DATAKON 2011

Příspěvky

Editoři

Jaroslav Zendulka
Marek Rychlý

Mikulov, Hotel Eliška
Česká republika
15. - 18. října 2011
http://www.datakon.cz

DATAKON® je prestižní česká a slovenská konference s mezinárodní účastí zaměřená na teoretické a technické základy, nejlepší postupy a vývojové trendy v oblasti využití informačních technologií při budování informačních systémů včetně výsledků jejich aplikace v praxi.

DATAKON® představuje ideální platformu pro výměnu zkušeností mezi českými i zahraničními odborníky z řad dodavatelů informačních technologií, jejich zákazníků a akademického světa.

DATAKON® oslovuje zkušené odborníky i nejlepší studenty.

Vydává: Vysoké učení technické v Brně, 2011
Tisk: SDRUŽENÍ MAC, spol. s r.o., U Plynárny 85, Praha 10

ISBN 978-80-214-4329-7

DATAKON 2011

Proceedings

Edited by

Jaroslav Zendulka
Marek Rychlý

Mikulov, Hotel Eliška
Czech Republic
October 15-18, 2011
http://www.datakon.cz

DATAKON® is a high-profile traditional conference focused on theoretical and technical background, best practices and development trends in deployment of information technology for information systems development including application results of described approaches in industry practice.

DATAKON® serves as an ideal platform for experience exchange among experts of information technology products and services suppliers, their customers and the academic community both Czech, Slovak and also foreign.

DATAKON® brings together researchers, professionals, and students.

Published by Brno University of Technology, 2011
Printed by SDRUŽENÍ MAC, spol. s r.o., U Plynárny 85, Praha 10

ISBN 978-80-214-4329-7

Předmluva

DATAKON® je prestižní česká a slovenská konference s mezinárodní účastí zaměřená na teoretické a technické základy, nejlepší postupy a vývojové trendy v oblasti využití informačních technologií při budování informačních systémů včetně výsledků jejich aplikace v praxi. Nosnými tématy ročníku 2011 jsou

- Architektury informačních systémů
- Cloud computing
- Dobývání znalostí
- Informační bezpečnost
- Integrace datových zdrojů
- Kvalita dat a zpracování neúplné informace
- Management znalostí a znalostní technologie
- Modelování procesů a služeb
- Multimediální, časová, časově prostorová a geografická data
- Nové trendy a moderní databázové technologie
- Servisně orientované architektury
- Sociální sítě na internetu
- Vyhledávání na webu
- XML technologie
- Zpracování rozsáhlých souborů dat
- Proudy dat
- Řídicí systémy v reálném čase
- Datové registry

Struktura sborníku odpovídá programu konference DATAKON 2011, která se bude 15. až 18. října 2011 v Mikulově, v hotelu Eliška. Výběr příspěvků zajišťoval programový výbor pro rok 2011. Všechny příspěvky byly posuzovány třemi nezávislými recenzenty. Z celkového počtu 20 podaných standardních příspěvků a dvou případových studií vybral programový výbor 9 standardních příspěvků a obě případové studie. Čtyři příspěvky byly doporučeny k přijetí jako postery. Na základě výzvy pro podání pozdních posterů byl přijat ještě jeden pozdní poster s redukovaným rozsahem.

Tato kniha příspěvků obsahuje texty čtyř zvaných přednášek, devíti přijatých standardních příspěvků, dvou přijatých případových studií, dvou případových studií partnerů konference (IBM Česká republika spol. s r.o. a KOMIX s.r.o.), čtyř posterů a jednoho pozdního posteru.

Samostatnou publikaci tvoří kniha tutoriálů, která obsahuje čtyři tutoriály prezentované na konferenci. Jde o původní texty zaměřené na škálovatelná řešení pro zpracování velkého objemu dat, servisně orientovanou architekturu, geolokaci a multiagentní systémy s jasně vymezenou problematikou. Tutoriály byly recenzovány členy programového výboru.

Závěrem je třeba poděkovat všem, kteří se zasloužili o vznik tohoto ročníku konference DATAKON a této publikace. V prvé řadě chci poděkovat autorům zvaných přednášek, tutoriálů, standardních příspěvků, případových studií a posterů, za úsilí, které vynaložili při jejich přípravě. Rovněž bych chtěl poděkovat členům programového výboru za jejich nápady a práci při přípravě programu konference. Dále chci poděkovat partnerům konference za jejich podporu při přípravě konference. Velký dík patří také organizačnímu výboru konference, zejména Anně Kotěšovcové, Marku Rychlému a Petru Šalounovi, za přípravu a zajištění průběhu konference DATAKON 2011.

V Brně, září 2011
Jaroslav Zendulka
předseda programového výboru

Preface

DATAKON® is a high-profile traditional conference focused on theoretical and technical background, best practices and development trends in deployment of information technology for information systems development including application results of described approaches in industry practice. This year the following main topics have been chosen:

- Information systems architecture
- Cloud computing
- Data mining
- Information security
- Data sources integration
- Data quality and incomplete information processing
- Knowledge management and technologies
- Process and service modeling
- Multimedia, time, time space and geographic data
- New trends and modern database technologies
- Service oriented architecture
- Social networks on the Internet
- Searching on the Web
- XML technology
- Very large data sets processing
- Data streams
- Real-time systems
- Data registers

The structure of the book corresponds to the DATAKON 2011 conference program. DATAKON 2011 will be held on 15th–18th October 2011, in Mikulov, Czech Republic. The Program Committee carried out the selection of papers. All papers were reviewed in advance by three reviewers. All papers were judged only on their own merits, independent of other submissions. Finally, nine standard papers and two case studies of the total number of 20 submitted standard papers and two case studies were selected for publication. Four submissions were accepted as posters. Based on the call for late posters, one more poster with a reduced size was accepted.

This book comprises four invited lectures, nine accepted standard papers, two case studies, two case studies of the industrial partners of the conference (IBM Czech Republic and KOMIX companies), four posters and one late poster presented on the conference.

Tutorials presented on the conference are published in a separate book. It contains four tutorials focused on scalable solutions for large data volumes processing, service-oriented architectures, geolocation and multiagent systems.

I would like to express my acknowledgements to all people who prepared the DATAKON 2011 conference. I would like to thank the authors of invited lectures, tutorials, standard papers, case studies and posters papers submitted to DATAKON 2011 for their efforts to prepare them. I would also like to thank all the Program Committee members for their excellent work during discussions on the topics of the conference, reviewing process and the conference program. I also wish to acknowledge to the conference partners for their support. My special thanks go to the members of the organization committee, namely to Anna Kotěšovcová, Marek Rychlý and Petr Šaloun. Without their assistance and excellent work the DATAKON 2011 conference could not have been possible.

Brno, September 2011
Jaroslav Zendulka
Program Committee Chair

Organizace konference (Conference Organization)

Řídící výbor (Steering Committee)

Předseda (Chair): Jaroslav Pokorný, MFF UK Praha
Členové (Members): Mária Bieliková, FIIT STU Bratislava
Ján Genči, FEI TU Košice
Jiří Gregor, GALEOS a.s.
Petr Hujňák, Per Partes Consulting Praha
Dušan Chlapek, FIS VŠE Praha
Karel Richta, MFF UK Praha
Jan Staudek, FI MU Brno
Petr Šaloun, FEI VŠB-TU Ostrava
Jaroslav Zendulka, FIT VUT Brno

Programový výbor (Program Committee)

Předseda (Chair): Jaroslav Zendulka, VUT Brno
Členové (Members): Maria Bieliková, STU Bratislava
Miroslav Benešovský, NESS Europe Brno
Dušan Chlapek, VŠE Praha
Marie Duží, VŠB-TU Ostrava
Ján Genči, TU Košice
Jiří Gregor, GALEOS a.s.
Ivan Halaška, ČVUT Praha
Petr Hanáček, VUT v Brno
Tomáš Hruška, VUT Brno
Petr Hujňák, Per Partes Consulting Praha
Karel Ježek, ZČU Plzeň
Štefan Kovalík, ŽU Žilina
Jaroslav Král, UK Praha
Pavel Král, ZČU Plzeň
Petr Kučera, Komix s.r.o.
Aleš Limpouch, TopoL Software, s.r.o.
Karol Matiaško, ŽU Žilina
Peter Mikulecký, UHK Hradec Králové
Martin Molhanec, ČVUT Praha
Jaroslav Pokorný, UK Praha
Lubomír Popelínský, MU Brno
Karel Richta, UK Praha

Václav Řepa, VŠE Praha
Vojtěch Svátek, VŠE Praha
Petr Šaloun, VŠB-TU Ostrava
Petr Tůma, UK Praha
Michal Valenta, ČVUT Praha
Tomáš Vlk, TRIL s.r.o. Kladno
Peter Vojtáš, UK Praha
Jaroslav Zelený, IBM ČR Praha

Organizační výbor (Organization Committee)

Anna Kotěšovcová, organizační agentura CONFORG, s.r.o.
Marek Rychlý, VUT Brno
Petr Šaloun, VŠB-TU Ostrava
Jaroslav Zendulka, VUT Brno
Michal Žemlička, UK Praha

DATAKON 2011 organizují (DATAKON 2011 is organized by)

Matematicko-fyzikální fakulta, UK Praha
Česká společnost pro systémovou integraci
Slovenská informatická společnost
Přírodovědecká fakulta, Ostravská univerzita v Ostravě
Fakulta informačních technologií, VUT Brno

Partneři konference DATAKON 2011 (DATAKON 2011 Partners)

Profinit, s.r.o.

Vema, a.s.

IBM Česká republika, spol. s r.o.

KOMIX s.r.o.

Obsah

Zvané přednášky

Enterprise Architecture a řízení ICT aktiv

Petr HUJŇÁK

Per Partes Consulting, s.r.o.
Complexity Centre, Bohunická 47a, 619 00 Brno
petr.hujnak@perpartes.cz

Abstrakt. Architektury IT intenzivních systémů jsou preferovanou cestou zvládání jejich komplexity. Architekturou se rozumí základní organizace systému začleněná do jeho komponent, jejich vzájemných vztahů a vztahů k okolí systému a principy určující integritu návrhu a postupného rozvoje systému. Současná praxe architektonického navrhování vychází z různých hledisek, kterými se na systém dívají zainteresované strany a ze vzájemné integrace těchto pohledů do integrované architektury systému. Příspěvek rozebírá teoretické základy a praktické zkušenosti autora v oblasti návrhu architektur a architektonického integračního managementu.

