je v svojem delu uporabil analizo angleških fraz v povezavi z veliko bazo znanja, na podlagi česar je organiziral izraze v hierarhije konceptov [Woods, 97]. Uspeh tehnike je temeljil na bogati bazi morfološkega znanja, s katerim je identificiral komponente fraz. Hierarhijo konceptov je uporabil za samodejno razširjanje povpraševanj pri iskanju dokumentov.

Sanderson in Croft sta se lotila gradnje hierarhije konceptov iz dokumentov, ki so bili pridobljeni na podlagi povpraševanja, s primerjanjem medsebojne vsebovanosti izrazov [Sanderson, 99]. Iz množice dokumentov sta najprej s pomočjo določenega povpraševanja pridobila tiste dokumente, ki so vsebinsko vezani na povpraševanje. Množico dokumentov so zato že sestavljali dokumenti, ki uporabljajo izraze na podoben način. Množico izrazov, ki bodo vsebovani v hierarhiji, sta pridobila iz izrazov razširjenega povpraševanja in izrazov, ki se pojavljajo v več kot 10% v pridobljenih dokumentih glede na celotno bazo. Izraze sta nato primerjala glede na relacijo 'vsebovanje' (ang. 'subsumption'): izraz A vsebuje izraz B, če nastopa v več kot 80% dokumentov, v katerih nastopa izraz B. Glede na medsebojno vsebovanje sta izraze uredila v hierarhijo, pri čemer sta izločila izraze z nizko frekvenco pojavljanja. Izrazi v njuni hierarhiji lahko imajo več staršev.

Yang in Lee pristopata h gradnji hierarhije z metodo nenadzorovanega učenja nevronskih mrež, ki se imenuje *samoorganizirajoče mape* [Yang, 00]. Avtorja sta z učnimi primerki dokumentov vzpostavila dve mapi, mapo grup dokumentov in mapo grup izrazov. Dominantne nevrone v mapi grup dokumentov (ki predstavljajo določeno grupo dokumentov) sta vzela kot centroide nadgrup, ki predstavljajo bolj splošno kategorijo. Izraze, ki so povezani z istoležečim nevronom v mapi grup izrazov, sta nato uporabila kot izraze, ki opisujejo kategorijo. Hierarhijo kategorij sta zatem gradila z iskanjem korelacij med nevroni v obeh mapah. Rezultati so žal predstavljeni s kitajskimi pismenkami (avtorji so testirali pristop na množici dokumentov v kitajščini), zato jih ne moremo ovrednotiti. Avtorja omenjata nujnost zmanjšanja dimenzij vhodnih podatkov, vendar tega postopka nista opisala. Iz prispevka tudi niso razvidni načini za določanje začetne velikosti samoorganizirajoče mreže, topologije mreže (kar lahko naredimo le empirično) in faktorja učenja, ki so pomembni parametri za uspešno delovanje samoorganizacije.

Pričujoče delo se od omenjenih razlikuje v bolj univerzalnem načinu za določanje splošnosti izrazov, ki upošteva za vsak izraz število in moč njegovih korelacij, in v načinu gradnje hierarhične strukture konceptov, ki omogoča natančnejše določanje konceptov. Za gradnjo strukture ne potrebujemo pripravljenih učnih primerkov. Struktura konceptov, ki jo zgradi v nadaljevanju predstavljena metoda, je sestavljena iz *več hierarhij*, znotraj katerih so koncepti jasno izraženi glede na dokumente, v katerih se pojavljajo, med hierarhijami pa so šibko povezani, kar omogoča hiter pregled različnih delov vsebine celotne množice dokumentov. Vsebina množice dokumentov je lahko raznovrstna.

3. Metoda za gradnjo hierahične strukture konceptov

Med opisom metode bomo predstavljali njeno delovanje na majhnem, preglednem primeru – tekstu desetih naslovov prispevkov, od katerih jih šest opisuje pridobivanje informacij in štirje aplikacije teorije grafov.

Delovanje metode smo testirali na bazah dokumentov velikosti od 10 do 300 dokumentov, ki smo jih tvorili iz standardnih baz dokumentov TIME (novice iz vsega sveta, objavljene v reviji TIME leta 1963), CRAN (povzetki člankov s področja aerodinamike) in na bazi 110 dokumentov, ki govorijo o terorističnih napadih in njihovem ozadju.

Za predstavitev konceptov v zgoščeni obliki smo izbrali *hierarhično drevesno strukturo* – neusmerjen, necikličen graf, v katerem lahko ima vsak otrok le enega starša. Struktura konceptov je sestavljena iz več dreves (hierarhij), ki vsebujejo koncepte. Koncepti znotraj hierarhij so močno, med hierarhijami pa šibko povezani. Koreni hierarhij so izrazi, ki so v konceptih hierarhij najmočneje zastopani.

Kot model za predstavitev vsebovanosti izrazov v dokumentih smo izbrali *model vektorskega prostora* [Salton, 71], ki je v praksi najbolj uporabljan model, saj omogoča učinkovito obdelavo s statističnimi algoritmi in algoritmi strojnega učenja, ki izvajajo združevanje, klasificiranje, analizo glavnih komponent in druge vrste obdelav.

V tem modelu so izrazi predstavljeni z vrsticami in dokumenti s stolpci matrike. Elementi matrike so uteži, ki jih priredimo posameznemu izrazu za določen dokument.

Metoda za samodejno gradnjo hierarhične strukture konceptov zajema štiri osnovne korake, ki jih bomo podrobneje predstavili v nadaljevanju:

1. Gradnja matrike izrazov ter dokumentov.
2. Računanje matrike korelativnosti izrazov.
3. Računanje globalnih korelacijskih stopenj in iskanje korenov hierarhij.
4. Gradnja hierarhične strukture.

1. Gradnja matrike izrazov ter dokumentov

Ta korak je običajen začetni korak v pridobivanju informacij in zajema večino naslednjih aktivnosti:

(a) izbor izrazov iz baze dokumentov,
(b) krnjenje izrazov,
(c) uteževanje in normaliziranje izbranih izrazov ter
(d) združevanje izrazov, predstavljenih z enakimi vektorji.

V prvi aktivnosti iz množice vseh izrazov, ki nastopajo v bazi dokumentov, *izločimo* blokirane izraze. V splošnem lahko izločimo pomensko šibke besede, ki se zelo pogosto pojavljajo v dokumentih in jih določimo s seznamom blokiranih besed (npr. vezniki in števniki) in pomensko šibke besede, ki jih določimo s pomočjo slovarja jezika (obdržimo npr. samo samostalniške besedne oblike). V sklopu te aktivnosti izločimo samodejno tudi izraze z zelo nizko frekvenco pojavljanja v bazi dokumentov, saj njihov delež pri preglednem opisu vsebine dokumentov ni pomemben (imajo nizko funkcionalno vrednost), obenem pa s tem precej zmanjšamo dimenzije matrike izrazov in dokumentov. Frekvenčni prag je odvisen tudi od tega, koliko dokumentov je v bazi in koliko specifičnih izrazov želimo vstaviti v hierarhično strukturo. Obdržanim izrazom nato priredimo frekvence, s katerimi nastopajo v določenem dokumentu.

Po predhodni aktivnosti lahko izraze tudi *krnimo*, torej izluščimo korene izrazov z odstranjevanjem predpon in predvsem pripon. Študija uporabe krnjenja v angleščini (kjer se večinoma uporablja Porterjev algoritem [Porter, 80; Porter, 01]) ni pokazala, da bi leto pri pridobivanju informacij prispevalo k natančnosti [Frakes, 92]. Nedvomno ima krnjenje večji pomen za morfološko kompleksne jezike, kot je slovenščina. Temu primerno so kompleksni tudi algoritmi, ki se lotevajo krnjenja v teh jezikih [Popovič, 91; Dimec, 99]. V naši raziskavi krnjenja nismo uporabili, menimo pa, da velja pri aplikaciji metode za določen jezik razmisliti o njegovi uporabi.

Uteževanje izrazov priredi elementu a_{ij} v matriki izrazov in dokumentov A določeno vrednost. Vrednost je odvisna od pomena izraza i za dokument j (temu faktorju pravimo *lokalna utež*) in pomena izraza i za celotno bazo dokumentov (faktor imenujemo *globalna utež*). Obstaja več načinov za računanje lokalnih in globalnih uteži, ki so bili razviti empirično

Dokumenti (naslovi prispevkov)

D1: CAFE: A Conceptual Model for Managing Information in Electronic Mail
D2: Agent-based approach for information gathering
D3: Dunhuang Frescoes retrieval based on similarity calculation
D4: Information Retrieval on the Web
D5: The Retrieval of Document Images: A Brief Survey
D6: Survey: Introduction to Content-Based Image Retrieval
D7: Graph Visualization and Navigation in Information Visualization: A Survey
D8: Floorplan generation and area optimization using AND-OR graph search
D9: Visual programming with graph rewriting systems
D10: A graph grammar approach to graphical parsing

		D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
1.	cafe,conceptual,model,managing,electronic,mail	1	0	0	0	0	0	0	0	0	0
2.	information	0,5	0,5	0	0,5	0	0	0,5	0	0	0
3.	agent-based,gathering	0	1	0	0	0	0	0	0	0	0
4.	approach	0	0,7	0	0	0	0	0	0	0	0,7
5.	dunhuang,frescoes,based,similarity,calculation	0	0	1	0	0	0	0	0	0	0
6.	retrieval	0	0	0,5	0,5	0,5	0,5	0	0	0	0
7.	web	0	0	0	1	0	0	0	0	0	0
8.	document,images	0	0	0	0	1	0	0	0	0	0
9.	survey	0	0	0	0	0,6	0,6	0,6	0	0	0
10.	introduction,content-based,image	0	0	0	0	0	1	0	0	0	0
11.	graph	0	0	0	0	0	0	0,5	0,5	0,5	0,5
12.	visualization,navigation	0	0	0	0	0	0	1	0	0	0
13.	floorplan,generation,area,optimization,and-or,search	0	0	0	0	0	0	0	1	0	0
14.	visual,programming,rewriting,systems	0	0	0	0	0	0	0	0	1	0
15.	grammar,graphical,parsing	0	0	0	0	0	0	0	0	0	1

Tabela 1: **Demonstracijski primer - matrika izrazov in dokumentov po prvem koraku**

in na podlagi teorije informacij [Kowalski, 97]. V praksi izkazujeta večjo učinkovitost uporaba logaritemske uteži za lokalno utež in entropijske uteži za globalno utež [Dumais, 91]. V našem primeru globalne uteži nismo uporabili, saj se vektorji izrazov v drugem koraku pri računanju matrike korelacij normalizirajo in se globalna utež, ki sorazmerno uteži komponente vektorjev izrazov, s tem izniči. Globalna utež, ki vključuje inverzno frekvenco izrazov za določen dokument [Yates, 99] in zato ne uteži vseh komponent vektorja izraza sorazmerno, pa je dajala na testih slabše rezultate, kot če globalne uteži nismo uporabili.

Kot lokalno utež smo uporabili logaritemsko, torej za vse elemente v matriki A je $a_{ij}=log(tf_{ij} + 1)$, pri čemer je tf_{ij} frekvenca pojavljanja izraza i v dokumentu j. S tem smo zmanjšali učinek večjih razlik pojavljanja izrazov v frekvencah, ker je za tvorjenje konceptov bolj pomembno, da neki izraz sodeluje v konceptu, kot to, kolikokrat sodeluje.

Vektorje izrazov na koncu *normaliziramo* in *združimo* izraze, ki so predstavljeni z enakimi vektorji (imenujemo jih bratje, saj nastopajo v istih dokumentih z istimi frekvencami in za tovrstno samodejno obdelavo med njimi ni razlik).

Rezultat prvega koraka na demonstracijskem primeru prikazuje tabela 1. Iz naslovov nad tabelo smo s pregledovalnikom izločili blokirane izraze s pomočjo seznama angleških blokiranih izrazov SMART [SMART], ki je nastal v okviru dolgoletnega projekta na univerzi Cornwell. Dimenzije začetne matrike A so bile 39 (izrazov) x 10 (dokumentov). Elemente matrike smo utežili z lokalno logaritemsko utežjo, vektorje izrazov normalizirali in združili brate. Tabela 1 prikazuje končno matriko A, ki je osnova za nadaljnje korake metode.

2. Računanje matrike korelativnosti izrazov

V drugem koraku metode izračunamo korelacije (podobnost, soodvisnost) poljubnih dveh izrazov. Vse korelacije predstavimo v matriki korelacij C. Korelacije dobimo z računanjem kosinusa kota w_{ij} med vektorji vseh izrazov i in j, $i<>j$. Kosinus kota nam pove, kako blizu so vektorji izrazov oziroma koliko sodelujejo pri opisu vsebine dokumentov. Ker so vektorji normalizirani, je računanje kosinusa enako množenju vektorjev:

$$c_{ij} = \cos\varpi_{ij} = (\boldsymbol{e}_i^T \boldsymbol{A})(\boldsymbol{A}^T \boldsymbol{e}_j), \qquad (1)$$

za $i,j=1,\ldots,m$, $\boldsymbol{e}_j$ je enotski vektor dimenzije n; m predstavlja št. izrazov, n predstavlja št. dokumentov.

Če je vrednost produkta blizu 1, pomeni, da sta vektorja izrazov skoraj vzporedna oziroma da sta izraza na podoben način zastopana v vsebini množice dokumentov.

Matrika C je simetrična, na diagonalo postavimo ničle. Algoritem za računanje matrike C je v splošnem reda $O(m^2n/2)$, vendar ga lahko v našem primeru pospešimo, ker so vektorji izrazov v matrikah izrazov in dokumentov A redki. Pri običajnih dimenzijah matrike A, to je nad nekaj tisoč izrazov in nad nekaj sto ali tisoč dokumentov, je v povprečju 99 odstotkov elementov enakih 0 [Berry, 92]. Redkost ne nastopa v obliki pravilnih vzorcev, lahko pa jo izkoristimo za pospešitev algoritmov za računanje z matrikami [Berry, 99].

Prvi korak metode za gradnjo hierarhične strukture konceptov je običajen korak pri metodah pridobivanja informacij. Drugi je običajen v primerih, ko želimo grupirati izraze po podobnosti. Naslednji koraki so izvirni in specifični za metodo, ki je predmet tega prispevka.

3. Računanje globalnih korelacijskih stopenj in iskanju korenov hierarhij

Elementi c_{ij} matrike korelacij C izražajo podobnost parov izrazov i in j. Da bi lahko ugotovili, kako so posamezni izrazi korelativni z vsemi izrazi v množici dokumentov, smo uvedli tudi globalno korelacijsko stopnjo Gks_i za vsak izraz i:

$$Gks_i = \frac{1}{m}\left(\sum_{j=1}^{m} c_{ij} + k_i\right), \qquad (2)$$

m predstavlja število izrazov; k_i predstavlja število izrazov, s katerimi je korelativen izraz i.

Globalna korelacijska stopnja, izračunana po enačbi 2, je dajala najustreznejše empirične rezultate, kar je potrdilo intuitivno pričakovanje, da je najustrezneje upoštevati enakomeren vpliv razmerja med številom korelacij izraza i in št. vseh izrazov ter povprečne višine korelacij za izraz i.

Izvedli smo tudi teste, pri katerih smo prišteli h Gks_i globalno entropijsko utež izraza i, ki smo jo omenili v prvem koraku, saj se v prejšnjih korakih metode ni mogla postati opazna. Vendar se je pokazalo, da njen vpliv ni dovolj velik (ali dovolj različen od že upoštevanih sumandov), da bi spremenil izbor korenov hierarhij ali izboljšal njihovo gradnjo.

Po izračunu vseh Gks, ki jih uredimo po velikosti od najvišje do najnižje, lahko določimo korene hierarhij po naslednjem postopku:

- prvi koren postane izraz z najvišjo Gks,
- vsak naslednji koren postane izraz z nižjo Gks, ki je z dosedaj izbranimi koreni korelativen nižje od korenskega praga Kp (ali z drugimi besedami: kosinus kota med kandidatom za koren in dosedanjimi koreni mora biti nižji od Kp).

Za določitev korenskega praga Kp so se pokazale kot najbolj uporabne vrednosti med 0,2 in 0,5, za večino

primerov lahko vzamemo vrednost 0,3. Višja vrednost vodi k več korenom, ki so večinoma močneje zastopani v konceptih hierarhij, nižja vrednost pa k manj korenom, od katerih so nekateri šibkeje zastopani (zaradi nizke *Gks*). Če želimo minimizirati število hierarhij, lahko določimo prag samodejno, z računanjem korenov na določenem intervalu in izborom praga, pri katerem je njihovo število minimalno.

Na opisani način izberemo za korene izraze, ki nastopajo večinoma v različnih konceptih in so najbolj zastopani v ustrezni hierarhiji.

Korene lahko določi tudi *uporabnik*, npr. na podlagi seznama izrazov, ki so urejeni po višini *Gks*. Ko jih izbere, se lahko določijo po zgornjem postopku samodejno še drugi koreni, da bodo v hierarhični strukturi zajeti vsi koncepti v dokumentih.

Rezultate drugega in tretjega koraka metode, uporabljene na demonstracijskem primeru, prikazuje tabela 2. V prvem stolpcu so globalne korelacijske stopnje (*Gks*) izrazov 1 do 15 (indeksi izrazov iz tabele 1), v drugem stolpcu so izrazi, ki so bili izbrani kot koreni pri pragu 0,3.

	Gks				Koreni
1.	0,1	9.	0,592	2.	information (0,743)
2.	0,743	10.	0,205	11.	graph (0,659)
3.	0,214	11.	0,659	9.	survey (0,592)
4.	0,408	12.	0,305	5.	dunhuang, frescoes, based, similarity, calculation (0,1)
5.	0,1	13.	0,1		
6.	0,588	14.	0,1		
7.	0,2	15.	0,214		
8.	0,205				

Tabela 2:
Demonstracijski primer – globalne korelacijske stopnje in izbrani koreni

Štirje koreni določajo štiri ločene hierarhije, znotraj katerih bodo umeščeni koncepti iz dokumentov. Četrti koren z nizko Gks_i je bil izbran kot koren zato, ker je z vsemi prejšnjimi korelativen nižje od praga *Kp* in bi lahko manjkal v celotni hierarhični strukturi konceptov, v katerem se nahaja, če bi ga izpustili.

4. Gradnja hierarhične strukture

Hierarhična struktura konceptov, ki jo zgradi metoda iz vsebine baze dokumentov, je sestavljena iz več drevesnih hierarhij, katerih korene smo določili v prejšnjem koraku. Gradnja vsake hierarhije poteka ločeno in po nivojih.

Preden začnemo z gradnjo, pripravimo za vsak izraz *j* korelacijske sezname Ks_j, to je sezname izrazov, ki so z njim korelativni. Sezname uredimo v padajočem redu po njihovi globalni korelacijski stopnji. S tem pospešimo gradnjo konceptov v hierarhiji, sicer bi ob iskanju otrok določenega izraza (prvi korak algoritma, ki je opisan v tem razdelku) iskali vsakič kandidata z maksimalno *Gks*.