Klíčová slova: Enterprise Architecture, aktivum, architektura, architektonický integrační management.

1 Úvod

Architektura IT intenzivních systémů a s ní neodmyslitelně svázaný architektonický management prodělává v posledním desetiletí prudký rozvoj. Hlavním hybatelem je potřeba dlouhodobě konsistentně plánovat rozvoj těchto systémů a současně je řešit konceptem výběru nejlepších aplikací a jejich integrací („best of breed“). Snaha o propojení architektury systému na podnikání organizací vedla ke konceptu Enterprise Architecture, který zapojuje do návrhu a řízení rozvoje architektury vazbu na business organizace. Vedoucí roli v oblasti Enterprise Architecture dnes sehrává architektonický rámec TOGAF [9]. Architektonický management začal ovlivňovat nejen způsob plánování a řízení rozvoje ICT, ale i s tím související věci, jako je např. způsob provádění ICT servisních služeb. Architektura systému se tak dostala do popředí zájmu v organizacích intenzivně využívajících informační technologie a systémy a zaujala integrační pozici pro všechny pod ni zařazené oblasti.

1.1 Co je to architektura a proč ji vytvářet?

Architektura systému je konceptuálním modelem systému a má za úkol vyjádřit základ (esenci) daného systému. Pod pojmem architektura se v praxi setkáváme s představami:

- architektury jako plánu nebo projektového záměru řešení (tj. produktu) projektu
- architektury jako dohodnutého jazyka pro popis systému
- architektury jako množiny rozhodnutí nebo pravidel relevantních pro systém.

Architektura v dnešním pojetí zahrnuje všechny tyto představy. Zkušenosti z praxe ukazují, že architektura navíc vždy neodmyslitelně souvisí s dosažením shody mnoha

zainteresovaných stran nad podobou daného systému ovlivněnou mnoha (partikulárními) oblastmi jejich zájmů, u kterých může docházet ke konfliktům.

Architekturou [8][9] rozumíme fundamentální organizaci systému ztělesněnou prvky systému, jejich vzájemnými vazbami vč. vazeb na okolí a principy vedoucími k návrhu a postupnému rozvoji systému.

V definici není jasně čitelný účel, proč je architektura systému vytvářena. Na rozdíl od koncepčního či hrubého návrhu systému je základním motivem pro vytvoření architektury zajistit integritu systému včetně jeho subsystémů a realizačních komponent. Do definice architektury by bylo rozumné zakomponovat pojem integrita:

Architekturou se rozumí základní organizace systému začleněná do jeho komponent, jejich vzájemných vztahů a vztahů k okolí systému a principy určující *integritu* návrhu a postupného rozvoje systému. [2]

Integritou rozumíme konzistenci (nerozpornost) a kompatibilitu (schopnost fungovat dohromady) očekávání, hodnot, zásad, principů, měřítek, metod, procesů / činností, technologií, produktů a řešení.

Hlavním motivem architektonického managementu probíhajícího nad navrženou architekturou systému je zajištění integrity systému. Architekturu systému bychom mohli formulovat pragmaticky tak, abychom zdůraznili její integrační charakter.

Architekturou dále také rozumíme množinu vytčených zásad a principů užívaných k dosažení integrity dílčích návrhů / řešení přes více vzájemně souvisejících oblastí zájmu. Architektura ovlivňuje individuální řešení vyžadováním společných zásad a principů, poukazováním na vazby k jiným systémům a požadováním zavedení těchto principů a vazeb.

Architektonickým managementem se zavádí návrhové zásady a principy za hranicemi řešení jedné oblasti či jednoho projektu a architektura má tak integritní účinek na řešení všech začleněných oblastí zájmu.

1.2 Pro jaké systémy vytvářet architekturu?

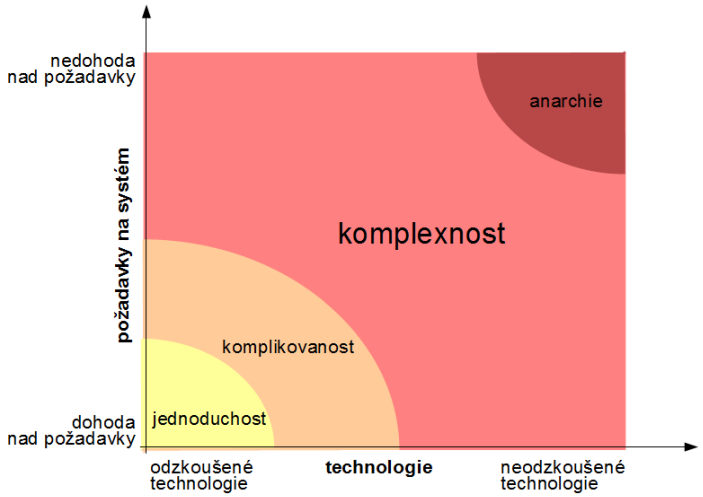
Architekturu lze sice vytvořit obecně pro jakýkoliv systém, ale úsilí spojené s jejím návrhem dává u některých systémů větší smysl než u jiných. Vyjdeme-li z předpokladu uvedeného v předchozí části příspěvku, tj. že hlavním motivem architektonického managementu je uřízení integrity systému, lépe uchopíme rozhodnutí o jejím vytvoření. Asi nejvhodnější rozhodování o tom, zda architekturu systému navrhnout a následně praktikovat architektonický management, lze v praxi provádět na základě posouzení těchto charakteristik:

- komplexnost systému
- postup realizace systému.

Komplexnost systému

Vodítkem pro rozhodnutí o míře komplexnosti ICT systému je následující schéma, které dává do souvislosti nejistoty kolem znalosti a nekonfliktnosti požadavků na systém (slabě strukturované zadání) na jedné straně a rozsahu technologické inovace na straně druhé.

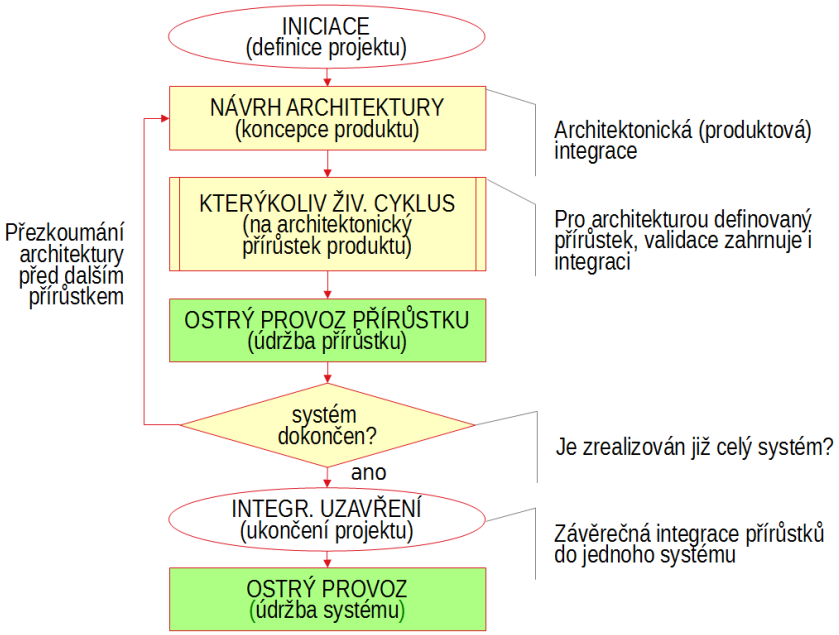
Komplexní systémy v ICT vykazují i ostatní rysy komplexity, jako je multidisciplinární předmětná oblast systému, změny očekávání a požadavků v průběhu realizace systému a slabá určitelnost projektového trojimperativu (předmět, čas a náklady na projekt).



Obr. 1. Vodítko pro určení komplexnosti systému

Postup realizace systému

Postup realizace systému je vymezen v životním cyklu projektu, kterým hodláme systém implementovat. Pokud je systém realizován přírůstkovým životním cyklem, potom je zcela zásadní uřízení integrity jednotlivých postupně realizovaných a do provozu zaváděných přírůstků (segmentů) realizace systému.



Obr. 2. Přírůstkový životní cyklus postupu realizace systému.

V přírůstkovém životním cyklu realizace systému nejde primárně o časový postup zakotvený do životního cyklu, ale o uřiditelnost vazeb mezi již v produktivním ostrém provozu využívanými a obtížně měnitelnými přírůstky systému a navazujícími přírůstky, které teprve budou detailně navrženy a tento jejich návrh může mít dopad do změn již provozovaných částí systému.

Přírůstkový životního cyklus je využíván pro implementaci systémů, které lze obtížně zavádět najednou („velkým třeskem“) právě díky jejich komplexnosti. Základním motivem je postupné uvádění přírůstků systému do provozu. Přírůstky tvoří z hlediska projektového pojetí dílčí produkty a je potřeba je vzájemně integrovat. Základem integračního managementu systému jako celku je dobrý návrh architektury systému a pod touto architekturou vytváření detailních návrhů jednotlivých přírůstků. U přírůstkového životního cyklu je třeba zdůraznit zpětnou kontrolní vazbu směřující od návrhu přírůstků systému k přezkoumání celkové architektury systému. Nutnou podmínkou pro postupnou realizaci přírůstků je zvládnutá architektonická integrace částí (tj. implementovaných přírůstků) do vzájemně provázaného celku (tj. systému), což ukazuje na stěžejní roli architekta při přírůstkovém postupu realizace systému.

2 Architektonická hlediska, pohledy a principy (horizontální integrace)

2.1 Architektonické pohledy a hlediska

Centrálním pojmem architektonického horizontálního integračního managementu je architektonický pohled, architektonické hledisko a model.

Architektonický pohled

Architektonickým pohledem (architectural view) se rozumí [8] reprezentace systému z perspektivy identifikované množiny s architekturou spojených zájmů. Zájmy (system concerns) představují oblasti systému, které jsou důležité pro některou ze zainteresovaných stran (stakeholders - zákazníci, uživatelé, operátoři, návrháři, vývojáři, testeři, integrátoři, hodnotitelé, projektoví manažeři a také architekti). Zainteresované strany představují zájmy pozorovatelů systému dané zpravidla jejich odpovědnostmi. Typické zájmy zahrnují funkcionalitu, integrovanost, modifikovatelnost, výkonnost, bezpečnost a spolehlivost nebo třeba také náklady, termíny či kvalitu systému.