Začnemo z uvrstitvijo korenskega izraza v hierarhijo. Ta je skupen vsem konceptom znotraj te hierarhije. Gradnja konceptov se nato nadaljuje po nivojih do listov – izrazov, ki nimajo otrok, kar pomeni, da ni več v okviru določenega koncepta korelativen z njimi noben drug izraz. Vsak otrok v hierarhiji ima le enega starša. Otroci določenega starša se uvrščajo v hierarhijo po višini *Gks*, najprej tisti z višjo stopnjo.

Koncepti so tako predstavljeni v drevesni hierarhiji kot *vse neciklične poti* od korena do lista.

Sledi postopek gradnje konceptov po nivojih:

Najprej vstavimo na prvi nivo vsake hierarhije indeks izraza, ki je njen koren, in mu dodamo množico indeksov dokumentov, v katerih nastopa (to so indeksi neničelnih elementov v njegovi vrstici matrike *A*). Množico indeksov dokumentov, ki so prirejeni indeksu izraza *j* v okviru določenega koncepta, imenujemo v nadaljevanju 'dokumenti izraza *j*'. Dokumenti se prenašajo navzdol do listov, s čimer zagotavljamo, da so izrazi (otroci) na nižjih nivojih vsebovani v istih konceptih kot njihovi starši.

Koncepte nato gradimo po nivojih. Izrazu *j*, ki se že nahaja v hierarhiji na nivoju *l*, dodamo otroke, ki nastopajo v okviru istega koncepta ali več konceptov, iterativno po naslednjih fazah:

I. Preverimo, ali obstaja v korelacijskem seznamu Ks_j kandidat za njegovega otroka. Če ne obstaja, preverimo, ali je izraz *j* list, kar pomeni, da v okviru tega koncepta noben izraz iz Ks_j ni mogel postati otrok. Če ni list, odstranimo iz njegovih dokumentov tiste, v katerih so vsebovani tudi njegovi otroci, da niso vsem izrazom v konceptu prirejeni dokumenti, ampak le listom. V obeh primerih označimo, da je izraz j v okviru tega koncepta zadnji - komponente vrstice *j*, ki ustrezajo njegovim dokumentom, v matriki *A* postavimo na 0 in končamo z gradnjo koncepta. Z ažuriranjem matrike *A* preprečimo, da bi se koncepti ali deli konceptov v hierarhiji po nepotrebnem ponavljali.

II. Vzamemo naslednjega kandidata za otroka iz seznama Ks_j. Pri njem preverimo:
   a. ali nastopa v okviru koncepta, ki ga gradimo – torej ali se pojavlja v dokumentih izraza *j*. Če se ne, začnemo ponovno s fazo I.
   b. ali se nahaja med predniki tega izraza. Če se nahaja, začnemo ponovno s fazo I.
   c. ali nastopa v enakem konceptu, kot kateri od že dodanih otrok svojega starša – torej ali je korelativen s katerim od njih. Če je, je smiselno, da ga

uvrstimo pod njega, zato ga ne dodamo pod izraz j. S tem zagotovimo celovitost konceptov. Če je iz korelacijskih seznamov takšnega otroka in izraza, ki ga dodajamo, bila izbrisana njuna relacija v katerem od prejšnjih konceptov, jo povrnemo, da ga bomo lahko kasneje res uvrstili pod njega.

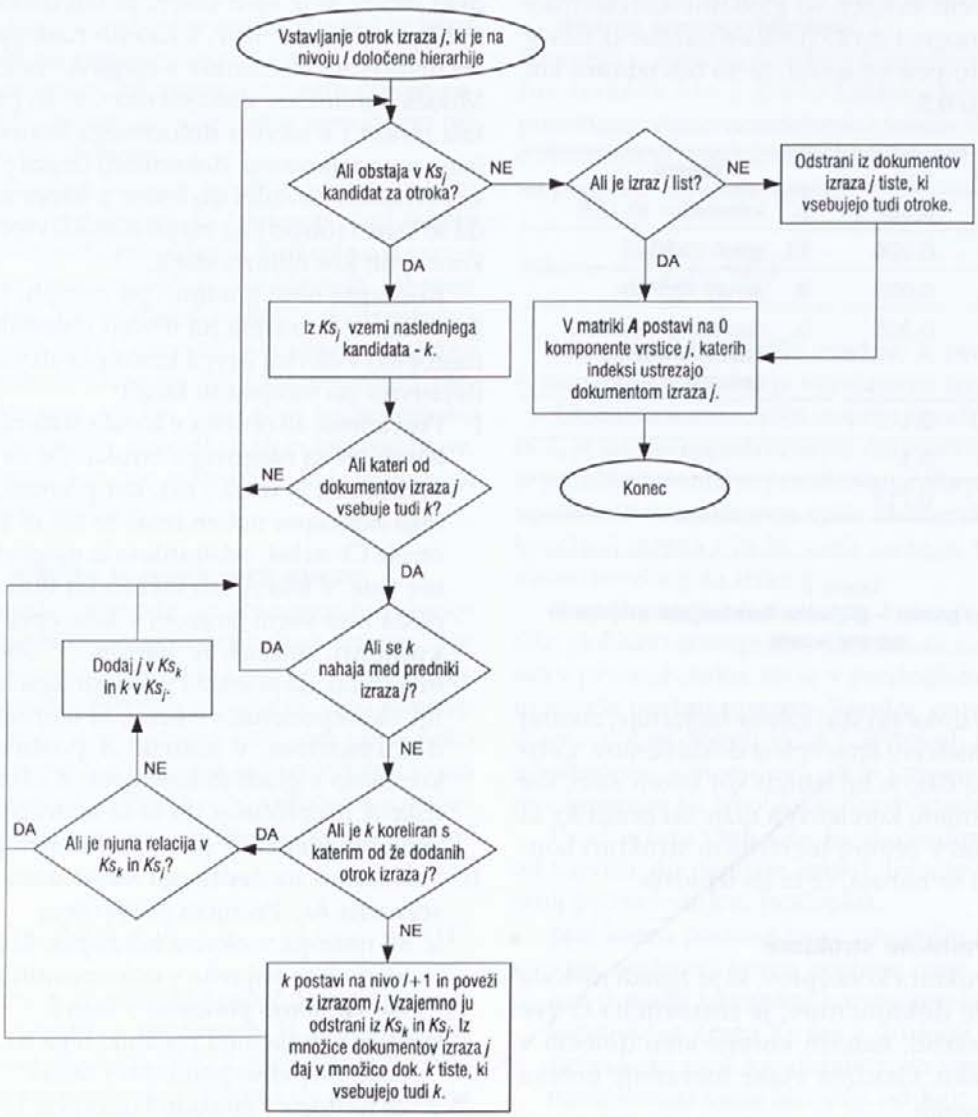
III. Izraz, ki je postal otrok, postavimo na nivo $l+1$ in ga povežemo z izrazom j. Dodamo mu podmnožico dokumentov izraza j, v katerih nastopa v okviru tega koncepta, in zbrišemo otroka iz korelacijskega seznama izraza j ter izraz j iz korelacijskega seznama otroka, da se ne ponovi njuna relacija (lahko jo povrnemo v fazi II).

Slika 1 prikazuje diagram opisanega algoritma za gradnjo konceptov. Na sliki 2 je predstavljen rezultat tega koraka metode, uporabljene na demonstracijskem primeru. Hierarhije so zgrajene in oštevilčene pri korenskih izrazih. Pri izrazih na nižjem nivoju so navedene v okroglih oklepajih njihove globalne korelacijske stopnje. Pri listih se nahajajo v koničastih oklepajih seznami indeksov dokumentov. Koncept, ki ga tvorijo izrazi od lista do korena, nastopa v vseh teh dokumentih.

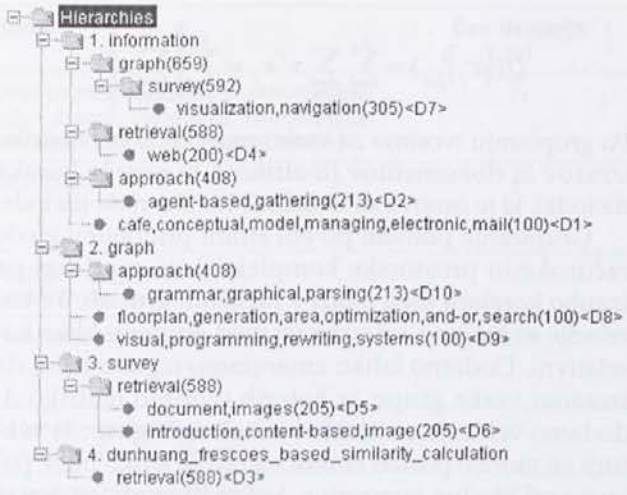
4. Razširitve metode v smeri večje učinkovitosti, neodvisne od jezika

4.1 Uporaba analize glavnih komponent

Pri pridobivanju informacij na osnovi vektorskega prostora se pogosto uporablja metoda za analizo glavnih



Slika 1: Algoritem gradnje konceptov – vstavljanje otrok izraza j, ki se nahaja na nivoju l določene hierarhije



Slika 2: **Demonstracijski primer – hierarhična struktura konceptov**

komponent, ki se imenuje razcep na singularne vrednosti (RSV) [Furnas, 88; Berry, 99]. RSV se uporablja sicer tudi pri procesiranju slik, obdelovanju signalov, stiskanju podatkov in v kriptografiji.

S to metodo lahko odstranimo iz začetne matrike izrazov in dokumentov "šum", to je vpliv, ki ga imajo dokumenti in izrazi, ki z ostalo večino vsebinsko niso povezani, in vpliv, ki nastane zaradi različne rabe besed. Uporaba RSV se je pokazala kot učinkovita pri prepoznavanju sopomenskih in večpomenskih besed. Ena od raziskav je pokazala, da je pridobivanje informacij na osnovi te metode, po učenju na študentski enciklopediji, vračalo enake odgovore na standardnem testu angleških sinonimov, kot so jih dajali uspešni tuji kandidati za študij v Ameriki [Landauer, 96]. Z RSV razcepimo osnovno matriko A (m x n) na

$$A = U\Sigma V^T,$$

kjer je U ortogonalna matrika dimenzij m x m, katere stolpci so levi singularni vektorji A, V je ortogonalna matrika dimenzij n x n, katere stolpci so desni singularni vektorji A in Σ je matrika dimenzij m x n, katere diagonalni elementi so singularne vrednosti A, urejene od najvišje do najnižje. Po razcepu lahko odrežemo k najnižjih singularnih vrednosti, s čimer aproksimiramo matriko A z matriko A_k (rečemo tudi, da zmanjšamo rang matrike na k). A_k, izračunana z RSV, velja za optimalno aproksimacijo ranga k matrike A [Eckart, 36; Stewart, 00].

Zaradi lastnosti metode RSV smo želeli ovrednotiti njen vpliv tudi v okviru naše metode, torej ugotoviti, kako bi RSV vplival na strukturo in povezavo konceptov. Razcep uporabimo v našem kontekstu na matriki A takoj po prvem koraku metode za gradnjo hierarhične strukture. Enačbo 1, po kateri smo v drugem koraku računali korelacije med vektorji izrazov, lahko sedaj preoblikujemo, saj ni potrebno po razcepu izračunati aproksimirane matrike A_k, da dobimo vektorje izrazov, ampak je dovolj, da izračunamo $\Sigma_k U_k^T$ (dimenzij k x m). Če definiramo $w_j = \Sigma_k U_k^T e_j$ (j=1,...,n), lahko enačbo 2 zapišemo kot:

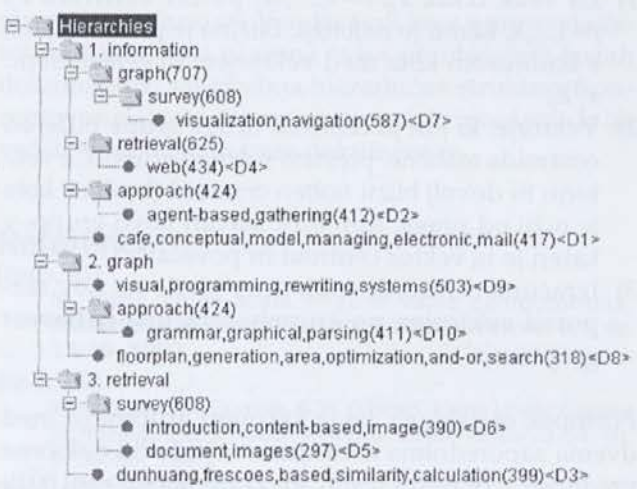
$$c_{ij} = \cos\varpi_{ij} = \frac{w_i^T w_j}{\|w_i\| \|w_j\|},$$

za i,j=1,...,m. Če se pojavijo po razcepu matrike A zaradi aproksimacije novi bratje (izrazi, katerih korelacija je 1), jih združimo, tako kot smo to naredili že v prvem koraku.

Razvite so bile hitre iterativne metode za izračun RSV, ki so prilagojene tudi redkim matrikam, kot sta Arnoldi-jeva [Lehoucq, 96] in Lanczos-eva [Parlett, 80], s katerimi se osnovna časovna in prostorska zahtevnost razcepa ($O(mn^2)$ za polno matriko) precej zmanjša (manj kot $O(mn)$).

Testi so pokazali, da z večjo aproksimacijo matrike A izgubljamo poleg šuma tudi natančnost korelacij, zaradi česar se vzpostavlja med izrazi vse več bratov in postajajo koncepti vse bolj splošćeni. Prednost uporabe RSV v gradnji hierarhične strukture konceptov je, da se združujejo koncepti, ki vsebujejo veliko skupnih izrazov, s čimer dosežemo manjšo občutljivost za variacije v rabi besed in za sinonime, kar je tudi sicer splošna značilnost RSV, znana iz pridobivanja informacij.

Težava pri uporabi te metode je, da se ne da vnaprej določiti optimalnega ranga aproksimirane matrike, ampak se to počne empirično. V pomoč so nam lahko raziskave, ki so v praksi [Berry, 95] in s statistično analizo [Ding, 99] pokazale, da je optimalni rang, v katerem se ohrani večina pomembne vsebinske strukture dokumentov, okrog 100 do največ 300.



Slika 3: **Demonstracijski primer – hierarhična struktura konceptov po razcepu na sing. vred., ohranili smo 6 sing.vred.**

Tak rang je optimalen pri bazah dokumentov, ki vsebujejo vsaj nekaj sto dokumentov.

Slika 3 prikazuje hierarhično strukturo, zgrajeno z aproksimacijo matrike A (iz demonstracijskega primera) z matriko ranga 6. Vidimo lahko, da se je tretji koren spremenil glede na hierarhijo na sliki 2, s čimer se je koncept, zajet v četrti hierarhiji na sliki 2, pravilno povezal s koncepti v tretji hierarhiji. Pri rangu 6 torej še nismo imeli nobene izgube v vsebini konceptov, pri nižjih pa je že prihajalo do izgub.

4.2 Pospešitev metode z grupiranjem izrazov

Računsko in prostorsko kompleksnost drugega in četrtega koraka gradnje hierarhične strukture konceptov lahko učinkovito zmanjšamo z uvedbo grupiranja izrazov. V ta namen smo prilagodili algoritem za sferično grupiranje v k grup [Dhillon, 01], ki je različica evklidskega algoritma za grupiranje v k grup - $W_1,\ldots, W_k$. Algoritem smo razširili z možnostjo dinamičnega ustvarjanja novih grup, kajti ne da se vnaprej oceniti, koliko grup bodo tvorili izrazi, ki opisujejo podobno vsebino, iz poljubne baze dokumentov.

Grupiranje izrazov izvedemo po koraku gradnje matrike izrazov in dokumentov. Najprej razporedimo m normaliziranih vektorjev izrazov $(x_1, \ldots, x_m)$ matrike A naključno v k grup. Nato izračunamo vsem grupam normalizirane centroide:

$$c_j = \frac{\sum_{x\in\Omega_j} x}{\left\| \sum_{x\in\Omega_j} x \right\|},$$

za $j=1,\ldots,k$. (3)

Iterativni postopek za grupiranje je naslednji:

1) Za vsak izraz x_i, $i=1,\ldots,m$, poišče centroid c_j, $j=1,\ldots,k$, ki mu je najbližji. Bližina je predstavljena s kosinusom kota med vektorjem in centroidom: $x_i^T c_j$.
2) Vektorje, ki jim je centroid druge grupe bliže od centroida matične, prestavi v drugo grupo. Če vektorju ni dovolj blizu noben centroid (kosinus kota je nižji od praga, npr. 0,1), ustvari novo grupo, v kateri je ta vektor centroid in poveča k.
3) Izračuna centroide vseh grup glede na nov razpored vektorjev po grupah. Izračuna kakovost grupiranja.

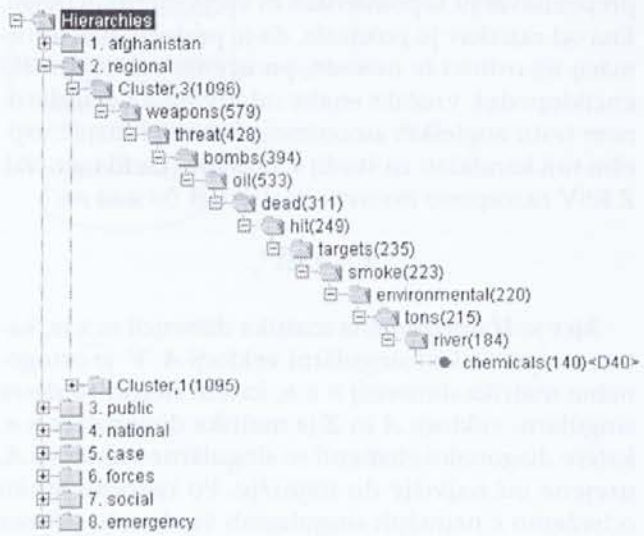
Postopek se zaključi, ko je kakovost grupiranja med dvema zaporednima iteracijama manjša od določene vrednosti e (v naših testih smo uporabljali e=0.001). Kakovost grupiranja izračunamo kot vsoto kakovosti ali koherenc posameznih grup in ima limito, h kateri hitro konvergira [Dhillon, 01]:

$$Q(\{\pi_j\}_{j=1}^k) = \sum_{j=1}^{k} \sum_{x\in\pi_j} x^T c_j = \sum_{j=1}^{k} \left\| \sum_{x\in\pi_j} x \right\|.$$

Po grupiranju tvorimo za vsako grupo posebej matriko izrazov in dokumentov (matriko A iz prvega koraka metode), ki je osnova za nadaljnje korake naše metode.

Grupiranje pomeni po eni strani pridobitev glede računske in prostorske kompleksnosti, po drugi pa izgubo korelacij med izrazi v različnih grupah. Te korelacije so majhne, saj so izrazi med grupami šibko korelativni. Dodatno lahko zmanjšamo izgubo tako, da izrazom vsake grupe, iz katerih tvorimo matriko A, dodamo vektorje centroide vseh drugih grup. Ti vektorji ne morejo postati koreni hierarhij, lahko pa se pojavijo pri gradnji konceptov, kot indikatorji izrazov iz druge grupe, ki nastopajo v tem konceptu.