Architektonický model (architectural model) přispívá k obsahu architektonického pohledu. Někdy namísto pohledu mluvíme o architektuře upřesněné přívlastkem určujícím o jaký pohled jde – např. informační architektura, technologická architektura.

Architektonický popis (architectural description) se skládá z jednoho či zpravidla z více architektonických pohledů. Každý z pohledů je vyjádřením architektury systému z určitého architektonického hlediska reprezentujícího zájmy některé ze zainteresovaných stran.

Architektonické hledisko

Architektonickým hlediskem (architectural viewpoint) se rozumí [8] konvence pro vytvoření, interpretaci a užití architektonického pohledu a k němu přispívajících architektonických modelů. Architektonické hledisko stanovuje konvenci, podle které je architektonický pohled vytvořen, znázorněn, interpretován a analyzován. Konvence hledisek zahrnují jazyk a notaci užitou pro architektonický pohled a zahrnují také

používané druhy modelů, modelovací metody a analytické postupy aplikované k vytvoření pohledu. Přitom mohou být užita obvyklá zaužívaná (předdefinovaná) hlediska nebo může být vytvořeno hledisko specifické pro potřebný architektonický pohled.

Pohled vzniká aplikací architektonického hlediska na předmětný systém a je znázorněn modelem či modely systému vytvořených z perspektivy uplatněných zájmů zainteresovaných stran. Pohled se skládá obecně z více architektonických modelů. Pro popis architektury systému se používá více pohledů vzniklých zpravidla z mnoha uplatněných hledisek. Každý architektonický pohled musí mít k sobě hledisko, jehož aplikací vznikl a které současně udává množinu konvencí pro interpretaci obsahu pohledu.

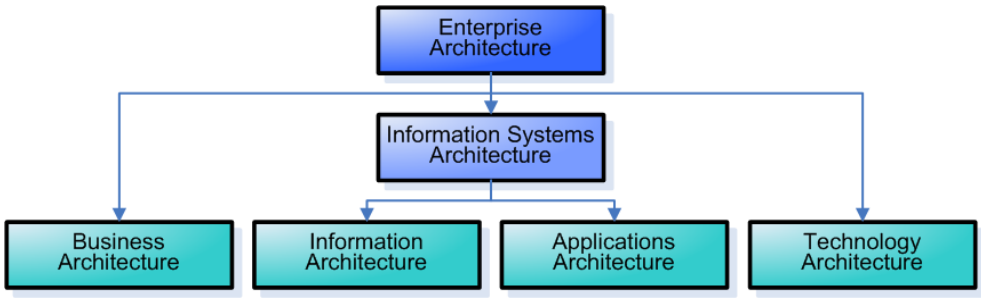
Architektonický model

Architektonický model (architectural model) je model přispívající k obsahu architektonického pohledu. Má přispět ke znázornění křížových aspektů v jednom nebo ve více architektonických pohledech a ke sdílení společných detailů napříč mnoha pohledy [8].

Pragmatická definice modelu vychází z toho, že model je jakýkoliv koncept nebo fyzická věc vztažená k systému, pokud slouží k zodpovězení otázek o předmětném systému [8].

2.2 Typické architektonické pohledy na systém

Úlohou architekta systému je volba pohledů, kterými pokryje multidisciplinaritu předmětné oblasti zahrnuté (tj. řešené) v rámci daného systému. Multidisciplinarita obvykle dobře koresponduje se zájmy zainteresovaných stran. Pro IT intenzivní systémy se zpravidla jedná o oblasti, které jsou užívány v TOGAFu [9].



Obr. 3. Čtyři základní domény TOGAFu [9].

TOGAF pracuje se 4 základními dílčími architekturami vzniklými z hlediska úhlu pohledu na čtyři základní domény architektonického zájmu :

– Business architektura, která adresuje potřeby business managementu, uživatelů a plánovačů. Zpravidla obsahuje klíčové business procesy, které budou systémem podporovány.

– Datová architektura, která popisuje potřeby databázových návrhářů, databázových administrátorů a systémových inženýrů.

– Aplikační architektura, která popisuje potřeby systémových a softwarových inženýrů.

– Technologická architektura, která popisuje potřeby provozních a servisních operátorů a technických administrátorů.

Pro mnoho architektů jsou užitečnými hledisky samotné domény TOGAFu. Architektonické domény TOGAFu jsou tedy dnes obecně uznávanými hledisky, která jsou zpravidla ještě doplněna specifickými hledisky, typicky např. hlediskem bezpečnosti. Hlediska jsou reprezentací architektury systému vyjádřené ve smysluplné podobě pro pozorovatele (zainteresovanou stranu) tak, aby mohl verifikovat, že systém bude odpovídat jeho požadavkům.

Výběr toho, které architektonické pohledy (views) vytvořit, patří mezi klíčová rozhodnutí architekta. Ten odpovídá za:

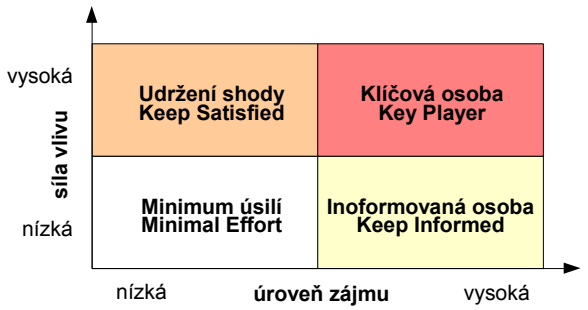
- úplnost pohledů z hlediska pokrytí všech relevantních záležitostí
- vhodnost pohledů směrem k dosažení jejich účelu
- integritu architektury vzniklé spojením jednotlivých pohledů
- rozkrytí konfliktních požadavků a ukázání kompromisních řešení.

Architekt vybírá více pohledů smysluplných pro osoby s rozhodovací pravomocí (zainteresované strany), které musí verifikovat, že architektura odpovídá jejich zájmům. Pohledům odpovídají architektonické modely, které společně poskytují celistvý popis architektury systému.

2.3 Rizika kolize hledisek a zájmů zainteresovaných stran

ISACA ve své metodice COBIT [1] reaguje na rizika, která vyvstanou, pokud není hledisko správně uplatněno pro zainteresovanou stranu. Každé jednotlivé hledisko představuje abstraktní model znázorňující, jak konkrétní zainteresovaná strana (stakeholder) vidí systém resp. jeho architekturu. Pohledem je to, co je vidět jako výsledek uplatněného hlediska.

Příkladem uplatnění technologického hlediska, které může způsobit nedorozumění zainteresovaných stran, je vytvořený pohled popsaný detailním modelem topologie sítě a osazením uzlů sítě s technicky detailně popsanými HW prvky vč. jejich IP adres. Takový detailní model je jistě užitečný pro technologického architekta, nicméně pro business management je výsledkem pohledu na technologie zcela něco jiného, tj. zpravidla model představující hrubé schéma datových center zasazených do topologie sítě namapované na organizační uspořádání pracovišť. Obě zainteresované strany si pak nad stejným hlediskem s jedním modelem v pohledu nevystačí. Proto je architektonický pohled potřeba znázornit zpravidla několika modely podle zainteresované strany, která jej bude verifikovat. Analogií této situace je model aplikace provedený z hlediska pohledu uživatele (popis užívání aplikace), nebo z hlediska pohledu programátora (popis výstavby aplikace).



Obr. 4. Klasifikace silové pozice zainteresovaných stran. Hlediska a použité modely v pohledech jsou ovlivněny jejich silou vlivu na architekturu systému [9].

TOGAF [9] doporučuje klasifikovat zainteresované strany podle pozice a síly, kterou mohou v dané organizaci nařídit, změnit či blokovat směrování architektury a podle toho, zda je v jejich zájmu se do prací na architektuře aktivně zapojit či nikoliv.

Zkušenosti (autora tohoto článku) vedou k využívání následujících nejčastějších architektonických hledisek.

Primární odpovědnost (zaint. strana)	Oblast zájmu	Hledisko (dílčí architektura)	Relevantní standardy a normy	Relevantní modely
Vrcholové vedení organizace popř. vlastníci organizace	Dlouhodobý plán strategických hodnot organizace	Strategie organizace	Balanced Scorecards	BCS schéma
Vrcholové vedení, vedoucí útvarů a vlastníci procesů	Business procesy a systém řízení organizace	Business architektura	VRM TOGAF	Procesní diagramy
Vedení ICT útvaru a vrcholové vedení organizace	Plán rozvoje informačních systémů ve vazbě na strategii organizace	Informační strategie	Balanced Scorecards	BSC schéma
Vedení ICT útvaru, vedoucí oddělení	Informatické procesy a systém řízení ICT útvaru	Strategie ICT útvaru	COBIT Balanced Scorecards	Procesní diagramy
Informační správce, správce databází, uživatel	Data / informace / znalosti	Datová architektura	TOGAF COBIT	Datové modely
Aplikační správce, uživatel	Aplikace	Aplikační architektura	TOGAF	Aplikační modely
Správce infrastruktury	Infrastruktura (HW a sítě)	Infrastrukturní architektura	TOGAF	Infrastrukturní modely
Bezpečnostní správce	Bezpečnost	Bezpečnostní architektura	Řada ISO 27000 TOGAF	Modely informačních aktiv a jejich rizik
Provozní správce	Servis	Servisní architektura	ITIL COBIT ISO 20000	Modely servisních služeb a jejich SLA parametrů

Tab. 1. Typické zainteresované strany, jejich zájmy, hlediska a modely.

Aplikací v tabulce uvedených architektonických hledisek získáme pohledy na architekturu, u kterých je nezbytné zajistit vzájemnou integritu. Do těchto pohledů jsou vloženy záměry architekta na strukturu systému, jeho komponenty a charakteristiky. Architektonický integrační management je popsán dále v textu.

2.4 Architektonické principy

Architektura je určena k řízení komplexity systému. Architektura má vliv na dílčí subsystémy požadováním respektování vzájemných vazeb a vazeb na okolní systémy a také tím, že požaduje, aby dílčí subsystémy byly vytvářeny na základě principů, které jsou v architektuře stanoveny.

Architektonickým principem [9] rozumíme kvantitativní vyjádření záměru, kterému má být vyhověno v architektuře. Architektonické principy vyžaduje architektonický rámec TOGAF [9] i COBIT [1] od ISACA.

Architektura v tomto pojetí představuje množinu principů užitých k harmonizaci návrhových voleb přes multidisciplinární oblasti, které je třeba řešit v souvislosti s realizací systému. Architektura tak vytyčuje požadavky, které jsou za hranice dílčích oblastí zájmu.

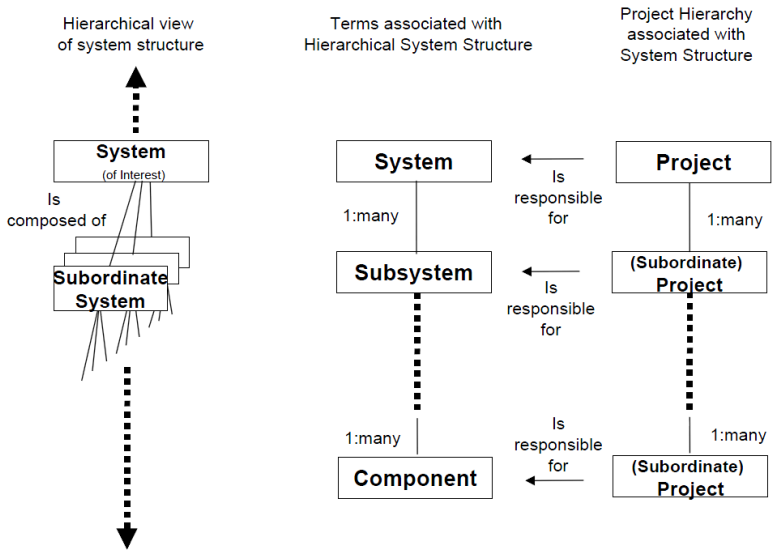
Shoda pohledů

Shody pohledů (view correspondences) se dosahuje pomocí pravidel shody, která zajišťují integritu mezi jednotlivými architektonickými pohledy na systém. V praxi se vyskytují nejčastěji ve formě binární relace mezi elementy různých pohledů. Příkladem pravidla shody je pravidlo, které přiděluje každé softwarové aplikaci z aplikační architektury (tj. pohledu vzniklému uplatněním softwarového hlediska/viewpoint) jednu nebo více hardwarových platforem z hardwarové architektury (tj. pohledu vzniklého uplatněním hardwarového hlediska).

3 Dekompozice systému a realizace subsystémů projekty (vertikální integrace)

Norma ISO 15288 [6] předpokládá, že zájmový systém se skládá z několika podřízených systémů a vychází z toho, že integrací systémů nižší hierarchické úrovně vzniká nadřízený systém. Každý systém lze rozložit na subsystémy (jeden až mnoho) a úroveň dekompozice se zastavuje na komponentách představujících nejnižší úroveň hierarchie systému. Předpokládá se, že realizací těchto komponent se přispěje k realizaci nadřízeného subsystému a integrací subsystémů zrealizujeme pak celý systém.

Analogicky tomu odpovídá organizace práce, která realizuje systém na úrovni projektu a komponenty na příslušné nižší úrovni hierarchie subprojektu, vždy s odpovědností za realizaci příslušného subsystému či na nejnižší úrovni dekompozice za realizaci komponent systému.



Obr. 5. Analogie systémové a projektové hierarchie potřebné pro uřízení integrity mezi subsystémy a subprojekty pro jejich realizaci podle ISO 15288 [6].

Zpravidla existuje silná korelace mezi subsystémy a odpovědnostmi zainteresovaných stran. Útvar v organizaci, který má odpovědnost za projekt zaměřený na realizaci určitého subsystému, nemívá odpovědnost za nadřazený systém, ale mívá odpovědnost za všechny podřízené subsystémy až komponenty. Přitom však musí příslušný útvar vnímat vliv nadřízeného systému, i když za něj nemá odpovědnost. Integrační úsilí by mělo být vyvíjeno nejen směrem k podřízeným subsystémům (tj. dolů), ale i nahoru směrem k nadřízenému systému, zpravidla nazývaného okolím systému.

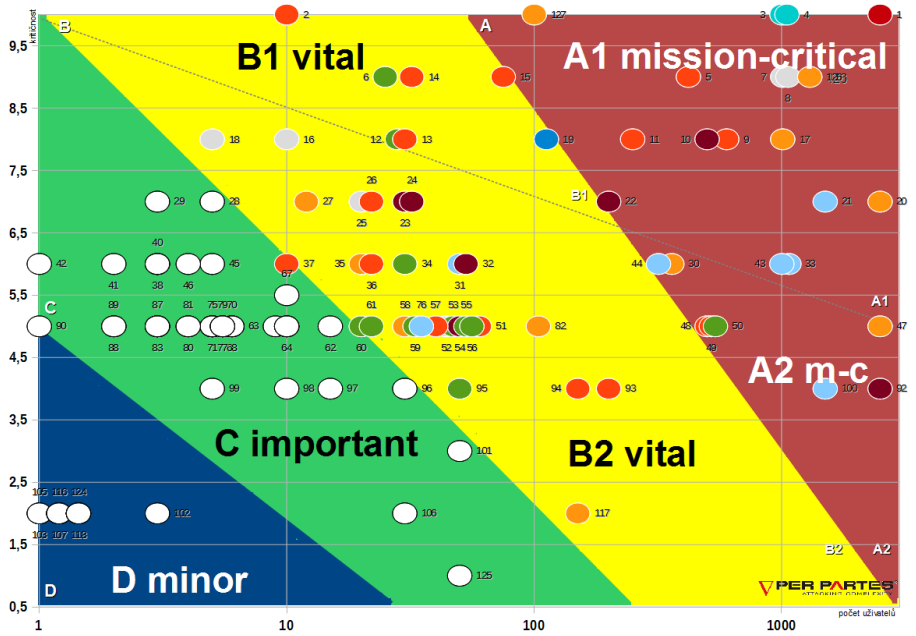
4 Architektonická rozhodování a aktiva (integrace aktiv)

Ve skutečnosti je motiv dekompozice na subsystémy v rámci architektonického managementu ovlivněn potřebami vyjasnit a rozhodnout o variantách subsystémů či komponent nižší úrovně, které mají architektonický význam.

4.1 Architektonické rozhodnutí

Architektonické rozhodnutí (architectural decision) [8] je provedená volba adresující jeden nebo více architektonických zájmů a ovlivňující (přímo či nepřímo) architekturu systému. Architektonická rozhodnutí jsou prováděna na základě znalosti předmětného problémového prostoru daného systému a nejsou omezena na řešitelský prostor ve smyslu návrhového rozhodnutí.

Příkladem architektonického rozhodnutí, které významně ovlivňuje přístup architekta a je zpravidla jedním ze základních rozhodnutí, které je potřeba v rámci architektonického managementu provádět, je rozhodnutí o prioritizaci aplikací. Prioritní aplikace musí být v centru pozornosti architekta a tvoří jádro aplikační architektury.



Obr. 6. Příklad zařazení cca 130 aplikací (zde označených čísly) v dané organizaci do zón podle jejich významu (prioritizace aplikací). Na vodorovné ose je počet uživatelů, na svislé pak kritičnost podporovaných procesů danou aplikací.

Jak je na obrázku znázorněno, v praxi se osvědčuje prioritizovat aplikace jejich zařazením do zón významnosti. Prioritizace aplikací představuje základní architektonické rozhodnutí potřebné pro uřízení rizika hodnoty a rizika rozsahu. Význam aplikace se v praxi osvědčuje vyjádřit ve vztahu kritičností jím podporovaných „business“ procesů a počtu (se subkritériem charakteru) uživatelů aplikace.

– zóna A – aplikace kritické (mission-critical) z pohledu zajištění jejich provozu pro organizaci, využívá je mnoho uživatelů a mají vysokou důležitost funkcionality

– zóna B – aplikace životně důležité (vital)

– zóna C – aplikace důležité (important)

– zóna D – aplikace nevýznamné, podpůrné a doplňkové (minor).

Z architektonického rozhodnutí o významu aplikací pro danou organizaci vychází zpravidla řada architektonických principů. Nejdůležitějším z nich bývá princip diferencovaného přístupu k aplikacím umožňující směřovat zesílené úsilí směrem k prioritním aplikacím.

Název principu	ID	Deklarace principu (co)	Zdůvodnění principu (proč-přínosy)	Dopad principu (požadavky na zdroje)	Reference
Diferencovaný přístup k aplikacím podle jejich významu	AD – PR1	Aplikace mají pro organizace různý význam (důležitost, prioritu) a proto je potřeba k nim přistupovat diferencovaně.	Rozlišováním aplikací podle významu dochází k přesnějšímu a hospodárnějšímu nastavení parametrů a prioritizaci informatických služeb.	Zařazování aplikací do skupin podle významu.	Per Partes metodika

Tab. 2. Příklad formulace principu *diferencovaného přístupu k aplikacím* na základě znalosti významu aplikací pro danou organizaci.

Další důležité principy vytvořené na základě rozhodnutí o významu aplikaci bývají směrovány do oblasti SLA parametrů servisních služeb vystavených kolem aplikací a do oblasti navázané prioritizace infrastruktury. Vychází se z vazeb mezi architektonickými aktivy.

4.2 Architektonická ICT aktiva

Pojem aktivum (asset) není ještě ustálen a používá se v různých významech. Tento pojem je nejčastěji používán pro obor informačních technologií v normách na řízení bezpečnosti [7].

Aktivem nazývá norma ISO 27001 [7] cokoliv, co má pro organizaci nějakou hodnotu. Z kontextu vyplývá, že aktiva mohou mít hmotnou i nehmotnou povahu (schopnost vykonávat službu či znalost) a je proto potřeba je řídit. Podle COBITu [1] jsou aktiva (asset) používána v podobném významu, tj. mohou být hmotná a nehmotná a jsou předmětem „IT governance“ (doslova v překladu a významu „panování“), právě proto, že mají pro organizaci hodnotu.

Architektonickým aktivem nazvěme ty komponenty architektury systému, které jsou zobrazeny v architektonických pohledech a současně mají význam (jsou relevantní) pro architektonický management.