Primer hierarhične strukture, zgrajene po grupiranju izrazov, predstavljamo na sliki 4. Prikazana je struktura, zgrajena iz druge grupe. Centroidi drugih grup, ki so vključeni v koncepte, so predstavljeni z izrazom 'Cluster', številko grupe in globalno korelacijsko stopnjo. Baza dokumentov za ta primer je enaka, kot v naslednjem poglavju pri prikazu časovne zahtevnosti. Začeli smo z grupiranjem v dve grupi, tretja je bila ustvarjena med grupiranjem. Grupiranje je bilo zaključeno po 19 iteracijah.



Slika 4: Hierarhična struktura po grupiranju

5. Časovna zahtevnost metode

Časovno zahtevna sta koraka metode za računanje matrike korelacij ($O(m^2n)$) in za gradnjo hierarhične strukture, katerega zahtevnost je zelo odvisna od korelacij med izrazi. Zahtevna je tudi razširitev z uvedbo razcepa s singularnimi vrednostmi ($O(mn^2)$), ki pa se z iterativnimi postopki zmanjša na $O(mn)$.

Korak	Čas izvajanja (s)	%	Čas izvajanja z razcepom (s)	%	Čas izvajanja z grupiranjem (s)	%
Gradnja matrike izrazov in dokumentov (izbor izrazov, uteževanje, normalizacija vektorjev izrazov, združevanje bratov)	77	13	77	13	77	27
Grupiranje	/		/		10 (3 grupe, 19 iteracij)	3,5
Razcep s sing. vrednostmi (aproksimac. na rang 40)	/		15	2,5	/	
Računanje matrike korelacij	118	21	137	23	55 (skupaj, za vse grupe)	19
Računanje globalnih korelacijskih stopenj in iskanje korenov hierarhij	3	0,5	3	0,5	3	1
Gradnja hierarhične strukture	373 (23 hierarhij)	65,5	354 (15 hierarhij)	61	141 (skupaj, za vse grupe, skupno 32 hierarhij)	49,5
Skupno	**571**	**100**	**586**	**100**	**286**	**100**

Tabela 3: **Razmerja časovnih zahtevnosti korakov metode**

Metodo smo implementirali v okolju za matematično modeliranje Scilab 2.6, ki ga je razvil INRIA – Francoski nacionalni inštitut za raziskave na področju računalništva in avtomatike (http://www-rocq.inria.fr/scilab/). V tem okolju se programski ukazi interpretirajo. Zelo počasno je v tem okolju tudi obdelovanje netipiziranih seznamov, ki jih uporabljamo pri gradnji strukture in obdelovanje redkih matrik. Zaradi teh razlogov so časovne vrednosti, ki jih navajamo v tabeli 3, namenjene zgolj približni oceni razmerij časovne zahtevnosti med posameznimi koraki metode. Hitrost izvajanja metode, implementirane s prevajanim programskim jezikom, npr. C++, v katerem bi lahko uporabili tudi bolj učinkovite podatkovne strukture, bi bila precej večja.

Baza dokumentov za prikaz časovne zahtevnosti je vsebovala 110 odstavkov desetih daljših člankov o globljih vzrokih terorizma in o vojni v Afganistanu. Od 3078 izrazov, ki so ostali po izločanju blokiranih besed, smo ohranili izraze, ki so se pojavljali v celotni bazi dokumentov s frekvenco, višjo od 4. Teh izrazov je bilo 305.

6. Zaključek

Metoda, ki smo jo razvili, gradi hierarhično strukturo konceptov iz vsebine baze dokumentov neodvisno od jezika, ki je uporabljen v dokumentih. Učinkovitost kompleksnih metod za pridobivanje informacij je v praksi odvisna od integracije zmogljivih statističnih pristopov z izčrpnimi jezikovnimi bazami – predvsem slovarji in tezavri. S slovarjem določenega jezika in/ali tezavrom določenega vsebinskega področja bi lahko naredili natančen izbor izrazov, ki jih želimo vključiti v hierarhično strukturo, in jih ustrezno utežili.

Vendar v praksi slovarji in tezavri večinoma niso na voljo in smo v teh primerih odvisni predvsem od statističnih značilnosti teksta. Na njihovi podlagi in v povezavi z uporabniškim izborom korenov hierarhij ter razcepom s singularnimi vrednostmi zgradi opisana metoda zelo uporabno hierarhično strukturo, ki omogoča učinkovit pregled konceptov in iskanje po njih ali njihovih delih – tako v hierarhijah kot v samih dokumentih, saj vsebujejo hierarhije informacije o dokumentih, v katerih se nahajajo posamezni koncepti.

Časovno zahtevnost lahko precej zmanjšamo z grupiranjem izrazov. Vendar tudi brez grupiranja časovna zahtevnost ni resna ovira pri običajnih bazah dokumentov, saj gradnja hierarhične strukture konceptov ni pogosta dejavnost, ampak jo izvedemo le ob večjih spremembah baze dokumentov.

Reference:

[Berry, 92] M. Berry, Large scale singular value computations. *International Journal of Supercomputer Applications*, 6:1, str. 13-49, 1992.

[Berry, 95] M. W. Berry, S.T. Dumais, G.W. O'Brien, Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, 37:4, str. 573-595, 1995.

[Berry, 99] M. W. Berry, Z. Drmač, E. R. Jessup, Matrices, Vector Spaces, and Information Retrieval. *SIAM Review*, 41:2, str. 335-362, 1999.

[Dhillon, 01]
I. S. Dhillon, D. S. Modha, Concept Decompositions for Large Sparse Text Data Using Clustering. *Machine Learning*, 42:1/2, str.143-175, 2001.

[Dimec, 99]
J. Dimec, L. Todorovski, D. Hristovski, S. Džeroski, The personalized search engine for Slovenian and English medical documents. V *Managing multimedia collections*, Bled, str.56-63, 1999.

[Ding, 99]
C. H. Q. Ding, A Similarity-Based Probability Model for Latent Semantic Indexing. V *Proceedings of 22nd International Conference on Research and Development in Information Retrieval*, Berkeley, str. 58-65, 1999.

[Dumais, 91]
S. T. Dumais, Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments & Computers*, 23:2, str. 229-236, 1991.

[Eckart, 36]
C. Eckart, G. Young, The approximation of one matrix by another of lower rank. *Psychometrika*, 1, str. 211-218, 1936.

[Forsyth, 86]
R. Forsyth, R. Rada, Adding an edge in Machine Learning. V *Machine Learning: Applications in Expert Systems and Information retrieval*, str. 198-212, 1986.

[Frakes, 92]
W. B. Frakes, R. Baeza-Yates, Information Retrieval: Data Structures & Algorithms, Prentice Hall, ZDA, 1992.

[Furnas, 88]
G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, K. E. Lochbaum, Information Retrieval using a Singular Value Decomposition Model of Latent Semantic Structure. V *Proceedings of ACM SIGIR*, str. 465-480, 1988.

[Harman, 95]
D. Harman. Overview of the third text REtrieval conference (TREC-3). V *Overview of the Third Text REtrieval Conference, National Institute of Standards and Technology Special Publication*, NIST, str. 1-21, 1995.

[Jansen, 98]
B. Jansen, A. Spink, J. Bateman, Searchers, the Subjects they Search, and Sufficiency: A Study of a Large Sample of EXCITE Searches. V *Proceedings of World Conference on the WWW, Internet and Intranet (WebNet-98)*, AACE Press, 1998.

[Kowalski, 97]
G. Kowalski, Information Retrieval Systems: Theory and Implementation, Kluwer Academic Publishers, 1997.

[Landauer, 96]
T. K. Landauer, S. T. Dumais, How come you know so much? From practical problem to theory. V *Basic and applied memory: Memory in context*, NJ: Erlbaum, str. 105-126, 1996.

[Lehoucq, 96]
R. Lehoucq, D. Sorensen, Deflation techniques for an implicitly restarted Arnoldi iteration, *SIAM Journal on Matrix Analysis and Applications*, 17:4, str. 789-821, 1996.

[Parlett, 80]
B. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, NJ, 1980.

[Popovič, 91]
M. Popovič, Implementation of a Slovene language free-text retrieval system. Doktorska disertacija, University of Sheffield, Department of Information Studies, Združeno kraljestvo, 1991.

[Porter, 80]
M. F. Porter, An algorithm for suffix stripping. *Program*, 14:3, str. 130-137, 1980.

[Porter, 01]
M. F. Porter, Snowball: A language for stemming algorithms, http://snowball.sourceforge.net/texts/introduction.html, 2001.

[Salton, 71]
G. Salton, The SMART Retrieval System, Prentice Hall, 1971.

[Sanderson, 99]
M. Sanderson, B. Croft, Deriving concept hierarchies from text. V *Proceedings of the 22nd Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, Kalifornija, str. 206-213, 1999.

[SMART]
SMART English stopword list, SMART project, Cornell University, ftp://ftp.cs.cornell.edu/pub/smart/.

[Stewart, 00]
G. W. Stewart, The Decompositional Approach to Matrix Computation. *IEEE Computing in Science & Engineering*, 2:1, str. 50-59, 2000.

[Woods, 97]
W. A. Woods, Conceptual Indexing: A Better Way to Organize knowledge. *Sun Labs Technical Report: TR-97-61*, Technical Reports, Palo Alto, 1997.

[Yang, 00]
H.C. Yang, C. H. Lee, Automatic Category Generation for Text Documents by Self-Organizing Maps. V *Proceedings of the IEEE-INNS-ENNS Int. Joint Conference on Neural Networks (IJCNN'00)*, Italija, str. 3581-3586, 2000.

[Yates, 99]
R.B. Yates, B.R. Neto, Modern Information Retrieval, Addison-Wesley Pub. Co., New York, 1999.

◆

Robert Leskovar je asistent – mladi raziskovalec na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru. Diplomiral je leta 1998 in končuje doktorski študij računalništva in informatike. Področja njegovega raziskovalnega dela zajemajo pridobivanje informacij s poudarkom na prepoznavanju konceptov v nestrukturiranem tekstu, tehnologije računalniško posredovanega komuniciranja in zagotavljanje kakovosti pri razvoju informacijskih sistemov ter drugih delovnih procesov.

◆

Dr. József Györkös je izredni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru. Področja njegovega dela zajemajo računalniško posredovano komuniciranje, teorijo upravljanja zahtev in zagotavljanje kakovosti pri razvoju informacijskih sistemov ter širše. Znanje in izkušnje je s sodelavci apliciral v nekaterih večjih družbah in na ravni informatike v vladnih institucijah. Objavil je več strokovnih in znanstvenih publikacij v mednarodnih revijah in knjigah. Trenutno opravlja delo dr•avnega sekretarja na Ministrstvu za informacijsko družbo.

◆

Dr. Ivan Rozman je redni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Je dekan fakultete in vodja Inštituta za informatiko. Področja njegovega pedagoškega, znanstvenega in raziskovalnega dela zajemajo programsko inženirstvo, objektno tehnologijo, vodenje projektov in kakovost v programskem inženirstvu. Je vodja več raziskovalnih projektov in avtor številnih znanstvenih ter strokovnih publikacij.

CLEARCASE – ORODJE ZA UPRAVLJANJE KONFIGURACIJE

Uroš Sajko
Hermes SoftLab d.d., Litijska 51, Ljubljana
projekt OmniBack
uros.sajko@hermes.si

Besedilo predstavlja osnovne probleme in zahteve, s katerimi se srečuje upravljavec konfiguracije pri velikem projektu. Predstavljeni so zgodovinski razvoj in orodja, ki omogočajo upravljanje konfiguracije. Eno izmed teh orodij je ClearCase, ki ga uporabljamo tudi pri projektu OmniBack v podjetju Hermes SoftLab. Prikazane so prednosti novega orodja, težave pri prehodu na orodje ClearCase. Opisana so osnovna pravila, ki jih je potrebno upoštevati, da bi lahko vsak član projekta nemoteno delal in je administracija okolja kar najbolj enostavna in razvidna.

Uvod

Pravočasno izdati kvalitetne programske proizvode in jih vzdrževati je osnovni cilj in politika večine programskih organizacij. Preden se proizvod lahko izda, ga je potrebno zgraditi. Za vzdrževanje pa je potrebno vedeti, kaj je tisto (katera inačica izvorne kode), kar stranka ima.

Problem in izziv je razviti programski proces za pretvarjanje tisočev datotek, ki jih spreminja desetine ljudi, v konsistentno celoto kode in dokumentacije. Potrebni so postopki raznih nivojev, predpisi, izbrana orodja in organizacijski napori, da stalne spremembe zahtev, in zato načrtov kode, privedejo do stabilnega proizvoda, primernega za izdajo. Potrebno je določiti vse, od načina shranjevanja in dostopa, do datotek, dogovorov o prevajanju modulov, komponent in celega proizvoda, do banalnosti, kot so varnostne kopije proizvoda v primeru nesreč.

Upravljanje konfiguracije (UK) je organizacija takega programskega okolja. S tem pojmom označujemo danes postopke in metodologije za organizacijo delovnega okolja, administracijo kode in drugih dokumentov, in izgradnjo programskega proizvoda. Kot formalizacija osnovnih politik organizacije je UK eden od temeljev proizvodnje programske opreme.

Upravljanje konfiguracije je hkrati veda in postopek. Njegov namen je zagotoviti organizaciji informacije, ali bo končni proizvod takšen kot je bil načrtovan. Ko govorimo o upravljanju konfiguracije, mislimo na sledenje in kontrolo razvoja programske opreme in povezanih aktivnosti. Ob tem mislimo na vodenje razvoja projektov programske opreme, ko več razvijalcev dela isto izvorno kodo ob istem času, razvija proizvod za več okolij, ob tem pa sta omogočena shranjevanje revizij kode in nadzor nad stanjem kode.

Upravljanje konfiguracije je postopek, ki obsega identifikacijo, organiziranje in kontrolo sprememb razvoja programske opreme v projektni skupini. Med drugim zagotavlja učinkovita orodja z namenom, da bi izdelali pravi proizvod. UK je posebej pomembno danes, ko uporabniki zahtevajo poceni proizvode, pričakujejo hitro dostavo in storitve, ki bi vedno bolj in bolj sovpadala z njihovimi pričakovanji. V teh primerih je UK ključno orožje v konkurenčnem boju.

Pogled v zgodovino

Z razvojem programske opreme se je pojavila potreba po spremembi le te. Prvo "upravljanje konfiguracije" je potekalo ročno. V času, ko so se programi še shranjevali na luknjanih karticah, je bila kontrola revizij izvedena tako, da so kup luknjanih kartic povezali v sveženj, katerega so označili z datumom in imenom programa. Svežnje so shranili, da bi lahko razvijalcem zagotovili vrnitev na prejšnjo verzijo.

Tehnologija je napredovala in fizične medije so zamenjali magnetni. V tem času so se arhivirali koluti trakov, kasneje so kolute zamenjali vedno manjši mediji z vedno večjimi kapacitetami. Kljub vsemu pa je bila v tem času kontrola revizij še vedno ročna, saj je bilo potrebno na vsak medij ročno shraniti določeno verzijo.

Kasneje so se pojavili veliki diski, kar je omogočilo skupno rabo izvorne kode. Odjava (check-out) kode je bila še vedno ročna, kar je pomenilo, da so se morali razvijalci dogovoriti in si rezervirati module, ki so jih želeli spreminjati. Za te zadeve so navadno uporabljali tablo, na katero so napisali svoje ime ob imenu modula, ki so si ga je rezervirali. Prijava (check-in) kode je potekala tako, da so izbrisali svoje ime ob imenu modula, kar je pomenilo, da je lahko modul rezerviral katerikoli izmed razvijalcev.

V poznih 60-ih in zgodnjih 70-ih letih je profesor Leon Pressor iz Univerze Santa Barbara v Kaliforniji

postavil tezo, da je potrebno spremeniti kontrolo konfiguracije. Posledica teze je bil podpis pogodbe z vojaškim dobaviteljem, ki je izdeloval letalske motorje za mornarico. Tudi vojaško letalstvo je želelo kupiti enako orodje za upravljanje konfiguracije.

Tako je bil leta 1975 zgrajen prvi komercialni proizvod, ki se je imenoval "Change and Configuration Control" in ga je prodajala programska hiša SoftTool. Od takrat naprej se na tržišču pojavljajo vedno novejša in boljša orodja, brez katerih si ni mogoče predstavljati upravljanja konfiguracije.

Skoraj istočasno z razvojem programske opreme za UK, so se začeli pojavljati prvi standardi. Tako za prvi standard na področju UK štejemo standard AFSCM 375-1, ki ga je leta 1962 izdal American Air Forces (Leon 2000). S standardom so želeli urediti komunikacijske probleme pri načrtovanju reaktivnih letal. Standardu so sledili drugi standardi večinoma iz ameriške vojske in oddelkov za obrambo.

Med bolj znanimi standardi je sveženj standardov znanih pod imenom niz 480, ki so bili sprejeti leta 1970. Leta 1991 so niz 480 združili v en standard znan kot MIL-STD-973, ki je še vedno v uporabi, vendar je tik pred razveljavitvijo. V začetku leta 1990 je bila dana pobuda za sprejetje nevladnega standarda, kjerkoli je to mogoče. To je bil začetek standarda, ki ga danes širši krogi sprejemajo kot temeljnega na področju upravljanja konfiguracije.

Poznamo ga pod imenom "National Consensus Standard for Configuration Management", ANSI/EIA-649-1998 in ga je izdelal komite G33 za upravljanje konfiguracije podjetja Electronic Industries Alliance. Seveda pa to ni edini standard, ki ga poznamo. Danes najbolj znani in uporabljeni standardi, ki obravnavajo področje upravljanja konfiguracije, so IEEE-828, ISO 9001, ISO 10007, SEI TR-8 in drugi.

Orodja za upravljanje konfiguracije

Pri upravljanju konfiguracije ločimo med dvema, v osnovi popolnoma različnima vrstama inačic - revizija in variacija. Revizija je nova inačica, ki je nadomestila staro. Revizije elementov navadno odražajo napredek v odstranjevanju napak v elementu, lahko pa tudi pomenijo dodajanje nove funkcionalnosti ali izboljšavo v učinkovitosti. Variacije elementa vsebujejo popolnoma enake funkcije, ki pa se različno obnašajo v določenih situacijah. Variacija določenega elementa je torej druga izbira (alternativa). Primer variacije je lahko modul za tiskanje, pri čemer vsaka variacija predstavlja določeno vrsto tiskalnika.

Orodja za UK avtomatizirajo številne procese, kot so nadzor konfiguracije, sledenje napak, spremljanje stanja, kontrola revizij in upravljanje izgradnje proizvoda. Prav te naloge so nujno zlo ročnega upravljanja konfiguracije. Orodja za UK z avtomatizacijo prej naštetih nalog prinašajo več prednosti:

- izboljšanje učinkovitosti razvoja (zaposleni lahko več časa namenijo razvoju, saj se jim ni treba ukvarjati z ročnimi postopki)
- informacijska integracija (vse informacije, ki si jih želijo uporabniki so integrirane in jih lahko kadarkoli prikličemo in prikažemo v želeni obliki)
- prilagodljivost (enostavno lahko izvedemo raznoliko, distribuirano ali paralelno, upravljanje konfiguracije na eni ali več lokacijah)
- boljša analiza in planiranje (uporaba osnovnih funkcija UK in podatkov le-te nam omogoča mnogo različnih zvrst odločitev, funkcij simulacije in analiz kaj če)
- zmanjšanje napak (avtomatizacija monotonih in ponavljajočih nalog, ki so jih prej izvajali zaposleni, odpravlja možnost napak)
- uporaba novejše tehnologije (z uporabo avtomatskih orodij uporabljamo tudi novejšo tehnologijo in pristope vgrajene v ta orodja).