TOGAF [9] používá pojem architektonické aktivum v rozšířeném významu a zařazuje mezi architektonická aktiva architektonický popis, referenční modely a různé použité vzory. Architektonická aktiva jsou ukládána v knihovně a je řízeno jejich využití. Tento význam pro účely dalšího textu nebude použit a pod architektonickým aktivem budeme rozumět komponentu systému včetně jejích charakteristik, kterou je potřeba z hlediska architektury systému řídit z pozice architekta. Koneckonců ISO norma pro architektonický popis [8] používá ve významu architektonického popisu, referenčních modelů a různých použitých vzorů pojem „AD element“ (Architecture Description) a řadí mezi ně pohledy, hlediska, modely, rozhodnutí, zdůvodnění, zainteresované strany a jejich zájmy.

Architektonická aktiva v pojetí tohoto článku jsou součástí architektonických pohledů a objevují se v modelech jako komponenty tvořící tyto pohledy. Ne všechny komponenty pohledů však musí mít architektonický význam a proto nemá cenu komponenty, které nepřispívající a neovlivňují architekturu systému, dále dekomponovat a zkoumat. Na listové úrovni rozpadu zájmového subsystému se dostáváme ke komponentám, z nichž jen některé jsou architektonicky významné. Proto architektura může obsahovat nevyrovnanou dekompozici z hlediska granularity rozpadu. Tam, kde by mohlo dojít k nejasnostem či variantám ovlivňujícím celkovou architekturu systému, je potřeba zvětšit podrobnost dekompozice a na detailní úrovni popsat a řídit právě ty komponenty, které mají architektonický význam.

Řízení rizik

Volbu architektonických aktiv je potřeba zacílit do rizikových oblastí systému. Zvýšená hloubka dekompozice a tím i hustota aktiv pro dílčí subsystém souvisí s riziky spojenými s realizací systému, která architekt identifikoval a proti nimž vyvíjí protiopatření.

Navržení architektury systému můžeme chápat jako opatření směřující k reakci na řadu rizik, z nichž nejobvyklejší jsou čtyři níže uvedená.

Riziko hodnot

– Hrozba: Architektonická koncepce není napojena na strategické cíle organizace a nepřináší hodnoty směrem k podnikání („business“).

– Zranitelnost: Vysoká u nepropojení business strategie - ICT strategie – architektura systému. Př. ICT strategie nezahrnuje cíle k odůvodnění nastavení úrovně bezpečnosti informací.

– Architektura: Vynucuje si otázky k rozhodnutí od strategického řízení.

Riziko návrhu

– Hrozba: Nezahrnutí správných požadavků do dílčích řešení, nedosažení očekávání zainteresovaných stran (stakeholders).

– Zranitelnost: Vysoká u konfliktních požadavků. Př. Postavení řešení na jiné technologické platformě.

– Architektura: Dodává požadavky na řešení za hranicemi řešení jedné oblasti či projektu.

Riziko soudržnosti

– Hrozba: Řešení není integrovatelné s vazbovými systémy.

– Zranitelnost: Vysoká u komplexních systémů. Př. Vzájemně se vylučující IT kontroly na formát datové věty mezi vazbovými systémy.

– Architektura: Dodává zásady a principy pro udržení integrovanosti částí v celek.

Rizika rozsahu

– Hrozba: Řešení příliš rozsáhlé (více než odpovídá požadavkům) nebo naopak sporé (méně něž požadováno).

– Zranitelnost: Vysoká při slabě strukturovaných zadáních. Př. Překryv funkcionality implementovaných aplikací bez jasné dělby a vzájemného propojení.

– Architektura: Dodává kontext nezbytný pro stanovení hranic řešené oblasti.

Význam rizikové analýzy pro architekturu spočívá v tom, že architektonická aktiva, zásady a principy je potřeba směrovat prioritně do rizikových oblastí. Tam je jejich největší hodnota při odstraňování nejistot.

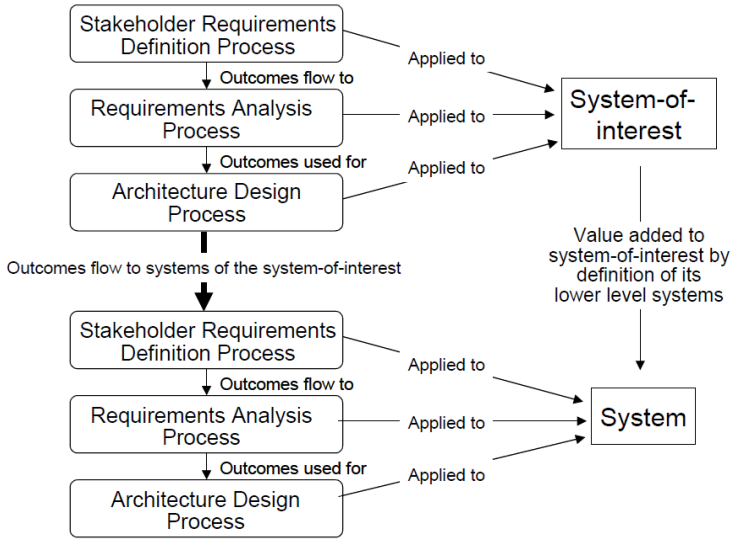
Klíčovým problémem architekta je tedy volba architektonických aktiv. Vyjdeme-li ze smyslu architektonického managementu, kterým je uřízení integrity komplexního systému zaváděného po přírůstcích (segmentech), dostávají se do popředí zájmu architektů požadavky na systém a jejich úplnost a bezrozpornost.

4.3 Řízení požadavků na systém

Rekurze a iterace požadavků a následného návrhu

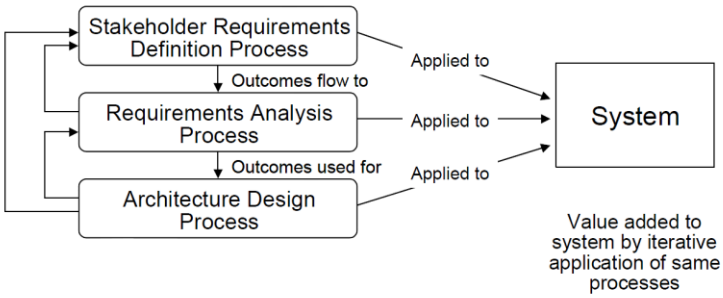
Požadavky na systém umožňují definovat jeho odpovídající architekturu. Norma [6] a její příručka [5] ukazují na rekurzi a iteraci architektonického navrhování ve vazbě na řízení požadavků.

Rekurzí architektonického navrhování se rozumí opakovaná aplikace procesů (definice požadavků zainteresovaných stran, analýza požadavků a návrh architektury) v systému shora dolů na dekomponované subsystémy až do úrovně, která je svojí hloubkou (detailností) dostatečná pro architektonický návrh.



Obr. 7. Rekurze procesů pro definici a analýzu požadavků a následný architektonický návrh v systému dekompozice shora dolů podle příručky k ISO 15288 [5].

Požadavkové a návrhové procesy se přitom neprovádějí jedním průchodem, ale dochází v nich k iteracím, kterými se postupně vytváří výsledný architektonické návrh. Podle normy ISO 14288 je tento postup žádoucí a podle zkušeností je architektonický návrh prováděn vždy v řadě přibližovacích iterací. TOGAF [9] používá iterační navrhování jako základ své metodiky ADM pro architektonický management.



Obr. 8. Iterace procesů pro definici a analýzu požadavků a následný architektonický návrh na stejný systém (subsystém, komponentu) podle příručky k ISO 15288 [5].

Architektonické navrhování představuje cyklický proces, při němž dochází v rámci navrhování systému k rekurzi návrhových procesů na subsystémy až do úrovně, která je určitá ve smyslu strukturované zadatelnosti dílčího subsystému a jeho komponent do realizačního projektu. Obdobně na stejné úrovni dekompozice dochází k iteračnímu opakování návrhových procesů až do úrovně, kdy návrh předmětného systému, subsystému či komponent odpovídá kladeným požadavkům.

„Nevyřčené" požadavky

Je zřejmé, že dobrý architektonický návrh přímo souvisí s úplností a bezrozporností požadavků zainteresovaných stran na předmětný systém. Kvalita požadavkové základny je pro architekturu natolik důležitá, že jí musí architekt věnovat zásadní pozornost. Návod na zjišťování a práci s požadavky dávají normy ISO pro řízení kvality [3][4].

Požadavkem je [3] potřeba nebo očekávání, které jsou stanoveny, obecně se předpokládají nebo jsou závazné. Obecně předpokládaný znamená, že se jedná o zvyklost nebo běžnou praxi organizace, jejich zákazníků a jiných zainteresovaných stran a že se uvažovaná potřeba nebo očekávání předpokládají. Specifikovaným požadavkem je požadavek, který je uveden např. v dokumentu. Požadavky mohou být vytvářeny různými zainteresovanými stranami.

Podle norem pro řízení kvality [4] jsou organizace závislé na zákaznících a zainteresovaných stranách a proto mají rozumět jejich současným a budoucím potřebám, mají plnit jejich požadavky a snažit se předvídat jejich očekávání.

Organizace musí určit [4, 7.2.1]:

– požadavky specifikované zákazníkem, včetně požadavků na činnosti při dodání a po dodání,

– *požadavky, které zákazník neuvedl, ale které jsou nezbytné pro specifikované nebo zamýšlené použití, je-li známo,*

– *zákonné požadavky a požadavky předpisů týkajících se produktu a*

– *jakékoli doplňující požadavky určené organizací.*

Základním předpokladem dosažení kvality architektonického návrhu je identifikace a zapracování tzv. „nevyřčených" požadavků na systém (jde o požadavky výše popsané kurzívou v posledních třech odrážkách). Zdrojem nevyřčených požadavků bývá pro architekta znalost předmětné oblasti a zkušenosti z realizace obdobných systémů. Schopnost architekta převést nevyřčené požadavky do specifikovaných

(zdokumentovaných) požadavků silně determinuje kvalitu architektonického návrhu a tím i úspěšnost realizace celého systému.

Příklady „nevyřčených“ požadavků:

– Systém musí být integrován do prostředí základních registrů veřejné správy (až budou dokončeny).

– Systém musí být odolný (samoopravný) proti výpadkům.

– Reakční odezva systému na akci uživatele nesmí být delší než 2 sekundy.

Obecně se dá ze zkušenosti říci, že významné požadavky ovlivňující architekturu systému se zpravidla objevují na rozhraní:

– Požadavky na vazby na okolí systému a směrem k nadřazené hierarchické úrovni.

– Požadavky na vazby mezi subsystémy.

– Požadavky na vazby mezi přírůstky (segmenty) postupně realizovaného systému.