Kljub veliko prednostim, ki jih prinašajo avtomatizirana orodja za UK, se je potrebno zavedati, da takšna orodja niso rešitev vseh problemov s področja UK. Namen teh orodij je le korak k učinkovitem upravljanju. Uporaba napačnih orodij ali nesmiselna uporaba pravih orodij nas lahko pripelje še v večje težave.

Vrste orodij

Na tržišču veliko orodij, ki nam omogočajo upravljanje konfiguracije. Orodja delimo v tri skupine (Leon 2000):

- specialna orodja za UK
- orodja za upravljanje in/ali nadzor sprememb
- orodja v javni lasti.

V prvo skupino spadajo orodja, ki popolnoma pokrivajo vse funkcionalnosti UK in so navadno procesno orientirana. V drugo skupino spadajo orodja, ki ne pokrivajo celotnega spektra UK, temveč le najpomembnejše funkcije upravljanja konfiguracije, kot so upravljanje sprememb, nadzor sprememb, upravljanje izgradnje ...

Orodja so lahko tudi v javni lasti, torej so brezplačna. Uporaba takšnih orodij je priporočljiva le za projekte, ki niso zelo zahtevni. Pri takšnih orodjih ne moremo pričakovati tehnične podpore. Navadno se takšna orodja uporabljajo v akademskih projektih.

Večina današnjih orodij za UK pokriva področje za upravljanje sprememb, kot tudi njihov nadzor. Vsako izmed orodij ima svoje prednosti in slabosti, zato je pri odločitvi potrebno vedeti kaj pričakujemo od takšnega orodja, na primer ali bomo razvijali le na eni platformi ali več, ali bomo izvorno kodo razvijali na eni lokaciji ali na več lokacijah in podobno.

Danes najzmogljivejša orodja za UK omogočajo vse osnovne funkcije UK, kot so določitev, nadzor, spremljanje stanja in presojanje konfiguracije. Takšna orodja so: AideDeCamp (True Software), CCC/Harvest (Platinum Technology), ClearCase (Rational Software Corporation), ClearGuide (Rational Software Corporation), Continuus (Continuus), PVCS Dimensions (MERANT) in drugi (Leon 2000). Orodja se med seboj razlikujejo v notranji arhitekturi in načinu shranjevanja in beleženja sprememb, številu podprtih operacijskih sistemov, možnosti dela na več lokacijah in podobno (Eaton 2001). Orodje ClearCase bo bolj natančno opisano v naslednjem razdelku.

V skupino orodij za upravljanje in nadzor sprememb, ki ne pokrivajo celotnega področja UK, spada večino orodij za UK. Naštejmo le nekatere izmed njih: +1CM, Alchemist, ChangeMaster, CONTROL, MKS Source Integritiy, PVCS Version Manager, Sun WorkShop TemaWare, Visual SourceSafe.

Orodja v javni lasti so dostopna po internetu ali jih dobimo z operacijskim sistemom UNIX. Takšna orodja imajo precej dobro dokumentacijo. Primeri teh orodij so Aegis, DVS, CVS, Revision Control System (RCS), Source Code Control System (SCCS), TkCVS.

Uvajanje upravljanja konfiguracije

Uvajanje novega orodja za upravljanje konfiguracije pomeni ogromno spremembo. Novo orodje uvajamo v organizacijo, ker z obstoječim orodjem nismo zadovoljni, ali ker ne uporabljamo še nobenega orodja in bi ga v bodoče želeli uporabljati.

Uvajanje orodja za UK pomeni pomembno priložnost in veliko odgovornost. Priložnost je vse dokler nam takšno orodje omogoča odkrivanje problemov UK in izvajanje procesnih izboljšav za upravljanje razvoja in vzdrževanja programov. Uvedba novega orodja pomeni tudi veliko odgovornost zaradi razvejanosti in zaradi virov, potrebnih za izvedbo spremembe. Vsaka sprememba v organizaciji je zahtevna, še posebej če se nanaša na UK. UK namreč vpliva na vse združene podatkovne shrambe v organizaciji in na vse njene razvojne, testne, kakovostne in upravljavske procese. Spremembe morajo biti dobro vodene, kar pomeni, da mora biti uvajanje izvedeno tako, da bo uporaba orodja za UK uspešna, učinkovita in optimalna.

Uvajanje orodja za UK v organizacijo je treba skrbno načrtovati. Tipična strategija za prehod na novo orodje za UK je sestavljena iz petih faz:

- priprava in planiranje
- definiranje procesov v organizaciji
- izvedba pilotnega projekta
- širjenje uporabe
- izboljševanje procesov.

Z zgoraj naštetimi koraki začnemo potem, ko smo že ovrednotili vsa orodja med katerimi smo se odločali in smo že izbrali orodje, ki ga bomo v bodoče uporabljali v organizaciji.

Namen faze priprave in planiranja je izvesti nujne priprave za uvajanje orodja. Ta faza je izjemno pomembna in vsako izpuščanje te faze je lahko zelo boleče, dolgotrajno in navadno se pokaže kot neuspešno uvajanje novega orodja.

V tej fazi je potrebno ustanoviti skupino, ki bo zadolžena za uvajanje novega orodja. Skupina je odgovorna za izvedbo strategije uvajanja. Njena naloga je spremljati in sodelovati pri vseh fazah uvajanja orodja.

Skupina za uvajanje najprej izdela načrt uvajanja orodja za UK. Načrt opisuje prednosti orodja za UK, določa urnike in postopke realizacije uvajanja orodja, ustanavlja kriterije uspeha in vlogo skupine za uvajanje orodja. V tej fazi je treba razviti tudi načrt tveganja, to je tveganja, ki povzročijo odstopanja od načrta uvajanja orodja.

Namen definiranja procesov v organizaciji je zbirati in razumeti proces upravljanja konfiguracije, da bi ga lahko kar najbolje avtomatizirali in razumeli vlogo orodja v odnosu do procesa upravljanja konfiguracije. V tej fazi definiramo obstoječi proces in napišemo novega, če je to potrebno, napišemo dokument procesnega modela, ga implementiramo in opredelimo tveganje.

Namen izvedbe pilotnega projekta je testiranje modela procesa in preseljevanja skupnih podatkov v podatkovno shrambo, z namenom preverjanja delovanja orodja, določitvi krivulje učenja za različne uporabnike in za prototipiranje tveganih vprašanj.

Pri izvedbi pilotnega projekta je pomembno, da je v njem zajeto področje velikega tveganja, ki pa ne vpliva na kritično pot projekta. S pilotnim projektom razvija skupina zadolžena za uvajanje standarde, predpise in postopke, hkrati pa se uri za kasnejše delo pri upravljanju konfiguracije. Uspehi in neuspehi pri delu se dokumentirajo ter se primerjajo s kriteriji, določenimi v načrtu uvajanja.

Namen faze širjenja uporabe je postopno uvajanje orodja med posameznike in projektne skupine. Šolanje in zmanjševanje odpora do sprememb je ključna aktivnost te faze. Orodje UK, procesi, postopki in šolanje morajo biti izvedeni in uvedeni za vsako projektno skupino. Skupina, ki uvaja novo orodje za UK izvaja, ovrednoti in spremlja aktivnosti razširjanja uporabe novega orodja. Faza je končana, ko se orodje, procesi in postopki tekoče uporabljajo.

Namen faze izboljševanja procesov je ovrednotenje aktivnosti uvajanja, potrditev delujočih strategij in priporočanje procesnih izboljšav. To nam omogoča, da lahko v organizaciji izboljšamo procese in tako uživamo vse prednosti, ki jih omogoča orodje za upravljanje konfiguracije.

Orodje ClearCase

ClearCase je aplikacija tipa odjemalec-strežnik, ki omogoča kontrolo revizij, upravljanje okolja, kontrolo postopka in upravljanje prevajanja. Na strežniku se nahaja projektna podatkovna baza, v kateri so shranjene verzije izvorne kode. Odjemalci dostopajo do projektne podatkovne baze s pomočjo posebnih vmesnikov in omogočajo izmenjavo izvorne kode. Strežnik je lahko okolje HP-UX ali Windows, odjemalec pa lahko katerokoli okolje Unix ali Windows.

Projektna podatkovna baza se v orodju ClearCase imenuje Version Object Base (VOB). VOB je skladišče elementov, ki so shranjeni v več verzijah, s katerimi upravlja ClearCase. Element je objekt s stisnjenimi verzijami organiziranimi v obliki drevesne strukture. Glede na lastništvo in mesto hrambe elementov, ločimo javne in zasebne VOB-e.

Na strežnik ClearCase lahko namestimo več VOB-ov, da bi lahko elemente združili glede na naše potrebe. Direktno pisanje in branje VOB-a ni mogoče. Branje je mogoče le s posebnim vmesnikom, ki ga namestimo na odjemalcu. Pred spreminjanjem elementov, ki se nahajajo v VOB-u, je potrebno zahtevati odjavo elementa, strežnik zaklene element in na lokalnem disku se shrani kopija določene verzije elementa, ki jo lahko spreminjamo dokler spremembe ponovno ne prijavimo. Ob prijavi se element odklene, na strežniku se kreira nova revizija in na odjemalcu se pobriše kopija.

Ker lahko vsak odjemalec prikazuje elemente različnih VOB-ov, vsak element pa ima več verzij, je potrebno nekako določiti katere VOB-e, katere elemente in katere verzije želimo videti. Z drugimi besedami, potrebno je določiti konfiguracijo, ki jo želimo imeti na odjemalcu.

Konfiguracijo v orodju ClearCase nastavimo s tako imenovanim ConfigSpec-om. ConfigSpec je zaporedje pravil, s katerimi določimo VOB-e, elemente in verzije, ki bodo vidne v zasebnem delovnem prostoru (pogledu) določenega razvijalca. V zasebnem delovnem prostoru prikazujemo verzije imenikov in datotek enega ali več VOB-ov, kot tudi privatne datoteke, ki niso vidne pogledom. Vsak razvijalec ima lahko več pogledov. Ker je ConfigSpec lokalnega pomena, ga je potrebno določiti za vsak delovni prostor posebej.

Privzeti ConfigSpec zasebnega delovnega prostora je:

```
element * CHECKOUTED
element * /main/LATEST
```

Omenjeni ConfigSpec določa, da bo zasebni delovni prostor vseboval elemente, ki jih je razvijalec odjavil ali zadnje inačice elementov, shranjenih na strežniku v glavni veji.

Ločimo dve vrsti pogledov:

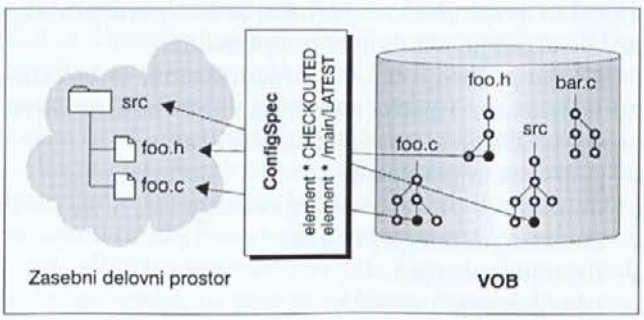
- Dinamični pogled se posodobi dinamično in ne zavzema prostora na disku, saj vidimo datoteke direktno iz VOB-a prek posebnega datotečnega sistema MultiVersion File System (MVFS)
- Posnetek (snapshot) vsebuje kopijo verzije elementov, ki jih shrani na lokalni disk in se ne posodobi dinamično, ampak le na zahtevo uporabnika

Slika 1 prikazuje VOB na katerem so shranjeni štirje elementi. Vsak izmed elementov ima več revizij. ConfigSpec se nahaja na odjemalcu in upošteva pravilo, ki določa, da želimo videti zadnje revizije vsakega elementa. Zato vidimo v zasebnem delovnem prostoru zadnjo verzijo imenika src in datoteki foo.c in foo.h. Datoteke bar.c ne vidimo, saj se ne nahaja v imeniku src. Datoteka je bila verjetno izbrisana v kateri izmed prejšnjih verzij imenika src, ali pa nikoli ni bila element tega imenika.

Verzije elementov so v VOB-u shranjene kot razlike (delte) med novo in staro revizijo. Zraven informacij o sami spremembi elementa shrani ClearCase še dodatne informacije, ki nam omogočajo slediti spremembe. Te dodatne informacije, ki se shranijo v seznam sprememb, so: datum in ura, uporabniško ime, dogodek (nova revizija, označevanje, kreiranje veje), verzija in komentar. Revizije so v orodju ClearCase predstavljene s krogom. Kvadrat predstavlja vejo ali variacijo. Privzeto ima vsak element glavno vejo, po potrebi pa lahko kreiramo nove stranske veje.

Orodje ClearCase omogoča označevanje revizij. Vsako revizijo, ki smo jo uporabili v postopku prevajanja in pakiranja, označimo z zaporedna število prevajanja. Takšno označevanje nam zagotavlja ponovljivost kateregakoli prevajanja, če le nastavimo ConfigSpec na želeno označbo.

Stranske veje lahko uporabljamo za razvoj novih funkcionalnosti in izdelavo popravkov zato, da s tem ne vplivamo na trenutni razvoj. Veja s popravki je primerna tudi zato, ker na enem mestu vidimo vse spremembe kode, ki so posledice odkritih napak,



Slika 1: **Preslikava elementov VOB-a v zasebni delovni prostor**

potem ko smo proizvod že oddali. Te popravke lahko po potrebi enostavno zlijemo tudi v glavno vejo ali katero drugo stransko vejo.

Orodje ClearCase je zelo prilagodljivo. Zraven opisanega nam nudi možnost dodajanja lastnih atributov in funkcij, ki jih vežemo na dogodke v orodju. To so na primer prijava elementa konfiguracije, odjava elementa, kreiranje označbe, kreiranje veje, brisanje elementa ... Vnaprej definirana funkcija se lahko sproži v dveh primerih:

- pred dogodkom
- po dogodku.

Funkcije navadno pišemo v jeziku Perl, saj je le-ta dodan orodju ClearCase in zato tako napisani sprožilci delujejo neglede na operacijski sistem, ki ga uporabljamo na odjemalcu.

Dodatek k orodju ClearCase z imenom Multisite nam omogoča, da izbrane VOB-e uporabljamo hkrati na več različnih lokacijah. Pri tem ima vsaka lokacija svoj strežnik ClearCase na katerem je nameščena kopija (replika) VOB-a. Strežniki na različnih lokacijah se med seboj sinhronizirajo s pošiljanjem nadgradnih paketov. Tak sistem nam omogoča nemoteno delo na različnih lokacijah, kljub slabi mrežni povezavi. Prvi sinhronizacijski paket vsebuje vse informacije izbranega VOB-a, vsi nadaljnji nadgradni paketi pa le razlike, ki so se zgodile v času od zadnje sinhronizacije.

Sinhronizacija se lahko sproži ročno ali avtomatsko. Avtomatska sinhronizacija se navadno izvaja ponoči, saj je mrežni promet takrat majhen. Nočna avtomatska sinhronizacija omogoča, da je zjutraj pred začetkom dela stanje na obeh lokacijah enako.

Uporaba pri projektu OmniBack

Projekt OmniBack II ima v Hermes SoftLab-u precej dolgo tradicijo. Njegovi začetki segajo v leto 1990, kar z drugimi besedami pomeni, da je bil OmniBack II prvi projekt v podjetju Hermes SoftLab. Na začetku so pri omenjenem projektu delali štirje ljudje. Z vsakim izidom programa je na projektu sodelovalo več ljudi, ki so pisali kodo za vse več in več platform. Tako se je projekt v prvih šestih letih povečal iz štirih programerjev na petindvajset (25) ljudi, ki so kodirali kodo za UNIX. Tega leta se je tudi prvič pojavila potreba po združitvi skupine, ki je razvijala kodo za sisteme UNIX z novo ustanovljeno skupino, ki je razvijala enak produkt za sisteme Windows in se je ustanovila sredi leta 1995. Do združitve obeh skupin je prišlo leta 1997. Sedaj šteje skupina OmniBack II 90 ljudi in ima za seboj 19 oddanih verzij programa.

Projekt OmniBack II razvija istoimenski produkt za naročnika Hewlett Packard. Produkt spada v skupino HP OpenView produktov in omogoča izdelavo varnostnih kopij podatkov in njihovo obnavljanje. Produkt je posebej prilagojen za globalne, široko-podjetne in distribuirane sisteme. Trenutno podpira približno 19 okolij (Windows NT i386, Windows DEC-Alpha, Novell, HP-UX 10.x, HP-UX 11.x, Silicon Graphics IRIX, IBM AIX, Sun Solaris, Sun OS, Alpha, Linux, SCO, NCR, Dynix, Sinix, ...), za katere ima lastne agente za delo z diskom in mediji, in se zna integrirati z večino najbolj znanih integracij (Sybase, Oracle, EMC, OPC, SAP, Informix, Symmetrix, Fastrax ...). V svetu si trenutno deli drugo mesto glede na število uporabnikov in se še vedno razvija.

Prehod na ClearCase

V začetni fazi smo pri projektu OmniBack II za Unix uporabljali program RCS, medtem ko smo pri sistemih Windows uporabljali program Source Integrity. Obe bazi sta bili ločeni. Vsaka izmed baz je zajemala približno štiri tisoč (4000) datotek izvorne kode, razvrščenih v približno dvesto petdeset (250) imenikov. Težava je bila, da smo na obeh sistemih (UNIX in Windows) razvijali isti produkt, kar pomeni, da je bila koda zelo podobna, nikakor pa ne enaka. Dnevno je bilo zato treba ročno sinhronizirati kodo med obema sistemoma, kar je bilo zelo zamudno in težavno opravilo. Vendar pa to ni bila edina pomanjkljivost prejšnjih orodij. Obe orodji sta bili zelo slabo zaščiteni pred zlorabami. Zato smo začeli kmalu razmišljati o orodju, ki bi rešilo te težave. Ob predhodnem pogovoru z ekspertom smo izbrali orodje ClearCase.

Prehod na ClearCase je potekal v več korakih:

- uskladitev kode s sistema Unix s sistemom Windows (imena datotek)
- uvoz kode v ClearCase (cca. 4000 datotek s povprečno 100 revizijami)
- prilagoditev starega okolja na novo orodje (sprememba poti, načina dela, ...)
- priprava tečaja in dokumentacije za razvijalce
- priprava skript za lažje delo (skripte za množično prijavo elementov v konfiguracijo).

Časovno najzahtevnejša koraka sta bila prva dva. Na srečo smo z uskladitvijo pričeli, še preden smo resno razmišljali o prehodu na novo orodje. Res pa je tudi, da smo se zadnji mesec pred prehodom ukvarjali samo z uskladitvijo.