Konfliktní požadavky

Zainteresované strany mají své zájmy a ty mohou vést k předkládání rozporných požadavků. Architekt pak stojí před dilema, kterému požadavku vyhovět. Schopnost vyjednat podobu řešení konfliktních požadavků je v praxi velmi častým úkolem, který je třeba v rámci architektonického navrhování dohodnout.

Příklady konfliktních požadavků:

– Požadavek zákona č.101, o ochraně osobních údajů, znemožňuje nadměrné propojení databází a sledování osob klientů požadované obchodním oddělením.

– Požadavek na informační bezpečnost (bezpečnostní politika organizace) snižuje uživatelskou přívětivost (přes heslový management) a omezuje přístup uživatelů k datům, která přímo nepotřebují pro svoji práci, ale rádi by k nim měli občasný přístup.

Znalost hodnoty systému

Podle metodiky VALIT od ISACA [10] je potřeba řídit nejen požadavky, ale zejména hodnotu (value), kterou organizace získá díky užívání zrealizovaného systému či jeho přírůstků (segmentů systému). Pro vymezení účelu (tj. smyslu) realizace systému pro organizaci je třeba podle VALIT zpracovat dokument nazvaný „Business Case“ definující přínosy. Přínosy mohou být finančně i nefinančně vyjádřeny. V „Business Case“ nejde tedy primárně o požadavky na systém, ale o stanovení toho, jaký smysl má jeho realizace. Je zřejmé, že architekt by měl rozumět tomu, za jakým účelem jím navržený systém bude využíván. Znalost efektů z využívání navrhovaného systému je zásadní pro stanovení rozsahu systému (scope) a jeho hranic vůči okolí, pro stanovení externích vazeb a také pro nalezení „nevyřčených“ požadavků.

Příklady přínosů (efektů z využívání), které mají dopad do architektury systému:

– Přínos *zrychlení objednávkového procesu* v novém systému – vyvolá požadavek na optimalizaci výkonnosti procesu – má přímý dopad do business architektury.

– Přínos *snížení servisních nákladů* – vyvolá požadavek na umožnění vzdáleného servisu systému – má přímý dopad do technologické architektury.

– Přínos *snížení chybovosti zadávaných dat* – vyvolá požadavek na automatickou kontrolu vstupních dat před jejich přijetím – má přímý dopad do datové a aplikační architektury.

Verifikace a validace požadavků

Verifikace je proces kontroly kvality zjišťující, že požadavky a specifikace na předmětný systém stanovené při zahájení jeho vývoje jsou splněny. Dochází k ověření správného vytvoření systému oproti architektuře (záměru) a v ní uvedeným charakteristikám a metrikám pro jejich ověření. Pokud záměr systému specifikuje produkt nevyužitelný k zamýšlenému užití, verifikací se to nepozná.

Validace je proces zajištění kvality prokazující, že předmětný systém je využitelný k zamýšlenému užití a plní vytčený účel. Dochází k ověření jeho užitné hodnoty ve smyslu vytvoření správné věci. Validací tak můžeme ověřit i ty požadavky na systém, které jsme do záměru formulovaném v architektuře systému nezahrnuli, ale přesto jsou pro užívání systému potřebné. Validace ověřuje, že jsou splněný i „nevyřčené“ požadavky. Validační ověření je typické pro integrační testování systému v provozních podmínkách nebo pro ověřovací provoz systému v produktivním prostředí.

Architektura je při svém návrhu ověřitelná verifikací. Postupně realizované přírůstky (segmenty) systému procházejí validačním ověřováním a představují tak cennou zpětnou vazbu pro správnost navržené architektury systému. Proto v přírůstkovém životním cyklu musí vždy dojít při realizaci přírůstků (segmentů) systému ke zpětnovazebnému přezkoumání architektury celého systému.

Provázanost architektonických aktiv

Aktiva neexistují osamoceně a vstupují do vazeb na ostatní subsystémy či komponenty popř. na okolí systému.

Diferencovaný přístup k aplikacím podle jejich významu pro organizaci umožňuje následně prioritizovat i technicko-technologická aktiva. Propojení aplikačních a technicko-technologických aktiv se snižuje riziko hodnot a riziko rozsahu. Je obecně nevýhodné, aby různě významné aplikační systémy byly umísťovány do stejně zabezpečeného např. vysoce dostupného technicko-technologického prostředí (fyzická infrastruktura).

Název principu	ID	Deklarace principu (co)	Zdůvodnění principu (proč-přínosy)	Dopad principu (požadavky na zdroje)	Reference
Vybalancování infrastruktury k aplikacím podle priorit	TT – PR2	Jsou identifikovány vazby mezi prioritizovanými aplikacemi a infrastrukturními prvky potřebnými pro jejich běh, Technicko-technologická Infrastruktura je tak prioritizována podle podmíněnosti pro provoz aplikací. Nasazení prioritních aplikací je soustředěno na vysoce dostupnou infrastrukturu.	Diferencovaný přístup k infrastruktuře podle její podmíněnosti k aplikacím umožňuje soustředit prostředky na zabezpečení klíčové prioritní infrastruktury. Na druhé straně nejsou neprioritní aplikace nasazovány na vysoce zabezpečenou infrastrukturu společně s prioritními aplikacemi.	Infrastrukturní prvky musí být prioritizovány z hlediska jejich významu pro běh aplikací.	Per Partes

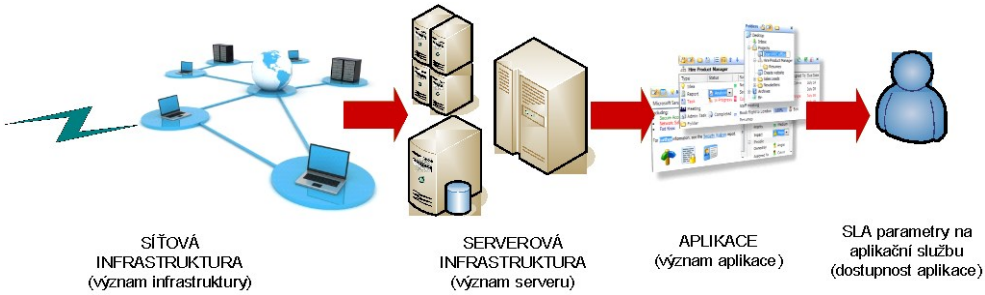
Tab. 3. Příklad formulace architektonického principu využívajícího architektonické provázanosti aplikačních a technických aktiv.

Výše popsaný příklad architektonického principu si vynucuje, aby prioritní infrastruktura byla přímo navázána na prioritní aplikace, které jsou zase navázány na podporu prioritních „business“ procesů. Úvahu lze propojit ještě dále směrem do servisu, který by měl také reagovat SLA parametry na významnost aplikací.

Integrační pohled propojuje aktiva do vzájemně provázaných vyšších celků (tzv. „superaktiv“), které jsou zdrojem pro poskytování aplikačních služeb. Aplikační služby

jsou z pohledu uživatelů chápány jako jeden celek, tj. uživatelé nerozlišují vnitřní strukturu a podmíněnosti aktiv, která jsou nutnou podmínkou pro fungování dané aplikace.

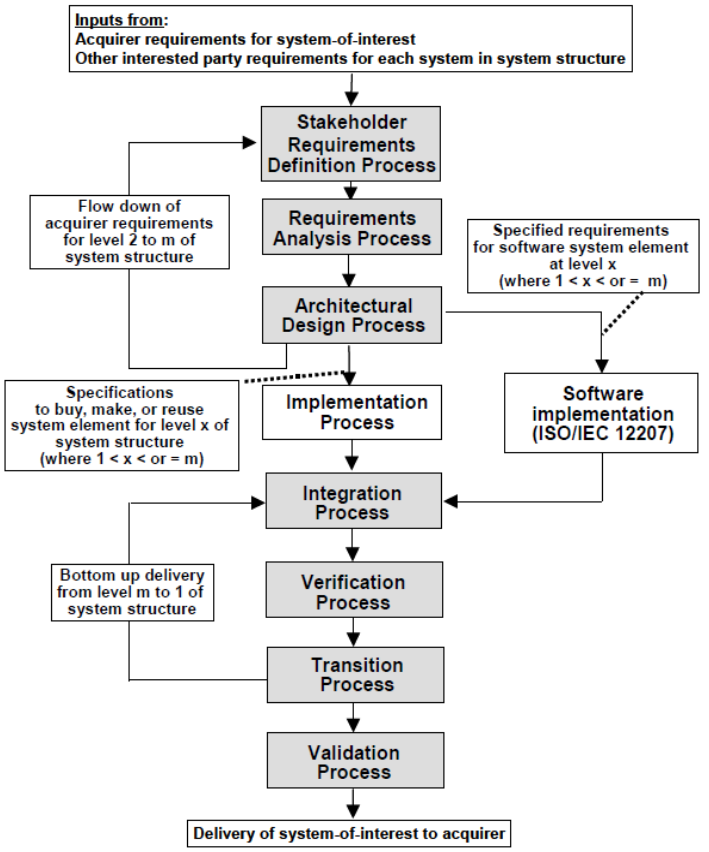
„Superaktiva“ zahrnující technicko-technologická aktiva a aplikačně-datová aktiva (AD) vytvářející společně pro uživatele aplikační službu.



Obr. 9. Cesta od infrastruktury je protažena přes aplikace a data až na aplikační služby a jejich SLA parametry pomocí architektonických superaktiv. SLA parametry služeb nejsou přáním ICT útvaru, ale „business“ potřebou konkrétní skupiny interních a/nebo externích uživatelů.

5 Architektonický integrační management

Návrhové procesy jsou iteračně a rekurzivně prováděny shora-dolů dokud se nedosáhne na komponenty systému zadatelné do implementačního projektu. Integrační management naopak postupuje zdola-nahoru, tj. skládá komponenty do vzájemně provázaného funkčního celku.



Obr. 10. Integrační management navázaný na architektonické navrhování podle příručky k ISO 15288 [5].

Integrace ve skutečnosti probíhá ve více rovinách, než jak ukazuje výše uvedené schéma normy [5]. Integrace probíhá také ve směru integrity pohledů tak, aby došlo ke sjednocení požadavků a zájmů zainteresovaných stran. Další úroveň dekompozice a následné integrace je dána architektonickými zájmy vyjádřenými pomocí aktiv. Ta jsou integrována do vzájemně architektonicky propojených skupin – tzv. superaktiv. Integrační úsilí je také vytvářeno směrem k přechodovým stavům systému – přírůstkům realizace.

Architektonický vertikální integrační management (integrita částí)

Stanovme si pod pojmem *architektonický vertikální integrační management* integrační úsilí, které je vynakládáno ke (vhodné) dekompozici zájmového systému na subsystémy až do úrovně detailních komponent a současně k integraci komponent opačným směrem na subsystémy až do vzájemně provázaného systému jako celku. Jde o integraci, která je detailně popsána v normě ISO 15228 [6] a její příručce [5].

Architektonický horizontální integrační management (integrita pohledů)

Stanovme si pod pojmem *architektonický horizontální integrační management* integrační úsilí, které je vynakládáno k vytváření multidisciplinárního pohledu na systém (či subsystém nebo komponenty) z hledisek zájmových skupin (zainteresovaných stran) a současně k vzájemné shodě pohledů (view correspondences) vzniklých uplatněním těchto hledisek. Tento přístup je dobře popsán v TOGAFu [9].

Architektonický integrační management aktiv (integrita aktiv)

Stanovme si pod pojmem *architektonický integrační management aktiv* integrační úsilí, které je vynakládáno k identifikaci klíčových architektonických aktiv systému (subsystémů či realizačních komponent) a jejich vzájemných vazeb a současně k propojení aktiv do vzájemně provázaných vyšších celků (tzv. „superaktiv“). Tento přístup vychází z úvah popsaných v článku autora [2].

Přechodový integrační management (integrita přírůstků)

Stanovme si pod pojmem *přechodový integrační management* úsilí, které je vynakládáno na vymezení implementačně vhodných přírůstků realizace systému (tj. přechodových stavů systému) a současně k integraci přírůstků do vzájemně provázaného systému jako celku.

V přechodovém integračním managementu jde o dekompozici systému vedenou motivem proveditelnosti a využitelnosti vhodné skupiny komponent představující přírůstky realizace systému (implementační strategie) a současně jde o úsilí řízení realizace implementačních přírůstků tak, aby konsistentně přispívaly k realizaci celého systému.

Přechodový integrační management stanovuje implementační strategii systému, kontroluje realizaci přírůstků na shodu a architekturou systému a provádí zpětnou integraci dekomponovaných realizačních přírůstků do vzájemně provázaného celku. Tento integrační management je součástí metodik pro řízení programů a projektových portfolií a částečně je rozebrán rovněž v TOGAFu [9].

6 Závěr

Architektonický management je nezbytně nutné provádět u komplexních systémů zaváděných po segmentech přírůstkovým životním cyklem a to za účelem zajištění integrity předmětného systému. Jde o řízení základních vlastností každého systému označovaných v teorii systémů jako „vztah částí k celku“ a „vztah celku k částem“. V realizační praxi pak zpravidla mluvíme o integračním managementu nebo o systémové integraci.

Koncept kolem Enterprise Architecture dává návod, jak tento složitý úkol uchopit. Architektonické navrhování ukazuje, jak komunikovat se zainteresovanými stranami, jaká používat hlediska, jak uřídit konzistenci architektonických pohledů a jako postupovat při realizaci s ohledem na dosažení cílového stavu systému.

Literatura

1. COBIT®5: The Framework Exposure Draft, © 2011 ISACA. All rights reserved.
2. Hujňák, P.: Zkušenosti z aplikace architektonických standardů do praxe. Příspěvek konference Systémová integrace 2011. http://si.vse.cz/archive/presentations/2011/bpo-04-hujnak.pdf .
3. ISO 10006F:2003 Quality management systems -- Guidelines for quality management in projects. © ISO 2003.
4. ISO 9001:2008 Quality management systems – Requirements. © ISO 2008.
5. ISO/IEC 15288:2008 Guide for ISO/IEC 15288 (System Life Cycle Processes)
6. ISO/IEC 15288:2008 Systems and software engineering -- System life cycle processes. © ISO/IEC 2008.
7. ISO/IEC 27001:2005 Security techniques – information security maqnagement systems – Requirements. © ISO/IEC 2005.
8. ISO/IEC/IEEE 42010:2011 – Recommended Practice for Architectural Description of Software-intensive Systems. Final Draft. © ISO/IEC 2011 © IEEE 2011 Http://www.iso-architecture.org/ieee-1471.
9. The Open Group Architecture Framework (TOGAF). TOGAF Version 9. © 2009 The Open Group.
10. The Val IT Framework. Enterprise Value: Governance of IT Investments. ISBN 1-933284-32-3. Copyright © 2006 IT Governance Institute.

Annotation:

Enterprise Architecture and management of ICT assets

Architectures of IT intensive systems are the preferred approach to embrace their complexity. By the architecture we mean the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the integrity principles of its design and evolution. The today practice of architecting comes from different viewpoints out which the system is viewed by stakeholders and from mutual correspondence of these views into an integrated system architecture. The paper deals with theoretical cornerstones and practical experience of the author in the area of architectural design and architectural integration management.

Fonetické vyhledávání v cizojazyčných textech – jak najít relevantní informace o "as-sauratu fí misra," i když neumím arabsky

Iveta MRÁZOVÁ[1], Ondřej SÝKORA[2], František MRÁZ[3]

[1,2] *Katedra teoretické informatiky a matematické logiky, MFF UK Praha*
Malostranské nám. 25, 118 00 Praha
iveta.mrazova@mff.cuni.cz, ondrasej@matfyz.cz
[3]*Kabinet software a výuky informatiky, MFF UK Praha*
Malostranské nám. 25, 118 00 Praha
frantisek.mraz@mff.cuni.cz

Abstrakt. Současný globální svět představuje velkou výzvu pro tradiční způsoby ukládání, poskytování a vyhledávání užitečných informací. Krom dalších požadavků, např. na vyhledávání obrazových informací a videa, by měly moderní technologie pro dobývání znalostí z webu podporovat i efektivní vyhledávání zdrojů dostupných v cizích jazycích. Ovšem pro někoho, kdo daný jazyk plynně neovládá, může být správná formulace příslušného dotazu poměrně obtížná – obzvlášť v případě, že má zvolený jazyk jinou abecedu (např. arabština) a strojový překlad není podporovaný.

Mnoho uživatelů by v takovém případě uvítalo možnost vyhledávat relevantní informace jednodušeji, např. zadáním vyhledávaných slov či frází v takovém tvaru, jak je slyší. Fonetické vyhledávání by pak mělo vyhledat v textu všechna taková slova, která mají stejnou výslovnost jako slovo, které uživatel slyšel a zapsal. Přitom předpokládáme, že uživatel je v obecném případě cizinec, který mluví jinou řečí a může používat jinou abecedu a jiná přepisovací pravidla. Přesto však, pokud uslyší v televizi anebo na ulici např. frázi „as-sauratu fí misra," bude schopen najít informace o „revoluci v Egyptě."

V článku představíme jak tradiční fonetické algoritmy použitelné k řešení výše uvedeného problému, tak také moderní přístupy inspirované teorií konečných automatů, které jsou mimořádně efektivní především v případě velkých textových souborů. Součástí příspěvku je i analýza možných výhod programovatelného hardwaru (FPGA) pro fonetické vyhledávání, a to v závislosti na použitém hardwaru, vyhledávaném slově, zvoleném jazyku, dostupnosti slovníku a délce prohledávaného textu.

Klíčová slova: fonetické vyhledávání, FPGA, zpracování přirozeného jazyka.

1 Úvod

Moderní široce dostupné webové technologie umožňují snadný přístup k dokumentům uloženým v různých jazycích na různých místech světa. Pro uživatele, který však není příliš zběhlý v příslušném jazyce, může být formulace správného dotazu značně problematická – obzvlášť v případě, kdy zvolený jazyk používá jinou abecedu – např. arabština – a strojový překlad není k dispozici. Pro podporu jednoduchého, nicméně efektivního vyhledávání informací i v takovém případě je možné použít tzv. fonetické vyhledávání [10] pomocí automatů Aho-Corasickové. Cílem fonetického vyhledávání je vyhledat v textech všechna taková slova, která mají stejnou (anebo z pohledu uživatele velmi podobnou) výslovnost

jako slovo, které uživatel slyšel a zapsal. Uživatel přitom může být cizinec, který používá jinou abecedu a jiný způsob zápisu než cílový jazyk.

Pro hledání slov se stejnou výslovností se používají připravené sady přepisovacích pravidel. Pomocí těchto přepisovacích pravidel sek fonetickému zápisu slova hledají všechny možné zápisy slov v cílovém jazyku a ty se potom vyhledávají zvoleným typem automatu. Příliš složitá přepisovací pravidla navržená pro práci s nejednoznačnou výslovností však mohou být spojena s extrémními paměťovými nároky. Vzhledem k aktuálnímu vývoji v oblasti programovatelného hardware (FPGA – Field-Programmable Gate Array) a paralelnímu charakteru výpočtů probíhajících v alternativním modelu nedeterministických konečných automatů tak vyvstává přirozená otázka: „Mohla by být FPGA-implementace fonetického vyhledávání efektivnější než tradiční přístup založený na využití CPU?" To, zda je efektivnější použít pro fonetické vyhledávání FPGA-technologie anebo tradiční přístup, bude záviset na použitém jazyce, vyhledávaném řetězci (tj. zda se jedná o slova obsažená ve slovníku nebo spíše o jména apod.), délce prohledávaného textového dokumentu a použitém hardware. Náš příspěvek se zaměří na návrh a analýzu metodologie pro podporu rozhodování výše uvedeného typu.

Následující kapitola podává stručný přehled základních principů programovatelného hardware. V kapitole tři představíme vybrané modely konečných automatů použitelné pro fonetické vyhledávání – deterministické automaty Aho-Corasickové a nedeterministické konečné automaty. Jejich implementace a návrh tzv. vícevrstvých nedeterministických automatů jsou předmětem čtvrté kapitoly. Pátá kapitola představuje novou metodologii pro rozhodování, zda pro fonetické vyhledávání použít FPGA-implementaci. Výsledky srovnávacích experimentů provedených pro němčinu, arabštinu a češtinu analyzuje kapitola šestá (některé z těchto výsledků již byly publikovány v [11]). Závěrečná kapitola shrnuje dosažené výsledky a podává nástin některých otevřených problémů z této oblasti.