Ker je bilo potrebno uvoziti okrog štiri tisoč datotek, nekatere med njimi pa so imele več kot 200 revizij, je to pomenilo, da bo celoten uvoz trajal 96 ur. Ker v tem času ni dovoljeno vpisovanje v staro bazo,

nova pa v tem času ni uporabna, bi to pomenilo, da razvijalci ne bi mogli nič delati. Zato smo se odločili za naslednji postopek:

1. Označimo celotno razvojno drevo (UNIX & NT) z označbo CC1, kar bo pomenilo prvo fazo uvoza v ClearCase.
2. Napravimo varnostno kopijo celotne baze (UNIX & NT) in jo obnovimo na strežnik ClearCase, da tako omogočimo razvijalcem nemoteno nadaljevanje dela.
3. Poženemo uvoz iz UNIX-ove varnostne kopije baze kar traja 96 ur, takoj po tem pa še uvoz iz NT-jeve varnostne kopije baze. Pri tem je treba paziti, da uvozimo le NT-specifične datoteke. Med tem časom, ko teče uvoz pa pripravimo tečaje in dokumentacijo za delo z orodjem.
4. Pred končno drugo fazo uvoza preverimo, ali so vse datoteke prijavljene.
5. Ponovno označimo celotno drevo (UNIX & NT) z označbo CC2.
6. Napravimo varnostno kopijo stare baze (UNIX & NT) in jo obnovimo na strežnik ClearCase. Staro bazo umaknemo iz uporabe, novo pa zaščitimo pred uporabo, dokler vsa koda ni uvožena. To pomeni, da kakšen dan ali dva nihče ne bo nič razvijal. Ta čas uporabimo za tečaj uporabe ClearCase.
7. Ponovno poženemo uvoz kode (UNIX & NT), vendar tokrat le morebitnih novih verzij, ki so bile prijavljene po označbi CC1, kar bi po naši oceni trajalo manj kot 24 ur.
8. S skriptami preverimo, če so bili uspešno vključeni vsi imeniki z vsemi datotekami in če so zadnje verzije kode znotraj ClearCase enake verzijam znotraj RCS.

Predstavitev pravil

Vsaka sprememba orodja zahteva precej dela in prilagajanja konfiguracije novemu okolju. V času zamenjave orodja lahko veliko stvari, ki so bile prej slabo realizirane ali definirane, izboljšamo in jih bolj natančno definiramo. Tukaj imamo v mislih predvsem poimenovanje VOB-ov, pogledov, datotek ... Ugotovili smo namreč, da se lahko z dobrim in jasnim poimenovanjem izognemo veliko težavam pri avtomatizaciji procesa, pri procesu usposabljanja novo zaposlenih in administraciji orodja.

VOB-i

Čeprav pojma VOB v prejšnjih orodjih nismo poznali, smo imeli celotno kodo kljub vsemu razdeljeno na štiri sestavne dele. Tem delom smo dodali nove in tako smo celotno kodo razbili na 12 VOB-ov ali delov. Vsak del tvori logično celoto in se večinoma lahko obravnava ločeno od ostalih delov. Ti logični deli so:

Ime VOB-a	Opis VOB-a
ADM	administracijski VOB - najvišji v hierarhični strukturi
BIN	datoteke, ki jih povezujemo v končni produkt in jih nismo napisali sami
BUILDS	končni paketi in produkti
CFG	orodja in skripte za prevajanje, povezovanje in pakiranje
DOC	projektna in interna dokumentacija
MANUALS	dokumentacije o produktu
PLAY	prostor za učenje, igranje in testiranje
SRC	izvorna koda produkta
SUPPORT	orodja, dokumentacija in skripte za skupino, ki skrbi za podporo strank
TEST	orodja in skripte za testno skupino
TOOL	orodja in skripte za administracijo celotnega okolja prevajanja
WEB	datoteke, orodja in skripte za WEB

Pogledi in datoteke

Ker se vsak pogled registrira na strežniku ClearCase, je potrebno poglede poimenovati, da vsakdo lahko določi kadarkoli, kdo je lastnik določenega pogleda in na katerem računalniku se ta pogled nahaja. Takšna pravila se izkažejo za nujna v primerih, ko razvijalec zapusti skupino in za seboj ne počisti delovnega okolja.

Tako se pri sistemih Windows zahteva, da poimenujemo pogled v obliki:

<uporabniško_ime>_<niz>_view

pri sistemih Unix pa ga poimenujemo v obliki:

<uporabniško_ime>_<niz>

pri čimer je niz poljubna kombinacija številk in črk, ki nam pomaga razlikovati poglede v primerih, ko ima en razvijalec več pogledov.

Ob kreiranju novih datotek v pogledih se zahtevajo imena brez presledkov. Pri sistemih Unix smo uvedli še dodatno zahtevo, ki prepoveduje poimenovanje datotek, ki bi se med seboj razlikovale le v velikih in malih črkah. Takšnih datotek ni mogoče pregledovati pri sistemih Windows, saj jih le-ti med seboj ne ločijo.

Novo datoteko (element) lahko v VOB doda kdorkoli, ki je član projektne skupine in je pred tem napravil osnovne teste s katerimi je preveril, da z dodajanjem ne ogrozi delovanja produkta. Dodana datoteka mora biti izvorna koda. Izvorna koda je datoteka, ki je ni mogoče kreirati iz nobene druge datoteke ali skupine datotek, ki so že shranjene v

VOB-u. Lokacija hrambe se določi v dogovoru z upravljavcem konfiguracije, katerega naloga je tudi, da nadzira upoštevanje pravila o dodajanju novih datotek v VOB.

Označbe, veje in zaklepanje

Spremenili smo tudi pravila označevanja in vejenja:

- z velikimi črkami deklariramo označbe.
- z malimi črkami deklariramo veje.

Za potrebe upravljanja konfiguracije se oboje kreira globalno za vse VOB-e, za razvijalce pa so globalne definicije prepovedane.

Za vsako uradno prevajanje celotne kode se avtomatsko kreira unikatn1a označba, ki nam omogoči ponovno izgraditi produkt. Ta označba ima format:

A.<verzija>_<zaporedna_številka>

Primer take označbe je A.03.50_103. Prvi del je izposojen iz HP-notacije za označevanje programske opreme.

Po oddaji se vsa oddana koda označi z označbo v formatu:

R<verzija>_MR

Istočasno se kreira tudi veja, na kateri se bodo izdelovali popravki (na glavni veji bomo razvijali novo verzijo produkta). Veja se poimenuje:

r<verzija>_fix

Označba, ki označuje oddano kodo, se zaklene, da je ne bi kdo po pomoti premaknil ali izbrisal. *Fix* veja se zaklene za vse razvijalce razen za tiste, ki skrbijo za podporo strank. Le-ti jo po potrebi odklenejo tudi za ostale razvijalce, v kolikor stranke potrebujejo popravke.

ConfigSpec

Configspec-e shranjujemo v ADM VOB-u, da ne prihaja do napak pri prevajanju in prijavljanju kode. Vsak ConfigSpec mora biti shranjen v dveh oblikah - prvi za Windows (končnica *.nt) in drugi za Unix (končnica *.ux).

ConfigSpec poimenujemo v naslednjem formatu:

Ime ConfigSpec-a	Opis
build_r<verzija>	ConfigSpec za uradno prevajanje
dev_r<verzija>	ConfigSepc za razvoj
support_r<verzija>	ConfigSpec za izdelavo popravkov
proto_<funkcionalnost>	ConfigSpec za razvoj novih funkcionalnosti, ki se bodo oddajale ločeno od produkta

Sprožilci

Ker zaradi zgodovinskih razlogov za prevajanje ne uporabljamo orodja *clearmake*, ki nam avtomatsko shranjuje informacije o tem, katera izvorna koda je bila uporabljena, smo morali sami izdelati skripto. Ta nam ob prijavi elementa avtomatsko pregleda izvorno kodo in zamenja niz znakov "$Header" z novim nizom, ki vsebuje podatek o datumu in uri prijave, razvijalcu, ki je spremembo naredil in zaporedni številki spremembe in imenu datoteke. Ti podatki so pomembni, saj jih ob povezovanju vpišemo tudi v končno izvršljivo datoteko. Z njihovo pomočjo lahko kadarkoli ugotovimo, kakšen program ima stranka, kdaj je bil preveden in katere izvorne datoteke so bile uporabljene. Te informacije pridejo še posebej prav skupini za podporo strankam, ki izdeluje popravke.

Multisite

Vse naše VOB-e sinhroniziramo z vsaj eno lokacijo - največkrat z našimi poslovnimi partnerji v Nemčiji in poslovno enoto v Sarajevu. Sinhronizacija poteka navadno avtomatsko s pomočjo najete linije, redkeje pa ročno po elektronski pošti. Prvi sinhronizacijski paket se vedno pošlje posnet na traku (DAT), saj so podatki reda velikosti 500 Mb. Sinhronizacija se vedno opravlja ponoči, da ne moti razvojnega procesa, saj se baza zaklene, da so podatki konsistentni.

Zaključek

Orodje ClearCase je zmogljivo orodje, brez katerega si pri projektu OmniBack enostavno več ne znamo predstavljati dela. Njegove največje prednosti glede na stari način dela so, da omogoča delo v mešanem okolju (UNIX in Windows), zaradi česar ne potrebujemo dveh ločenih projektnih podatkovnih baz z izvorno kodo in kode ni potrebno ročno usklajevati med njima. Naslednja ogromna pridobitev je tudi *Multisite*, s čimer smo se izognili ročnemu pošiljanju kode našim poslovnim partnerjem in oddaljenim enotam. Prej je bilo potrebno v vsaki beta fazi zapakirati izvorno kodo in končni produkt in ga poslati na strežnik FTP. Sedaj to več ni potrebno, saj se koda vsako noč sinhronizira, odpade pa tudi skrb, ali je bilo vse preneseno, saj za to skrbi samo orodje. Če torej želimo, da se končni produkt pošlje poslovnemu partnerju, ga je potrebno le dodati pod kontrolo ClearCase, vse drugo pa se zgodi avtomatsko. Obstaja še nekaj manj pomembnih razlogov za uporabo orodja, kot so centralno upravljanje, integriranje z velikim številom razvojnih orodij in poizvedovanja po konfiguraciji, ki nam omogočajo dober pogled nad spremembami kode.

Seveda pa ima vsaka medalja dve plati in zato ima tudi ClearCase svoje pomanjkljivosti. Največja med

njimi je rahla nestabilnost odjemalcev ClearCase in orodja za dostop do mrežnega podatkovnega sistema (NFS) na sistemih Windows. V takšnih primerih je neobhodno potreben ponoven zagon operacijskega sistema. Naslednja, mogoče manj pomembna slabost, je visoka cena in zahtevnost glede strojne opreme (pomnilnik, omrežna kartica).

Odgovor na vprašanje ali uporabljati orodje za upravljanje konfiguracije, je očiten. Odločitev ali boste izbrali orodje ClearCase ali katero drugo, pa je prepuščena vam. Izbrati je namreč treba orodje, ki bo zadostilo vašim zahtevam!

Literatura

1. Configuration Management Information Center (2001), What is Configuration Management, http://www.pdmic-com-cmic-introtoCM.shtml
2. Configuration Management FAQ (2001), http://www.iac.honezwell.com pub/Tech/CM/CMFAQ.html, junij 2001
3. Leon, A. (2000): A Guide to Software Configuration Management, Artech House Publishers
4. IEEE/ANSI 828-1990 Standard for Software Configuration Management Plans, Institute of Electrical and Electronics Engineers, 1990
5. ISO 10007:1995 Quality management - Guidelines for configuration management, International Organization for Standardization, Geneva, Switzerland, 1995
6. ISO 9001:2000 Quality management systems - Requirements, International Organization for Standardization, Geneva, Switzerland, 2000
7. Dart A. Susan (1993): CMU/SEI-93-TR-8, A Study in Software Maintenance, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr08.93.pdf
8. Eaton, D. (2001): Configuration Management Tools Summary, verzija 8.3b, http://www.daveeaton.com/scm/CMTools.html, junij 2001
9. Rational University (1998a), ClearCase Fundamentals for Windows NT version 3.2, Student Guide, Rational Software Corporation
10. Rational University (1998b), ClearCase Meta-Data for UNIX version 3.0, Training Manual, Rational Software Corporation

◆

Uroš Sajko je diplomiral na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru. Sedaj je študent podiplomskega magistrskega programa računalništva in informatike na tej fakulteti. Zaposlen je v podjetju Hermes SoftLab d.d. v Ljubljani, kjer je uvedel orodje Rational ClearCase, s katerim že nekaj let uspešno vodi upravljanje konfiguracije pri projektu OmniBack.

◆

Računalniško podprto načrtovanje ukrepov za razbremenitev kritičnih prenosnih poti

David Grgič Fakulteta za elektrotehniko, david.grgic@fe.uni-lj.si
Marko Bajec, Fakulteta za računalništvo in informatiko, marko.bajec@fri.uni-lj.si
Ferdinand Gubina, Fakulteta za elektrotehniko, ferdinand.gubina@fe.uni-lj.si
Robert Golob, Fakulteta za elektrotehniko, robert.golob@fe.uni-lj.si
Marko Senčar, Agencija za energijo RS, marko.sencar@agen-rs.si

Po deregulaciji slovenskega elektroenergetskega sistema je bila za razbremenjevanje kritičnih prenosnih poti, to je preobremenjenih vodov in transformatorjev, izbrana metoda s prerazporeditvijo proizvodnje delovne moči (Grgič, 2001). Ker gre za matematično zahtevno operacijo z elementi optimizacije, je Agencija za energijo Republike Slovenije naročila računalniški program za pomoč pri načrtovanju ukrepov za razbremenitev kritičnih prenosnih poti po navedenem principu. Članek predstavlja ta program. Program je zasnovan modularno in sicer iz lokalne baze podatkov, uporabniškega vmesnika za vnos in urejanje podatkov ter pregled rezultatov, modula izračuna optimalne prerazporeditve proizvodnje ter dela za pripravo standardiziranih poročil z rezultati izračuna.

1. Uvod

Uvajanje trga z električno energijo v slovensko elektrogospodarstvo zahteva nove rešitve vodenja obratovanja sistema, ki zagotavljajo zanesljivo in kakovostno obratovanje elektroenergetskega sistema (EES-a). Večina teh rešitev spada v sklop sistemskih storitev, za katere je zadolžen upravljavec sistema. Med temi storitvami je tudi razbremenjevanje kritičnih prenosnih poti, to je preprečevanje preobremenitev vodov in transformatorjev. Za slovenski EES je bila predlagana rešitev s prerazporeditvijo proizvodnje delovne moči (Grgič 2001). Postopek določanja optimalne prerazporeditve, ki odpravi vse preobremenitve v sistemu, kot tudi njena evaluacija in preverjanje rešitev, zaradi računske zahtevnosti kot tudi zaradi velikega obsega vhodnih podatkov zahteva računalniško podporo.

Tako je Agencija za energijo RS naročila izdelavo računalniškega programa, ki ga bo uporabljala za nadzor in preverjanje odločitev upravljavca prenosnega omrežja o razbremenjevanju kritičnih prenosnih poti ter pri razreševanju morebitnih sporov, ki izvirajo iz zavrnitve dostopa do omrežja. Naročnik je zahteval program s sodobnim uporabniškim vmesnikom, ki bi deloval na platformah MS Windows 98 in MS Windows 2000. Za potrebe podatkov, nujnih za opis elektroenergetskega sistema, je projekt predvideval tudi analizo in izvedbo ustrezne podatkovne baze.

2. MODEL

Ideja razbremenjevanja omrežja s prerazporeditvijo proizvodnje delovne moči je, da na ekonomski osnovi geografsko tako prerazporedimo proizvodnjo, torej jo v enem delu omrežja povečamo in v drugem delu zmanjšamo, da s tem spremenimo pretoke moči po vodih in transformatorjih do te mere, da noben element prenosnega omrežja ni preobremenjen.

V ta namen upravljavec omrežja od proizvajalcev zbira ponudbe za spremembo proizvodnje. Vsak proizvajalec lahko ponudi za svoje generatorje več sprememb proizvodnje, vsako po svoji ceni.

Ker imajo različni generatorji različen vpliv na pretoke moči po preobremenjenih vodih in transformatorjih in so cene teh ponudb načeloma različne, naleti upravljavec omrežja na težavno odločitev, katere ponudbe izbrati, da bo razbremenil vse preobremenitve ob kar najnižjih stroških. Upravljavec omrežja je soočen z optimizacijskim problemom, ki ga lahko strnemo v naslednjih točkah:

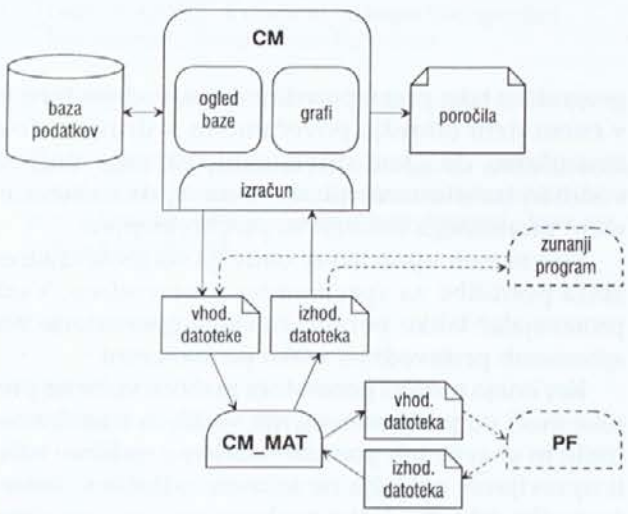
- doseči minimalne stroške, povezane s prerazporeditvijo proizvodnje,
- posameznemu generatorju se sme spremeniti proizvodnjo največ za velikost njegove ponujene spremembe,
- vsi vodi in transformatorji v omrežju morajo biti obremenjeni z njihovo nazivno obremenitvijo,
- ker se skupna poraba v sistemu ne spremeni, mora biti tudi vsota vseh sprememb proizvodnje enaka nič. Če torej nekaterim generatorjem povečamo proizvodnjo, jo moramo drugim zmanjšati.

Podrobnejšo matematično razlago zgornjih točk kot tudi opis celotne metode razbremenjevanja podaja literatura (Grgič 2001).

3. Zasnova programa

Program je zasnovan modularno, slika 1. Osrednji modul (CM) je program z grafičnim uporabniškim vmesnikom, ki omogoča iteracijo z bloki. Sem spada povezava z bazo podatkov in njeno urejanje, izris raznih grafov in histogramov, zagon izračuna optimalne prerazporeditve proizvodnje z namenom razbremenitve sistema, izdelava standardiziranih poročil z rezultati izračunov, itd.

Modul CM je bil razvit v okolju Borland Delphi™ 5.0 (BD™ 5.0), za katerega smo se odločili predvsem zaradi narave problema, ki ne zahteva rešitev, podprtih z večnivojskimi arhitekturami. Program je namreč namenjen delovanju v lokalnem, enouporabniškem okolju, kjer rešitev tipa strežnik-odjemalec povsem zadošča. Poleg omenjenega je pri izbiri razvojnega okolja pripomoglo tudi dejstvo, da BD™ 5.0. omogoča hiter razvoj kakovostnega in funkcionalnega uporabniškega vmesnika.



Slika 1: **Blok shema programa.**

Matematično jedro sistema (CM_MAT) je razvito v okolju Matlab (MathWorks, 1999), ki je zaradi dobre matematične podpore računsko intenzivnim operacijam primernejše od BD™. Programiranje v okolju Matlab poteka preko skriptnega jezika. Matlab omogoča avtomatsko pretvorbo skriptnega programa v C++ izvorno kodo, od tu naprej pa tudi prevajanje programa v samostojno izvršljivo datoteko (EXE). Modul CM_MAT je tako popolnoma samostojen, kar dovoljuje njegovo uporabo neodvisno od Matlaba in obenem postavlja ločnico med računskim delom sistema ter uporabniškim vmesnikom. Komunikacija med računskim delom CM_MAT ter drugimi deli sistema poteka preko vhodno-izhodnih datotek tekstovnega tipa.

3.1 Uporabniški vmesnik

Ker je program izdelan za operacijske sisteme Windows™ 98 in Windows™ 2000, je grafični uporabniški vmesnik prilagojen konvenciji teh okolij. Sestavni del programa je tudi pomoč v standardizirani obliki.

3.2 Baza podatkov

Podatki o parametrih elementov elektroenergetskega sistema, kot tudi urni podatki o topologiji omrežja, proizvodnjah in odjemih moči so shranjeni v relacijski podatkovni bazi, ki temelji na programskem paketu MS Access. Izbiro katerekoli zmogljivejše baze podatkov smo zaradi majhnega obsega podatkov kot tudi zaradi enouporabniške narave sistema ocenili za nepotrebno. Sistem je razvit tako, da omogoča morebiten poznejši prehod na drugo bazo podatkov. Baza podatkov obsega poleg drugega za vsako uro tudi ponudbe proizvajalcev za spremembe njihovih proizvodenj (Grgič 2001). V prihodnosti je predvidena povezava te baze podatkov z glavno bazo podatkov Agencije za energijo RS, v kateri se bodo avtomatsko zajemali podatki iz EES-a.

Zaradi zaupne narave podatkov je dostop zaščiten z geslom. Po prijavi dobi uporabnik možnost pregledovanja, urejanja in dodajanja zapisov v bazo podatkov.

3.3 Izračun optimalne prerazporeditve proizvodnje

Matematični del programa, ki vključuje izračun optimalne prerazporeditve delovne moči z namenom razbremenitve preobremenjenih delov prenosnega omrežja, se izvrši v modulu CM_MAT, slika 1. Zato mora osrednji program CM za izbrana obratovalna stanja iz podatkovne baze pripraviti vhodne datoteke za ta modul. Modul CM_MAT te podatke prebere, preveri njihovo strukturo in izvrši razbremenjevanje ter rezultate zapiše v izhodno datoteko, ki jo lahko nato vnesemo v podatkovno bazo.

Modul CM_MAT je izdelan v razvojnem okolju Matlab z uporabo optimizacijske knjižnice. Program je nato s pomočjo knjižnic Matlab Compiler, C/C++ Math Library in C/C++ Graphic Library ter razvojnega orodja Watcom C/C++ preveden v samostojno aplikacijo. Z uporabo preverjenih optimizacijskih algoritmov, ki so del Matlabove optimizacijske knjižnice, smo zagotovili hiter in zanesljiv izračun optimalne prerazporeditve delovne moči za razbremenjevanje preobremenitev v elektroenergetskih sistemih.

Modul CM_MAT lahko poleg osrednjega programa CM kličejo tudi druge aplikacije, pri čemer je treba upoštevati format vhodno-izhodnih datotek tega programa. S tem smo povečali prilagodljivost programa, saj je lahko matematični del izračuna razbremenjevanja

prenosnih preobremenitev uporabljen v aplikacijah, ki bodo šele razvite.

Metodo razbremenjevanja preobremenitev lahko dodatno izboljšamo z uporabo programa za izračun pretokov moči (Grgič 2001). Zato je program CM_MAT zasnovan tako, da omogoča klic programa za izračun pretokov moči (PF), ki je zaenkrat še v fazi razvoja. Izmenjava podatkov med njima bo potekala preko vhodno-izhodnih datotek.

3.4 Poročila

Program omogoča izdelavo standardiziranih poročil o razbremenjevanju kritičnih prenosnih poti. S tem dobi uporabnik možnost hitre izdelave poročil, ki jih lahko hrani za poznejšo analizo v digitalni obliki ali pa arhivira v papirni obliki.

3.5 Uporaba programa

Upravljanje s programom poteka prek menijev in gumbov v orodni vrstici. Istočasno je lahko odprtih več oken s pregledom različnih podatkov in diagramov.

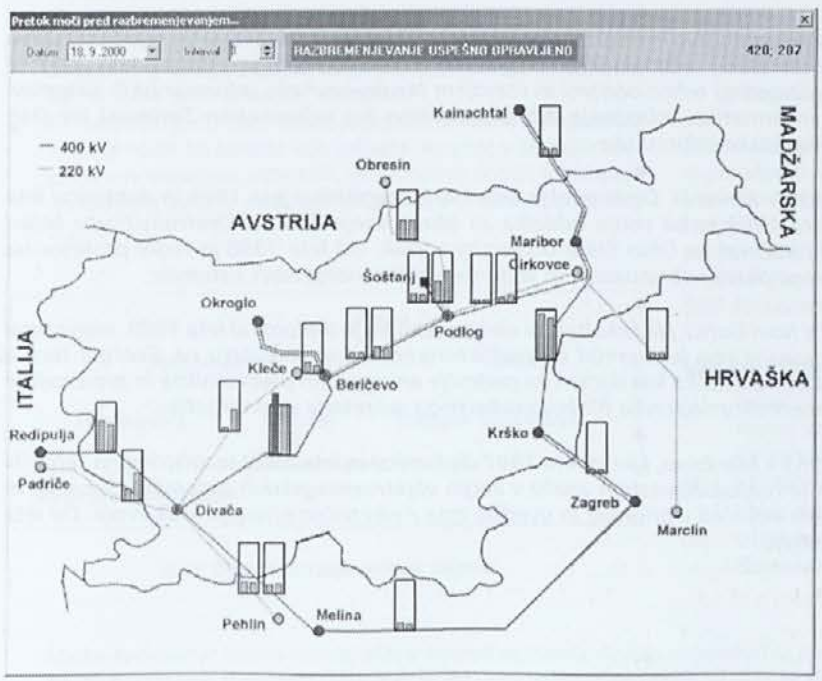
Zamišljeni koncept dela s programom je, da uporabnik najprej analizira izbrana obratovalna stanja, npr. za vse urne intervale trgovanja naslednjega dne. Ker bodo lahko v prihodnosti intervali trgovanja tudi krajši, npr. polurni ali 15 minutni, odvisno od razvoja trga z električno energijo, nastavimo dolžino intervala oziroma število intervalov v enem dnevu v nastavitvah programa.

Pred razbremenjevanjem po izbrani metodi (Grgič 2001) je potrebno od proizvajalcev električne energije zbrati ponudbe za spremembo proizvodnje delovne moči. Analiza teh ponudb je mogoča na dva načina. Prva možnost je pregled v tabelarični obliki, kjer so za izbran časovni interval podane vse ponudbe vseh proizvajalcev. Še bolj zanimiva je druga možnost, ko ponudbe analiziramo v grafični obliki za vsako posamezno elektrarno. V tem primeru izberemo časovni interval in elektrarno, nakar se nam izriše skupna ponudba te elektrarne, ki je v splošnem sestavljena iz več posameznih ponudb. Če je bilo razbremenjevanje za opazovani interval že opravljeno in je bila predlagana sprememba proizvodnje te elektrarne, je sprememba ustrezno označena. Podani so tudi številski podatki o posameznih ponudbah in parametrih krivulje. Tako v tabeli kot tudi v grafičnem pregledu ponudb je omogočeno dodajanje novih ponudb.

Če se izkaže, da bi bilo omrežje za določene časovne intervale preobremenjeno, lahko uporabnik za te intervale požene izračun optimalne prerazporeditve proizvodnje delovne moči z namenom razbremenitve preobremenjenih vodov omrežja. Rezultate lahko po končanem izračunu vnesemo v podatkovno bazo, kjer so na voljo za nadaljnjo analizo oziroma izdelavo poročil.

Ogled rezultatov razbremenjevanja kot tudi pomoč uporabniku pri odkrivanju preobremenitev je uporabniku olajšana s pomočjo grafičnega prikaza preobremenjenih vodov omrežja, slika 2. Za vse vode z vklopljenim prikazom v nastavitvah se izrišeta dva histograma. Prikazana sta v okvirčku, ki je normiran glede na nazivno prenosno kapaciteto voda. Levi histogram prikazuje obremenitev voda pred, desni pa po razbremenjevanju. Če je vod preobremenjen, je histogram obarvan rdeče, njegova višina pa posledično presega velikost okvirčka. Na konkretnem zgledu, slika 2, vidimo, da je pred razbremenjevanjem preobremenjen 400 kV vod Divača-Beričevo ter, da je bilo razbremenjevanje uspešno, saj so po njem vsi pretoki moči v slovenskem EES nižji od njihovih nazivnih vrednosti.

Program omogoča avtomatsko izdelavo nekaj standardiziranih tipov poročil. Na začetku izberemo časovne intervale, za katere želimo izdelati poročila. Če za določen interval razbremenjevanje ni bilo potrebno, se nam v poročilu to samo zabeleži. V primerih, ko je razbremenjevanje uspešno opravljeno, so za vse preobremenjene vode



Slika 2: **Algoritem izboljšane metode z uporabo izračuna pretokov moči.**

v sistemu izpisani pretok navidezne moči pred razbremenjevanjem, maksimalen dovoljen pretok in pretok moči po razbremenjevanju. Temu sledi seznam vseh elektrarn, ki so s svojimi ponudbami sodelovale pri razbremenjevanju, z informacijo o proizvodnji delovne moči pred razbremenjevanjem, angažirano spremembo proizvodnje, proizvodnjo delovne moči po razbremenjevanju in s tem povezanimi stroški. Stroški spremembe proizvodenj delovnih moči so na koncu sešteti za vse elektrarne. V primeru neuspešnega razbremenjevanja so v poročilu navedeni samo preobremenjeni vodi skupaj z začetnimi pretoki navideznih moči in njihovimi maksimalnimi dopustnimi obremenitvami.

4. ZAKLJUČEK

Članek predstavlja računalniški program za načrtovanje ukrepov pri razbremenjevanju kritičnih prenosnih poti po metodi prerazporeditve proizvodnje delovne moči. Za izdelavo programa je bilo potrebno eno leto, naročniku, Agenciji za energijo Republike Slovenije, pa je bil predan v uporabo v prvi polovici leta 2001 vzporedno z uvajanjem trga z električno energijo v Sloveniji.

Trenutno je v teku povezava tega programa s programom za izračun pretokov moči. Poleg tega predvidevamo, da bo naročnik v prihodnosti zagotovil avtomatsko zajemanje podatkov o obratovalnih stanjih elektroenergetskega sistema iz centra vodenja upravljavca prenosnega omrežja.

Program bo služil kot pomoč pri preverjanju in razsojanju v morebitnih sporih, povezanih s preobremenitvami in iz tega izhajajočimi zavrnitvami dostopa do elektroenergetskega omrežja, ravno tako pa bi ga lahko upravljavec prenosnega omrežja uporabljal pri kratkoročnem načrtovanju obratovanja elektroenergetskega sistema.

VIRI IN LITERATURA

D. Grgič, F. Gubina, R. Golob (2001):
»Razbremenjevanje kritičnih prenosnih poti v pogojih trga z električno energijo«, Elektrotehniški vestnik, Letnik 68, št. 2-3.

MathWorks (1999):
»Matlab Optimization Toolbox User's Guide«, Mathworks Inc.; version 2, Natick MA 01760-1500, USA.

Vladimir Batagelj (1998):
»Optimizacijske metode«, skripta v pripravi, http://vlado.fmf.uni-lj.si/vlado/optim/optim.htm.

◆

Dr. David Grgič je diplomiral leta 1995, magistriral leta 1998 in doktoriral leta 2001 na Fakulteti za elektrotehniko Univerze v Ljubljani. Od leta 1995 je zaposlen kot raziskovalni sodelavec na Fakulteti za elektrotehniko, Katedri za elektroenergetske sisteme in naprave. Vodi vaje pri predmetu Obratovanje in načrtovanje EES. Njegovo raziskovalno delo je povezano z obratovanjem elektroenergetskih sistemov.

◆

Dr. Marko Bajec je predavatelj na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Diplomiral je iz računalništva leta 1996. Isto leto se je vpisal na podiplomski študij računalništva. Magistriral je leta 1998, doktoriral pa leta 2001. V okviru Katedre za informatiko se ukvarja z razvojnimi tehnologijami in razvojem ter prenovitvijo informacijskih sistemov. Marko Bajec je član Slovenskega društva Informatika, združenja AIS (Association for Information Systems) ter član programskega odbora posvetovanja Dnevi slovenske informatike.

◆

Dr. Ferdinand Gubina se je rodil 16. maja 1939 v Sloveniji. Diplomiral je leta 1963, magistriral leta 1969 in doktoriral leta 1972 na Fakulteti za elektrotehniko. Od leta 1963 je bil vodja oddelka za obratovanje EES na Elektroinštitutu Milan Vidmar. Leta 1970 je eno leto raziskoval in poučeval na Ohio State University v ZDA. Od leta 1988 je redni profesor na Fakulteti za elektrotehniko. Področje njegovega dela je obratovanje in vodenje elektroenergetskih sistemov.

◆

Dr. Robert Golob se je rodil 23 januarja 1967 v Novi Gorici. Na Fakulteti za elektrotehniko je diplomiral leta 1989, magistriral leta 1992 in doktoriral leta 1994. Po doktoratu je eno leto prebil na podoktorskem izpopolnjevanju na Georgia Tech v ZDA. Po vrnitvi se je zaposlil na Fakulteti za elektrotehniko kot docent za področje energetskih pretvornikov in proizvodnje električne energije. Od leta 1999 s krajšim presledkom opravlja funkcijo državnega sekretarja za energetiko.

◆

Mag. Marko Senčar se je rodil 20. avgusta 1963 v Mariboru, kjer je leta 1987 diplomiral in leta 1992 magistriral na Tehniški fakulteti. Deloval je na področju varnostnih in razpoložljivostnih analiz v večjih elektroenergetskih sistemih v Sloveniji in Švici. Po vrnitvi je bil na pristojnem ministrstvu vključen v pripravo in uvedbo trga z električno energijo v Sloveniji. Od leta 2001 je pomočnik direktorja Agencije za energijo.

SKRB ZA DOMAČE IZRAZJE

Marko Kerševan, Emona Efekta, d. o. o., Ljubljana, marko.kersevan@guest.arnes.si

Prevajanje strokovnega gradiva, posebej s področja informatike in računalništva, je izjemno težavno. To področje je mlada veda, ki se izredno hitro razvija. Zaradi mladosti vede le-ta še nima primerne zgodovine v slovenskem jeziku in s tem tudi še ni usidrana v slovenskem razumništvu, kaj šele v narodu, pa čeprav se že otroci v šoli učijo računalništva. Vendar se je v zadnjem desetletju zadeva precej spremenila. Nastali so novi slovarji in pojmovniki, nekateri so dostopni celo na spletu in odprti za razpravo, v šolah uporabljajo pridobljene slovenske izraze s tega področja in jih zahtevajo celo v diplomah - vse to kaže na dobro zavest za iskanje ustreznih izrazov.

Pri uveljavljanju domačih besed in predvsem pri slovarju je pomembno, da sta pri besedi zapisana tudi razlaga in pomen izraza. V veliko pomoč pri iskanju ustreznih izrazov je tudi drevesna razporeditev pojmov, ki predstavljajo neko področje pomena, v širino in globino, ne samo z gledišča ene stroke (informatike oz. računalništva). Tako npr. za 'hardware' ni dober izraz samo 'računalniška oprema', ampak je to 'računalniška strojna oprema', ker imamo tudi '(računalniško) programsko opremo', samo 'strojna oprema' ne more biti, ker imamo še drugačne strojne opreme; obenem pa naj bo izraz 'računalniška oprema' pomen za 'hardware' in 'software' skupaj, kar je v drevesni razporeditvi teh računalniških izrazov višja raven [3].

Zaradi številnih težav pri slovenjenju tujih izrazov nam lahko pomagajo naslednja izhodišča:

- Vsekakor na silo ni dobro izbrati premalo ustrezen izraz, še manj slabo skovanko.
- Poiščimo domače besede, ki naj bi bile kar najbolj primerne za razumevanje.
- Težko najdemo popolnoma ustrezen slovenski izraz tujki za vse njene pomene. Tujko je možno opisati z več izrazi, ki se naj posamično ustrezno uporabljajo v smislu pomena. Zato moramo vedno vedeti razlago vsakega izraza - pojma, tako za tujko kot za domači izraz. Lahko si pomagamo s pomenskimi izrazi iz več tujih jezikov, ker vsak jezik nekoliko drugače opisuje pomen. Tako kot ima tujka lahko osnovni in preneseni (ali dodani) pomen, ima lahko domača beseda tudi kakšen prenesen (ali dodan) pomen. Na primer: center ni samo središče kroga, je tudi zbir in zbirališče raznovrstnega dogajanja - splošnega ali strokovnega - zato lahko računalniški center nadomestimo z računalniškim središčem.
- Do težav pride tudi zaradi soznačnic (sinonimov) tako v tujem kot v domačem jeziku, ker bi radi našli nedvoumen izraz. Zato so potrebne razlage pojmov, da jih pravilno uporabimo oziroma, da najdemo pravi slovenski ustreznik.
- Pri uveljavljanju domačih besed je povsod potrebna razlaga izraza in na ta način bo beseda laže zaživela. Ko prvič v besedilu uvajamo nov slovenski ustreznik, je primerno dodati v oklepaju uporabljeno tujko. V veliko pomoč je tudi drevesna razporeditev pojmov.

Slika: **Drevesna razporeditev pojmov**

- Iskanje primernega izraza ni vedno lahko in enostavno. Vendar zaradi lenobe ne smemo popustiti. Slovenski jezik je bogat in ima dosti izrazov, le najti jih je treba. Kdor išče - ta najde, velja tudi tukaj. Neverjetno, koliko slovenskih izrazov nam je skritih in pozabljenih. Zaradi lenobe, neosveščenosti in tudi zaradi videza učenosti so nekateri raje uporabili kakšno tujko kot pa lep domač (slovenski) izraz. Tuji izraz je obstal - sedaj ni možno več uveljaviti slovenskega, ker se je tujka že udomačila - pri tem se zdi domači izraz celo smešen (npr. psihologija - dušeslovje, filozofija - modroslovje).
- Tudi če trenutno nimamo genialnih prebliskov za kako novo izpeljanko ali celo za kako dobro skovanko, se trudimo naprej in mislimo o tem - kasneje najdemo ustrezen izraz. Nikakor pa ne očitajmo čistunstva, ker je uporaba tujke pogosto samo odraz lenobe.
- Dosti škode naredijo prvi prevajalci novih besed, ker se ne potrudijo dovolj, da bi strokovno opredelili izraz in ga skušali posloveniti - po pomenu. Tako se slab prevod ali pa kar tujka obdrži in kasneje jo je težko ali nemogoče spremeniti v ustrezen slovenski izraz.
- Izraz, ki ga prvič slišimo in ga ne poznamo, nam je vedno tuj, čeprav je slovenski. Z uporabo (vsaj 10-krat ga na glas ponovi [1] pa nam postane domač, če se le načelno ne otepamo novih izrazov. Tudi razne miselne povezave ovirajo sprejem nekaterih dobrih slovenskih izrazov, ker je že zakoreninjeno, da veljajo nekateri izrazi samo za določeno področje (kot npr.: za izraz "komuniciranje" je najbolj prav slovenski ustreznik občevanje - pa ljudi popade smeh, ko slišijo to povezavo, ker takoj pomislijo samo na spolno občevanje). Z večkratnim ponavljanjem izraza nam le pride v uho kakor katerakoli druga tujka, ki jo takoj prevzamemo, da se čutimo bolj imenitni. Osveščenost za lep jezik je bogatenje duha z domačo besedo, ki jo lahko vsak razume.

Zato bi bilo zelo smiselno, da pri tako majhnem narodu združimo sile in povežemo obstoječe strokovne skupine, ki se ukvarjajo z izrazoslovjem na področju IT, v večjo krovno skupino za IT. Tu naj bi precejšnjo vlogo v povezavi imelo SDI (Slovensko društvo Informatika) s podporo MID (Ministrstvo za informacijsko družbo) ter kot strokovni uradni organ še USM (Urad za standardizacijo in meroslovje, sedaj SIST /Slovenski inštitut za standardizacijo/ - tehnični odbor za IT).

Viri in Literatura

[1] BATAGELJ, Vladimir:
Razvoj slovenskega računalniškega izrazja - osebni pogled,
Uporabna informatika, letnik IX št. 2, Ljubljana 2001

[3] KERŠEVAN, Marko:
Zakaj prevajati, DSI 2002, Portorož 2002

[5] OTER, Mija:
Slovensko računalniško izrazje, DSI 2002, Portorož 2002

Marko Kerševan je tvorno sodeloval že v komisiji za standardizacijo računalništva pri Zveznem zavodu za standardizacijo v Beogradu, nato pa pri slovenskem standardu (SIS), ki je bil na novo ustanovljen pri Ministrstvu za znanost in tehnologijo RS. Deloval je pri standardizaciji RIP, EDIFACT in predvsem pri strokovnem slovenskem izrazoslovju informacijske tehnologije (IT). Vključen je bil tudi v terminološko komisijo pri SAZU za izdelavo slovarja s področja informatike in dokumentacije.

Ivan Turk: Pojmovnik uporabniške informatike

Slovenski inštitut za revizijo, Ljubljana, marec 2002

Prof. dr. Ivan Turk je avtor številnih knjig in razprav, zlasti s področja ekonomike podjetja in računovodstva. Med njegovimi zaslugami je tudi posebna skrb za pospeševanje informacijske kulture v Sloveniji, z vzpostavitvijo domačih strokovnih posvetovanj Ekonomika in organizacija informacijskih sistemov in ureditvijo Pojmovnika poslovne informatike, ki je izšel v založbi Društva ekonomistov Ljubljana leta 1987.

Prof. dr. Ivan Turk se je z jezikom ukvarjal tako rekoč trajno – ne samo kot avtor strokovnih knjig in člankov, temveč tudi kot zbiralec izrazja in avtor slovarskega gradiva v reviji Revizor. V letu 2000 je izdal Pojmovnik računovodstva, financ in revizije. Zdaj je pred nami njegov Pojmovnik uporabniške informatike, v katerem je združil vse svoje dosedanje jezikoslovno delo na področju informatike. Pojmovnik je recenziral dr. Jože Gričar.

Pojmovnik je zgledno urejen. Ima tri dele:

1. Uvod
2. Opredelitve izbranih strokovnih pojmov uporabniške informatike s slovensko-angleškim slovarjem izrazov zanje
3. Angleško-slovenski slovar izrazov za izbrane strokovne pojme uporabniške informatike.

V uvodu opiše avtor naravo in pomen pojmovnika, izbor pojmov in tehnično ureditev pojmovnika. Pojmovnik obravnava področje praktičnega informacijskega delovanja najširšega kroga ljudi. Vsebuje pojme, »s katerimi se srečujejo sodobni uporabniki tehnično obdelovanih in posredovanih informacij«. Sestavljen je torej iz organizacijsko-uporabniškega zornega kota. Dodani so tudi splošni pojmi z drugih področij, ki jih je treba poznati pri delovanju obravnavanega področja.

Paleta izbranih pojmov je torej zelo široka, za vsak pojem pa je praviloma navedenih več slovenskih izrazov, ki jih avtor vrednoti z vrstnim redom navajanja ali z napotilom na drug izraz. Kot pravi avtor v predgovoru, »poskuša pojmovnik zajeti čim več izrazov za posamezne pojme, ponuja pa tudi nekaj novih slovenskih izrazov... Njegov namen ni standardizirati uporabo kakih strokovnih izrazov in opredelitev pojmov, temveč odpirati različne možnosti za izbiro, čas pa bo pokazal, kaj se bo prijelo in kaj bo odpadlo. Zato sem vanj vključil tudi uporabljane izraze, s katerimi se sicer ne strinjam. Strokovni pojmi se razvijajo po svoji vsebini in jezikovne rešitve po svoji obliki, zato je izdelovanje katerega koli pojmovnika trajno nedokončano delo.«

Avtorjeve besede navajamo zato, ker lepo predstavljajo njegovo tolerantnost, pa hkrati tudi skromnost glede na ustvarjeno delo. Med zbranimi izrazi navaja na prvem, prednostnem mestu tistega, ki je po njegovem mnenju vsebinsko najbliže izbranemu pojmu, pogosto pa pogumno ustvari nov izraz. Tako posega tudi med sedanje temeljne informacijske izraze, ki so pogosto nerodni ali celo neprimerni.

Naj navedemo samo nekaj takih izrazov, ki bi jih s pridom lahko zamenjali:

- za strojna oprema <hardware> pojmovnik napoti na *računalniško opremo*
- za operacijski sistem ponuja *obratovalni računalniški programi*
- za informacijski sistem organizacije pa *informacijska ureditev organizacije*.

Morda bi z vsesplošnim dogovorom in posebno skrbjo te izraze in še nekatere druge lahko uvedli v šole in univerze?

Po drugi strani bi avtor želel zamenjati nekatere izraze, ki so se že povsem udomačili. Pri navedbi izraza nas napoti (z znakom →) na svoj predlog, ki mu dodaja razlago pojma. To so predvsem tujke, ki nastopajo tudi pri drugih področjih in dandanes niti nimajo povsem enakega pomena kot predlagane slovenske besede. Po našem mnenju je tukaj bitka izgubljena, vprašanje pa je tudi, ali je smiselna. Nekaj takih primerov je:

sistem → *sestav*, infrastruktura → *podlaga*, analiza → *proučitev, razčlenitev* (SP[1] ima *razčlenjevanje, razčlemba*), koprocesor → *soobdelovalnik*, konfiguracija → *izoblikovanost* (Leksikon[2] navaja *postavitev*).

Ponekod je ponudba izrazov kar prevelika, kar se v praksi verjetno ne bo izkazalo. Numerični je v nekaterih zvezah *številski* (*številski element podatkov*), drugod *številčni* (*številčna spremenljivka*), ali pa *številni* (nabor *številnih znakov*). Pri izrazu alfanumeričen imamo kar dve možnosti: *črkovno-številni*, *abecedno-številni* (SP pa navaja *črkovno-številčen*).

V skrbi za jezik predlaga pojmovnik kot slovenske ustreznice angleškim tudi nekatere po našem mišljenju preveč zapletene izraze. Ne glede na njihovo slovnično pravilnost mnogi od njih slovenskim uporabnikom niti ne bi bili razumljivi; pač pa bi uporabili opisno obliko, verjetno z odvisnim stavkom. Tak primer je *integrirani paket uporabnostnih računalniških programov.*

Razlage so obvezni del vsakega pojmovnika, razen tega pa šele one izraz dokončno opredelijo. Namen avtorja je, na čim krajši način pojasniti izbrane pojme. Kot pravi avtor, so razlage pojmov avtorsko obdelane in ne zgolj prevzete. V te razlage je bilo brez dvoma vloženega veliko dela. Prav v njih se kaže avtorjevo poznavanje področja. Po obsegu so ponekod že bliže leksikografskem obravnavanju. Po potrebi pravilno navajajo večje število uveljavljenih pomenov (npr. za informacijska tehnologija, informatika). Nekatere med njimi pa so kar preveč obsežne, tako da ima uporabnik lahko težave pri razumevanju (informacijska ureditev v podporo skupinskemu odločanju - razlaga zavzema kar 6 vrstic v slovarju, kar bi se dalo rešiti z napotilom k dodatnim pojmom).

Uporabniki imajo po avtorjevem mnenju zdaj še veliko prostosti, ker v slovenščini strokovni izrazi s področja informatike še niso standardizirani. Glede standardiziranja smo še skeptični, predvidevamo pa lahko, da se bo z intenzivnejšo uporabo informatike slovensko informacijsko izrazje v naslednjih letih še bolj izoblikovalo. Pri tem bodo uporabnike koristno usmerjale temeljne publikacije, ki smo jih prejeli v zadnjem času. Pojmovnik uporabniške informatike je brez dvoma ena od teh.

Značilnost Pojmovnika uporabniške informatike, da poskuša zajeti vse obstoječe informacijsko izrazje, namenjeno uporabnikom, in ga dodatno obogatiti, seveda presega naravo običajnih pojmovnikov. Menimo, da je delo potrebno vrednotiti prav s tega vidika. Potrjuje, da je slovenščina bogat jezik in sposobna izraziti vse, kar izraža dandanes angleščina. Brez dvoma se je avtor prav v tem prepričanju lotil obsežnega dela, ki je pred nami; pa tudi v upanju, da bodo njegove predloge in rešitve upoštevali.

Katarina Puc

[1] *Slovenski pravopis, SAZU in ZRC SAZU, Ljubljana 2001*

[2] *Leksikon računalništva in informatike, Založba Pasadena, Ljubljana 2002*

Leksikon računalništva in informatike

Založba Pasadena, Ljubljana 2002, 799 strani

Letos je dala Založba Pasadena Slovencem prvi leksikon računalništva in informatike. Pred njim so izšli že slovarji, geslovniki in pojmovniki informatike, vendar takega leksikona še nismo imeli in že samo to je civilizacijski in kulturni dosežek. Razen leksikona, ki je glavnina knjige, obsega knjiga še spremno besedo, vabilo k sodelovanju in angleško-slovenski slovarček. Slednji je v pomoč, ko iščemo razlage za angleške izraze, ki bi nekatere težko našli, če bi se ravnali le po iz znanja angleščine izhajajoči intuiciji in izrazih, ki smo jih navajeni (tudi že) iz informatike in računalništva. Leksikon je delo skupine strokovnjakov; ima dva urednika, dva glavna avtorja, 15 avtorjev, tri recenzente (ki so obenem tudi avtorji), lektorico in jezikovnega svetovalca. Že samo ta povzetek da slutiti, da je delo nastajalo precej let, saj usklajevanje najprej obveznosti in potem razlag in pogledov tolikšnega števila odličnih strokovnjakov ni niti enostavno niti hitro. V tem pogledu imajo podobni izdelki ene osebe prednost, ker so lahko prej objavljeni in ker se ena oseba sama s seboj pretežno kar strinja. Slabost izdelka enega avtorja je v tem, da novih izdaj ni več, če avtor iz kakršnegakoli razloga z delom ne nadaljuje. Tega se Pasadena zaveda, saj se v spremni besedi urednik poslovi z obljubo naslednje izdaje leksikona.

Napisati objektivno oceno tako obsežnega in tudi temeljitega dela je naloga, ki bi zahtevala čas, primerljiv z onim, ki so ga porabili za svoje delo avtorji. Zato je pričujoči prispevek predvsem osebni pogled, ki pa skuša biti pravičen do ustvarjalcev in ne po nepotrebnem kritičen in še manj napadalen. Za začetek naj ugotovimo, da je v leksikonu zbranih preko pet tisoč glavnih gesel, ki pa niso urejena v razdelke po abecedi, kakor smo navajeni, ampak se kar nadaljujejo ne glede na začetnico. Zbrano je pravo bogastvo strokovnih izrazov in pričakujemo lahko, da se ga bodo posluževali predvsem pedagogi in tako pomagali ustvarjati lep in pravilen slovenski strokovni jezik. S tem bodo tudi ohranjevali slovenščino, kar je brez dvoma eden od najpomembnejših dolgoročnih nacionalnih interesov.

Poglejmo najprej, kaj je knjiga, ki jo držimo v rokah, da je ne bi po krivici merili z napačnim vatlom. Leksikon[1] je besedni (pojmovni) priročnik, ki po abecedi razvrščena gesla pojasnjuje manj obširno kakor enciklopedija[2], ki je strnjen pregled človeškega znanja (v tem primeru) področja ali stroke, vendar obširneje od slovarja[3], ki se ukvarja zgolj z besediščem. Avtorji in založba se, kakor je videti, niso mogli odločiti za naravo dela in opazen del gesel je obdelanih na slovarski način, kar ni nič narobe, vendar glede na naslov knjige tega ne pričakujemo. V leksikon bi spadali pač samo sprejeti izrazi. Delo v nekaterih razlagah zahteve leksikona bistveno presega in celo za enciklopedijo so nekatera gesla zavidljivo obširno pojasnjena (npr. *večpredstavnost, zgodovina računalništva, operacijski sistem, informacijska družba, internet*). To sicer ni slabo, vendar ne vemo, po kakšnem kriteriju so bila določena gesla deležna tolikanj obširne razlage. Nekoliko moteče je, da za gesla posebej ali vsaj za leksikon v celoti ni navedbe virov razlag, definicij in pojmov. Ker tega ni, ne vemo, ali je leksikon od prve do zadnje črke izvirno delo ali pa so nekatera gesla le prevedena in razlage prevzete. Nekateri znaki nas navajajo k mnenju, da gre morda v preveč primerih za iskanje izvirnih opisov in razlag namesto že sprejetih in uveljavljenih definicij, kar bralca bega. To je nerodno, saj leksikona ne prebiramo kot leposlovno delo, temveč ga vzamemo v roko pri strokovnem delu.

Ker je nemogoče obdelati vse izraze, ki nastopajo v leksikonu, si za prvo oceno in vtis navadno pomagamo z osnovnimi izrazi področja, ki ga pokriva. Glede na to, da gre za informatiko in računalništvo, nas čaka pri osnovah kar nekaj presenečenj. Za začetek: gesla *podatek* ne najdemo. Najdemo geslo *podatki* z angleškim izrazom *data*, glede definicije (ali opisa) pa smo lahko vsaj skeptični, saj naj bi bili podatki *določeni deli informacije*. To se ne ujema z mednarodno sprejeto definicijo in je v nasprotju z uveljavljenim opisom, da so podatki nosilec informacije. Da nastopajo le v množini, je mogoče posledica angleškega gesla, vendar moramo znati tudi nekoliko angleško, da bi bil prevod pravilen. V slovenščini ima *podatek* množino - *podatki*, vendar ima v angleščini *data* tudi ednino - *datum*! Res je, da se *data* skoraj izključno uporablja kot edninski samostalnik, s čimer se pa seveda ne more odpraviti slovenske slovnice kar tako. Res je tudi, da je v angleščini meja med *data* in *information* manj stroga kot v slovenščini, ker se izraza uporabljata skoraj kot sopomenki, vendar tudi v angleščini strogo vzeto velja, da so podatki nosilci informacij. Geslo *informacija* je pojasnjeno (zelo ohlapno) kot *podatek oziroma množica podatkov, ki določeni skupini ljudi nekaj pomeni*. Če odštejemo dejstvo, da v definicijah beseda *oziroma*, ki nejasnost le povečuje, nima kaj iskati, se moramo vprašati, ali podatek ni več nosilec informacije? Ali informacija ne povečuje več znanja? Ali ne nastaja več v relaciji med podatki? In ne nazadnje, ali se je Rajko Jamnik[4] trudil zaman, ko je učil, da nastane informacija ob poizkusu? Iz opisa *informatike* je obdelava podatkov izpadla in le v oklepaju se pojavi kot *obdelava informacij*, kar je dvomljivo, če so nosilci informacij podatki. V *računalništvu* je tudi ni, nastopa pa kot samostojno geslo, kjer je izhodiščni pojem *uporaba računalnikov*, potem *veda o računalniškem upravljanju podatkov* in končno (pod 3) kot *sopomenka za računalništvo*. Tudi prav, vendar je to precej inovativen, da ne rečemo novotarski

[1] *po Leksikonu Cankarjeve založbe, Ljubljana 1998*
[2] *isto*
[3] *isto*
[4] *Rajko Jamnik, Elementi teorije informacije, Knjižnica Sigma, Ljubljana 1964*

pristop, ki od klasičnih opisov, če jih lahko tako imenujemo, opazno odstopa. Doslej smo jo šteli za disciplino znotraj informatike in računalništva, zdaj se pa pojavi kar samostojno. Če bi hoteli biti dosledni, bi morali naslov leksikona razširiti še z obdelavo podatkov. *Podatkovno bazo* iščemo neuspešno; najdemo le *banko podatkov*. Najdemo tudi *bazo podatkov*, pa vendar obstajata obe in obstaja tudi pomenska razlika med njima. *Datoteko* najdemo (z angleškim izrazom *file* in ne – ali vsaj tudi - kot *data file*), ampak s preskromnim opisom, ki niti ne namigne na dejstvo, da se vsi podatki v datoteki nanašajo na eno entiteto.

Ob tem, da v leksikonu imenoma nastopajo gospodarske družbe, npr. Oracle, Intel, IBM in druge, se vprašamo, kaj je bilo merilo za vključitev med gesla. Burroughs, ki ga ni več, najdemo, za Sperry Rand na primer, ki je imel v preteklosti kar vidno mesto, pa ni bilo prostora. V tej zvezi nas nekoliko preseneča, da med gesli, ki se začno s številko (in ki bi jih kakor tudi kratice pričakovali v posebnem razdelku), najdemo 386 in sorodnike in izvedenke, ne pa tudi 360 (ali /360 ali na drugem mestu S/360) in 370 (/360, S/360), kar je glede na vlogo in prispevek IBM na področju računalništva v šestdesetih in sedemdesetih letih že kar manjša krivica.

Leksikon je nekako kanoniziral *strojno opremo* in *programsko opremo*, čeprav gre, kot je razumeti iz angleških izrazov, za računalniške naprave in programe in ne za njihovo opremo. Glede tega bi se le kazalo nekoliko poglobiti v oblikoslovje slovenskega besedišča in vsaj poizkusiti uporabiti analogije iz drugih področij z daljšo tradicijo. Hitro bi uvideli, da je programska oprema nekaj, s čimer se opremljajo programi (npr. pri zalaganju in izdajanju) in da je to kvečjemu oprema računalnika. Podobno je strojna oprema nekaj, s čimer so opremljeni stroji, oprema strojev torej in sploh ne nujno računalnikov, tako kot je avtomobilska oprema nekaj, s čimer se opremi avtomobil; kaj šele, da bi bili to računalniki in spremljajoče naprave.

Ob listanju po leksikonu nas nekoliko tudi navdaja občutek, da je pri pisanju prevladoval skoraj puristične pristop pri pojasnjevanju in opisih gesel. Kako naj si drugače razlagamo to, da je v zvezah z *objektnim* prednostna oblika *predmetno*? Pa vendar objekta ne preganjata niti slovar slovenskega knjižnega jezika[5] niti pravopis[6]. Še več, vemo celo, da ima včasih tujka drugačen, lahko širši ali ožji pomen kakor slovenska beseda. *Objekt* je že tak primer, da ima širši pomen kakor *predmet* in verjetno gradbeniki zato v zvezi z mostovi, podvozi, zgradbami govorijo o objektih in ne o predmetih, kot so na primer lopate, brez katerih pri gradnji objektov seveda ne morejo[7]. Podoben primer je *soprocesor* – mar koprocesor ni več dovolj lep? Menili bi, da je, saj imamo še vedno koordinate in ne soordinat in končno so nas tudi učili, da so skovanke, ki so pol tujke pol slovenske besede, manj primerne od čisto iz tujk ali čisto slovenskih izrazov narejenih besed. Tudi *geografski informacijski sistem* je zapostavljen na račun (bolj slovenskega?) *zemljepisnega informacijskega sistema*. Purizem je mogoče zaznati tudi pri rabi slovenskih besed: pri pojasnilu *strojne napake* je podan primer, vendar je naveden kot *zgled*, kar je vsaj sporno, saj naj se po napaki ne bi zgledovali, četudi jo pojasnimo s primerom. Opazimo tudi močno željo po novih izrazih: na primer pri *rokavici VR* so v pojasnilu nepotrebna *zaznavala*, čeprav je (npr. v merilni tehniki) že uveljavljen izraz *tipalo*. Nekoliko na silo poslovenjen se zdi tudi *oprimek* (za angleški *handle*), ker smo ga bili doslej navajeni pri plezanju (angleški *grip*) in ne v računalništvu. Res je ponujena tudi *ročka*, ki pa nas spomni prej na posodo in vprašamo se, ali bi *držalo* in *roč(aj)*, ki ju že dolgo poznamo, v informatiki res toliko manj povedala, da ju moramo kar odpraviti.

Res je, da novi izrazi jezik bogatijo, ampak pomislimo na to, da inflacija siromaši. Gotovo pa je tudi res, vsaj kar zadeva jezik, da je treba preizkusiti in predstaviti precej možnosti, da bi se lahko prijeli najprimernejši izrazi. Avtorjem je treba priznati, da so se v tem pogledu res trudili in pogosto tudi uspevali: za *cluster* so ponudili zelo lepo možnost *gruča* in toplo upamo, da se bo hitro prijela, Težava takega pristopa pa je zlasti v tem, da nepotrebno troši moči in čas za popravljanje že uveljavljenih izrazov, s katerimi pravzaprav ni nič kaj hudo narobe in mogoče bi bilo treba napraviti prednostni seznam besed, ki jih je treba ustvariti ali poiskati na novo. Nemara gre pri ponudbi novih izrazov nekoliko tudi za spomenike, saj le kdo si jih ne želi in tisti v jeziku so trajnejši celo od marmornih, najbolj spoštovani pa so vendar tisti, ki nam jih postavijo drugi.

Kaj naj rečemo za konec? Ne glede na vse kritične misli je treba priznati avtorjem, urednikom, recenzentom in založbi, da so opravili ogromno delo, ki je zahtevalo tudi kar precej poguma, česar se zavedamo vsi, ki smo kdaj kaj objavili, saj je raztrgati neskončno laže kot ustvariti. Leksikon smo Slovenci potrebovali in vsaka čast vsem, ki so k temu prispevali. Nobena beseda v tem zapisu nima namena zmanjševati vrednosti dosežka ali zaslug sodelujočih. Pred nami je dragulj, dolžnost vseh nas pa je, da ga pomagamo zbrusiti in to je tudi edini namen pričujočega prispevka. Resnično upamo na svidenje ob naslednji izdaji.

Niko Schlamberger

---

[5] *Slovar slovenskega knjižnega jezika, DZS, Ljubljana,1994*

[6] *Slovenski pravopis, ZRC, SAZU, Ljubljana 2001*

[7] *Nasprotno ima v dvojici promet - transport širši pomen domača beseda.*

DEVETO POSVETOVANJE

DNEVI SLOVENSKE INFORMATIKE 2002

INFORMATIKA KOT PRILOŽNOST IN IZZIV

Kongresni center Grand hotel Emona, Portorož
17. – 19. april 2002
www.drustvo-informatika.si

■

Organizator: Slovensko društvo INFORMATIKA
Soorganizatorja: Gospodarska zbornica Slovenije - Združenjem za informatiko in telekomunikacije, Infos d.o.o.

Letos je bilo v tradicionalnem pomladanskem okolju slovenskega Primorja organizirano že deveto nacionalno posvetovanje Dnevi slovenske informatike (DSI). Posvetovanja se je udeležilo več kot 400 ljudi, nekateri kot avtorji prispevkov, drugi kot poslušalci. Rdeča nit letošnjega posvetovanja je bila Informatika kot priložnost in izziv Sloveniji. To pomeni, da nam informatika vsak dan ponuja možnosti in priložnosti za večjo uspešnost poslovanja, hkrati pa pomeni izziv, kajti pomembnih informacij za nove razvojne načrte ni nikoli preveč.

Pomen posvetovanja

Posvetovanje prirejamo informatiki z namenom, da se srečamo, izmenjamo informacije in spoznanja ter jih posredujemo javnosti. Namenjeno ni le informatikom, temveč vsem, ki se srečujejo z uporabo informacijske tehnologije. Veliko je tudi priložnosti za vzpostavljanje poslovnih stikov. Posebna odlika DSI je poudarek na strokovni vsebini, zato je posvetovanje s tem upravičeno dobilo in ohranilo sloves najpomembnejšega slovenskega strokovnega dogodka na področju informatike.

Program posvetovanja

DSI 2002 so ohranili programsko shemo prejšnjih let, ki se je uveljavila in potrdila. Program je potekal v obliki sekcij, okroglih miz in delavnic. Vodili so jih najvidnejši slovenski informatiki iz univerz, gospodarstva in javne uprave. Letošnja novost je bila predstavitev projektov, ki jih je financiralo Ministrstvo za informacijsko družbo. Udeležil se je je tudi minister za informacijsko družbo dr. Pavel Gantar. Popestritev posvetovanja je bil razstavni del, kjer so se na enem delu predstavili nekateri pokrovitelji, na drugem pa so bili razstavljeni eksponati iz Računalniškega muzeja.

Razmeroma stalne sekcije omogočajo časovno sledenje razvoja določenega področja, potrebne, vendar ne radikalne dopolnitve in spremembe, pa so jamstvo, da se v programu posvetovanja odraža tudi razvoj informatike. Program letošnjega posvetovanja je obsegal deset sekcij, to so:

A. Mednarodna sekcija
B. Metodologija, informacijska tehnologija in pristopi
C. Poslovna informatika in povezovanje poslovnih sistemov
D. Internet in tehnologija elektronskega poslovanja
E. Informacijske rešitve in uvajanje IS
F. Izobraževanje in usposabljanje na področju informatike
G. Operacijska raziskovanja
H. Strokovni jezik
I. Sociološki, pravni in etični vidiki informatike
J. Študentska sekcija

V okviru sekcij so strokovnjaki iz slovenskih in tujih podjetij ter ustanov predstavili 85 prispevkov, ki so objavljeni v zborniku posvetovanja. Tudi letos smo organizirali mednarodno sekcijo, kjer so predavali strokovnjaki iz Hrvaške, Slovaške, Madžarske, Avstrije, Italije, Grčije ter Slovenije. Posebno veliko zanimanje je bilo za sekcije B, C, D in E, pomen pa pridobiva tudi študentska sekcija, saj želimo med študenti informatike zbuditi čim večje zanimanje za posvetovanje in jim tako preko spoznavanja problemov, s katerimi se informatiki srečujejo v praksi, omogočiti poglabljanje znanja.

Veliko zanimanja je bilo za vabljena predavanja predavateljev iz Slovenije, Finske in Danske:

- The Extensive Role of Reuse in Software Engineering (dr. Hannu Jaakkola, Finska)
- Realnost virtualnih organizacij (dr. Cene Bavec, Slovenija)
- Strateško načrtovanje - kaj več kot izgubljanje časa (Peter Menhard, Danska)
- Zgodba o milijardi dolarjev in kako smo prenovili poslovne procese v Oraclu (Žiga Vaupot, Slovenija)
- Microsoft .NET - platforma za sodobno e-poslovanje (Rok Palčič, Slovenija)

Na okroglih mizah so bila obravnavana zaokrožena tematska področja, način obravnave pa je omogočil tudi udeležencem, da so se vključili v razprave z mnenji in kritikami. Letos smo organizirali štiri okrogle mize:

- Možnosti informacijskih podjetij za pridobitev sredstev Evropske unije
- Začetki računalništva in informatike na Slovenskem
- Celovite programske rešitve (ERP): mit ali resničnost?
- Trendi spletnih predstavitev

Vsem, ki so se želeli seznaniti z nekaterimi novostmi in tako dobili konkretne napotke za svoje delo, sta bili namenjeni delavnici:

- Organizacija sodobne pisarne
- Mobilna informacijska prihodnost

Posvetovanje sta kot priložnost za sestanek uporabila dva prireditelja:

- Gospodarska zbornica Slovenije - Združenje za informatiko in telekomunikacije je sklicalo sejo upravnega odbora
- predstavniki Slovenskega društva INFORMATIKA pa so se kot člani International Federation for Information Processing (IFIP) in soustanovitelji mednarodnega regionalnega odbora IT STAR udeležili sestanka tega odbora, na katerem je bil prisoten tudi njegov predsednik, dr. Walter Grafendorfer.

Strokovni del posvetovanja so popestrili tudi trije družabni dogodki, katerih izvedbo so omogočili generalni pokrovitelj Mobitel ter pokrovitelja dneva Microsoft in Oracle. Vabilu k sodelovanju se je odzvalo še veliko pokroviteljev iz vrst najpomembnejših slovenskih podjetij s področja informatike, brez pomoči katerih posvetovanja ne bi mogli organizirati.

Posvetovanje smo zaključili s podelitvijo priznanj glede na mnenje udeležencev in nagrad. Nagrado za najzanimivejši prispevek z naslovom *Nevronske mreže* je prejela Teja Batagelj, za najaktualnejši prispevek *Realnost virtualnih organizacij* dr. Cene Bavec, najbolje pa je po mnenju udeležencev svoj prispevek *Evidentiranje medicinsko-tehničnih pripomočkov na kartici zdravstvenega zavarovanja* predstavila Anka Bolka. Letos smo prvič podelili tudi nagrado za najboljši študentski prispevek, prejele so jo Bojana Lobe, Marjeta Leskovšek in Marjeta Novak. Na zaključku smo izmed prisotnih udeležencev tudi izžrebali tri, ki smo jim podelili nagrade.

Prepričani smo, da so se Dnevi slovenske informatike tudi letos izkazali kot zanimiv dogodek. Ker želimo posvetovanje še bolj približati željam udeležencev, z veseljem sprejemamo predloge za izboljšave.

dr. Mojca Indihar Štemberger,
predsednica organizacijskega odbora
dr. Ivan Vezočnik,
predsednik programskega odbora

10th EADI General Conference	19. - 21. 09. 2002	Ljubljana, SI	European Association of Development, Bonn Inštitut za ekonomska raziskovanja, Ljubljana	stanovnikp@ier.si
Forum on Specification & Design Languages FDL 2002	24. - 27. 09. 2002	Marseille, FR	ECSI, IFIP WG10.5 5th IEEE/IFIP	villar@teisa.unican.es http://www.ecsi.org/fdl02 Fax: +34 42 201873
Work. Conf. on Communication and Multimedia Security CMS 2002	26. - 27. 09. 2002	Portorož, SI	IFIP TC6/TC11, IJS	centre@setcce.org http://www.setcce.org/cms2002 Fax: +386 1 4232118
2nd IFIP Conference on E-Commerce, E-Business & E-Government	7. - 9. 10. 2002	Lisbon, PT	IFIP TC6/TC8/TC11, API, APDC, ACEP	lavt@civil.ist.utl.pt Fax: +351 21 8409884,
IFIP Workshop on Internet Technologies, Applications & Societal Impact WITASI 2002	10. - 11. 10. 2002	Wroclaw, PL	IFIP WG6.4	grzech@ists.pwr.wroc.pl http://www.ists.pwr.wroc.pl/witasi/ Fax: +48 71 320332848
14th Symp. on Comp. Arch. & High Performance Computing SBAC-PAD 2002	28. - 30. 10. 2002	Vitoria, BR	SBC, IFIP WG10.3, IEEE	sbac2002@inf.ufes.br http://www.inf.ufes.br/sbac2002/
IST 2002 Partnership for the Future	4. - 6. 11. 2002	Copenhagen, DK	IST, European Commission, Danish EU Presidency	http://www.voco.dk
Intl. Conf. on Formal Techn. f. Networked & Dist. Syst. FORTE 2002	11. - 14. 11. 2002	Houston,TX, US	IFIP WG6.1 5th IFIP WG11.5	vardi@cs.rice.edu http://www.ece.utexas.edu/FORTE/
Work. Conf. on Integrity and Internal Control in Information Systems IICIS 2002	11. - 12. 11. 2002	Bonn, DE	IFIP WG11.5 IFIP WG9.6/11.7	gertz@cs.ucdavis.edu http://sirius.cs.ucdavis.edu/IICIS2002/ Fax: +49 228 734382
Work. Conference on Global and Organizational Discourse about Information Technology	12. - 14. 12. 2002	Barcelona, ES	IFIP WG8.2	andreu@iese.edu http://is.lse.ac.uk/staff/whitley/resources/discourse Fax: +34 93 2534343
Intl. Conf. on High Performance Computing HiPC	18. - 21. 12. 2002	Bangalore, IN	ACM, IEEE/IFIP WG10.3	http://www.hipc.org/hipc2002/ Fax: +1 213 740 4418
Work. Conf. on Quality Education at a Distance QED	3. - 6. 02. 2003	Geelong, AU	IFIP WG3.6	http://education.deakin.edu.au/ifipwg3.6 estacey@deakin.edu.au Fax: +61 3 92446834
18th IFIP Intl. Information Security Conf. SEC 2003	26. - 28. 05. 2003	Athens, GR	IFIP TC11, GCS	mka@aueb.gr http://www.sec2003.org

Pristopna izjava

Želim postati član Slovenskega društva Informatika

Prosim, da mi pošljete položnico za plačilo članarine SIT 5.200 (kot študentu SIT 2.400) in me sproti obveščate o aktivnostih v društvu.

(ime in priimek, s tiskanimi črkami)

(poklic)

(domači naslov in telefon)

(službeni naslov in telefon)

(elektronska pošta)

Datum: Podpis:

Včlanite se v Slovensko društvo INFORMATIKA.
Članarina SIT 5.200,- (plačljiva v dveh obrokih) vključuje tudi naročnino za revijo Uporabna informatika.
Študenti imajo posebno ugodnost: plačujejo članarino SIT 2.400,- in za to prejemajo tudi revijo.

Izpolnjeno naročilnico ali pristopno izjavo pošljite na naslov:

Slovensko društvo INFORMATIKA, Vožarski pot 12, 1000 Ljubljana.

Lahko pa izpolnite obrazec na domači strani društva
http://www.drustvo-informatika.si

■■■

INTERNET ■ INTERNET ■ INTERNET ■ INTERNET ■ INTERNET ■ INTERNET

Vse člane in bralce revije obveščamo, da lahko najdete domačo stran društva na naslovu:

http://www.drustvo-informatika.si

Obiščite tudi spletne strani mednarodnih organizacij, v katere je včlanjeno naše društvo:
IFIP: **www.ifip.or.at,** ECDL: **www.ecdl.com,** CEPIS: **www.cepis.com**

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

Naročilnica

Naročam(o) revijo UPORABNA INFORMATIKA

- ❑ s plačilom letne naročnine SIT 4.600
- ❑ . . . . . izvodov, po pogojih za podjetja SIT 13.800 za eno letno naročnino in SIT 8.900 za vsako nadaljnjo naročnino
- ❑ po pogojih za študente letno SIT 2.000

Naročnino bom(o) poravnal(i) najkasneje v roku 8 dni po prejemu računa

_____

(ime in priimek, s tiskanimi črkami)

_____ _____

(podjetje) *(davčna številka)*

_____

(ulica, hišna številka)

_____

(pošta)

Datum: Podpis:

UPORABNA INFORMATIKA

ISSN 1318-1882

Ustanovitelj in izdajatelj:
Slovensko društvo Informatika, 1000 Ljubljana, Vožarski pot 12

Glavni in odgovorni urednik:
Mirko Vintar

Uredniški odbor:
Vesna Bosilj Vukšić, Dušan Caf, Aljoša Domijan, Janez Grad, Milton Jenkins, Andrej Kovačič, Tomaž Mohorič, Katarina Puc, Vladislav Rajkovič, Heinrich Reinermann, Ivan Rozman, Niko Schlamberger, John Taylor, Ivan Vezočnik, Mirko Vintar

Tehnična urednica: Katarina Puc

Oblikovanje: Zarja Vintar, Dušan Weiss, Ada Poklač
Naslovnica: Bons

Tisk: Prograf
Naklada: 700 izvodov

Revija izhaja četrtletno. Cena posamezne številke je 3.500 SIT.

Letna naročnina za podjetja SIT 13.800, za vsak nadaljnji izvod SIT 8.900.
Letna naročnina za posameznika SIT 4.600, za študente SIT 2.000.

Celotni Oraclov E-Business Suite.

	Oracle E-Business Suite
Marketing	✓
Spletna trgovina	✓
Prodaja	✓
Podpora uporabnikom	✓
Nabava	✓
Dobavna veriga	✓
Finance	✓
Človeški viri	✓
Aplikacijski strežnik	✓
Podatkovni strežnik	✓

Oraclove rešitve so razvite za povezano delovanje.

Aplikacije različnih proizvajalcev zahtevajo sistemsko integracijo.

Sistemska integracija stane veliko več kot sama programska oprema.

Razmislite o tem.

ORACLE®

SOFTWARE POWERS THE INTERNET™

www.oracle.si

Razprave

Jasmin Malkič
Remote Database Access with Use of
Java Security Functions

Miroslav Ribič, Andrej Kovačič
Integracija informacijskih virov - portalna arhitektura

Denis Trček
Tehnologije agentov - uporaba v elektronskem poslovanju

Alojz Tapajner, Peter Kokol
Uporaba inteligentnih sistemov

Robert T. Leskovar, József Györkös, Ivan Rozman
Gradnja hierarhične strukture konceptov iz
nestrukturiranih tekstovnih dokumentov

Poročila

Uroš Sajko
Clearcase - orodje za upravljanje konfiguracije

Rešitve

David Grgič, Marko Bajec, Ferdinand Gubina,
Robert Golob, Marko Senčar
Računalniško podprto načrtovanje ukrepov
za razbremenitev kritičnih prenosnih poti