2 Programovatelný hardware a jeho využití při vyhledávání v textech

Tradiční algoritmy použitelné i pro fonetické vyhledávání – např. Soundex [15], Kolínská fonetika [13], PHONEM [24], arabský Soundex a arabský Phonix [1] – jsou optimalizované pro vyhledávání jmen v databázích. Každému dotazu v podstatě přiřadí jednoduchý typ kódu tak, aby všechna slova s podobným zápisem dostala stejný kód. Mnohá slova je ale možné zapsat různým způsobem (anebo mohou mít dokonce různý význam), přestože mají stejnou (anebo velice podobnou) výslovnost. Postup navržený v [10] měl proto umožnit přijímat všechna slova s výslovností podobnou zadanému vstupnímu slovu namísto hledání (jediného) kódu, který by byl přiřazen všem takovýmto slovům.

Na první pohled taková myšlenka nevypadá příliš efektivně. Množina slov s podobnou výslovností totiž může být poměrně velká a s rostoucí délkou vstupního slova bude narůstat. Pro dlouhá slova bude i zkonstruovaný akceptor složitější a pomalejší. Tuto nevýhodu lze ovšem omezit použitím algoritmu Aho-Corasické [1]. Vhodnou alternativou k algoritmu Aho-Corasickové pak mohou být regulární výrazy [19]. Ke každému regulárnímu výrazu totiž můžeme sestrojit konečný automat, který přijímá právě tu množinu řetězců reprezentovaných daným regulárním výrazem.

Větší paměťové i časové nároky spojené s konstrukcí automatu Aho-Corasickové ovšem představují jistou nevýhodu jen v případě velmi dlouhých slov, která se budou vyhledávat ve velmi krátkých textech. Poté, co se automat Aho-Corasickové vytvoří, probíhá vlastní vyhledávání už deterministicky a poměrně rychle. V [10] se podařilo ukázat, že vyhledávání pomocí automatu Aho-Corasickové je v průměru čtyřikrát rychlejší než implementace regulárních výrazů (přijímající tatáž slova) v Perlu. Na tradičních

počítačových architekturách je pak vyhledávání pomocí regulárních výrazů rychlejší pouze pro krátká slova vyhledávaná v krátkých textových souborech (menších než 1 MB.

Současné aplikace výše uvedených modelů zahrnují jak tradiční přístupy využívající CPU, tak také moderní, nově vyvinuté techniky založené na programovatelném hardware. Klasické centrální výpočetní jednotky (CPU) zpracovávají programy sekvenčně (případně na omezeném počtu navzájem oddělených vláken). Paralelizované výpočty implementované pomocí „programovatelného hardware“ by proto mohly efektivitu univerzálních CPU podstatně zvýšit [20].

Pod pojmem „programovatelný hardware“ chápeme většinou hardware, který může na základě softwarové konfigurace změnit své uspořádání i vzájemné propojení svých výpočetních prvků. Existují dva základní typy programovatelných obvodů – tzv. programovatelná pole hradlových obvodů (FPGA-desky) a složitá programovatelná logická zařízení (CPLD – Complex Programmable Logic Device). FPGA-desky jsou konstruované formou polí programovatelných jednotek označovaných jako „logické bloky“. Jednotlivé logické bloky je možné vzájemně propojovat a konfigurovat pro výpočet kombinací logických funkcí. Alternativně mohou sloužit jako jednoduchá logická hradla. Běžné FPGA-desky obsahují deseti- až statisíce logických bloků.

Základní jednotky se v případě CPLD nazývají „makrobuňky“ a podporují implementaci logických výrazů v disjunktivní normální formě. Kapacita dostupných CPLD odpovídá FPGA-deskám s tisíci až desetitisíci logickými bloky. Pro jednoduchost budeme však v textu označovat pojmem FPGA oba typy programovatelného hardwaru. FPGA-zařízení lze programovat pomocí programovacího jazyka Verilog. Vytvořené programy lze nicméně snadno převést z jedné třídy programovatelných zařízení na druhou.

Ve srovnání s univerzálními počítači nabízí programovatelný hardware jen velmi omezené množství pracovní paměti a omezenou množinu programovacích prostředků [12]. Zato podporuje extrémně vysokou rychlost zpracování a masivní paralelismus uvnitř obvodů. Výrobci Altera [3] a Xilinx [25] vyvinuli i modely, které lze k běžnému počítači připojit přes PCI (anebo PCI Express) sběrnici. To umožňuje zpracovávat data efektivněji v paměti počítače [7]. Programovatelný hardware nalezl široké uplatnění především v oblastech, kde je kritická maximální propustnost, jako např. ve zpracování signálů a v počítačových sítích. Klesající ceny hardware pak z této technologie činí ideální platformu pro vytváření síťových monitorovacích zařízení na zakázku.

Existující přístupy pro vyhledávání řetězců pomocí FPGA se zaměřují především na systémy pro detekci a prevenci průniků (IDS – Intrusion Detection System) [4]. Ve smyslu propustnosti zde mohou FPGA- implementace řádově překonat tradiční „softwarové IDS“ systémy jako např. Snort [14]. IDS-systémy kontrolují nejenom hlavičku (header) paketu, ale prohledávají také obsah zpracovávaných paketů s cílem najít hledané vzory. Pokud jsou na vstupu takové vzory detekovány, znamená to, že se po síti šíří útok (atak). Obecně vyhledávají IDS-systémy odpovídající slova (výrazy) podle množiny pravidel, kterou navrhl administrátor. Tato pravidla obsahují informaci o požadované IP-adrese, TCP-hlavičce a často i o vzorech, které je třeba v proudu dat lokalizovat. V současné době dosahují počty vzorů v IDS-databázích tisícovek vzorů, což už představuje z hlediska výpočetní složitosti poměrně hodně náročnou úlohu.

Ke snížení paměťových nároků algoritmu Aho-Corasickové použili [8] a [6] v FPGA-implementacích pro IDS rozdělení automatu Aho-Corasickové do několika menších automatů. Podobně jako jiné implementace stavových strojů, které předpokládají v každém cyklu nějaký přechod, mají totiž i automaty Aho-Corasickové vzhledem k dostupným FPGA obrovské paměťové nároky. Tento problém způsobuje velké množství hran (odpovídající velikosti vstupní abecedy), které vedou ke všem možným následujícím stavům.

S podobným cílem vyvinuli i Mitra, Najjar a Bhuyan [9] přístup založený na převodu Perl-kompatibilních regulárních výrazů (PCRE) do nedeterministických konečných automatů, které následně běží na FPGA. Scarpazza, Villa a Petrini [17] nicméně ukázali, že pro malé množiny vyhledávaných frází může být efektivnější spouštět vysoce optimalizované vyhledávací algoritmy na univerzálních CPU.

3 Analyzované automaty

Automaty Aho-Corasickové použité pro fonetické vyhledávání [10] patří ke třídě deterministických konečných automatů definovaných následujícím způsobem:

Definice 3.1.
Deterministický konečný automat (KA) *je uspořádaná pětice* $A = (Q, \Sigma, \delta, q_0, F)$, *kde:*

- Q *je konečná neprázdná množina stavů,*
- Σ *je konečná neprázdná vstupní abeceda,*
- $\delta : Q \times \Sigma \rightarrow Q$ *je přechodová funkce,*
- $q_0 \in Q$ *je počáteční stav,*
- $F \subseteq Q$ *je množina přijímajících stavů.*

Slovo $w = w_1 w_2 \ldots w_n$, *kde* $w_1, w_2, \ldots, w_n \in \Sigma$, *je přijímané automatem A, jestliže existuje konečná posloupnost stavů* $q_1, \ldots, q_{n+1} \in Q$ *taková, že* $q_1 = q_0$, $q_{n+1} \in F$ a $\forall\ i \in \{1, \ldots, n\}$ *platí, že* $q_{i+1} \in \delta(q_i, w_i)$. *Prázdné slovo je automatem přijímáno, pokud* $q_0 \in F$.

Jazyk rozpoznávaný daným automatem je množina slov, které automat přijímá. Z teoretického hlediska existuje ke každému konečnému automatu nekonečně mnoho ekvivalentních automatů přijímajících přesně ten samý jazyk. Vzhledem k praktickým důvodům však obvykle preferujeme co možná nejmenší automat (tj. ten s minimálním počtem stavů), který přijímá daný jazyk.

3.1 Automat Aho-Corasickové

Algoritmus Aho-Corasickové je jednoduchá, přesto však efektivní technika, která ve zpracovávaném textu lokalizuje všechny výskyty řetězců ze zadané konečné množiny řetězců. Algoritmus Aho-Corasickové konstruuje vyhledávací stroj (mírně rozšířený konečný automat) pro vyhledávání požadovaných řetězců zapsaných obecně v libovolné abecedě. Pro konečnou množinu slov $S = \{w_1, \ldots, w_s\}$ se tak nejprve zkonstruuje vyhledávací stroj sestávající ze třech funkcí – dopředné $g : Q \times \Sigma \rightarrow Q$, chybové $f : Q \rightarrow Q$ a výstupní $o : Q \rightarrow \mathcal{P}(S)$, kde $\mathcal{P}(S)$ označuje množinu všech podmnožin množiny S.

Funkce g a f odpovídají přechodům vytvářeného konečného automatu, výstupní funkce udává pro každý stav automatu, zda byl při jeho dosažení lokalizován nějaký řetězec z S. Stavy automatu odpovídají řetězcům z P(S), tedy množině všech prefixů (předpon) slov z S. Počáteční stav odpovídá prázdnému řetězci, koncové stavy charakterizují ty řetězce z $P(S)$, jejichž sufix (přípona) patří do $P(S)$.

Při vyhledávání se tedy k zadaným klíčovým slovům (vytvořeným podle použitých přepisovacích pravidel) nejprve vytvoří pomocí Algoritmu 1 a Algoritmu 2 odpovídající konečný automat. Poté se jediným průchodem zpracuje prohledávaný vstupní text pomocí Algoritmu 3 a předem vytvořeného stroje pro rozpoznávání vzorů. Deterministické automaty Aho-Corasickové lze přitom vytvořit v čase, který je lineárně závislý na součtu

délek klíčových slov. Počet stavových přechodů uskutečněných během prohledávání vstupních dat nicméně na počtu klíčových slov už nezávisí.

Pro konečnou množinu slov S tedy algoritmus Aho-Corasickové zkonstruuje stroj s konečným počtem stavů, který přijímá právě ta slova, která mají příponu z S. Vlastní konstrukce má dvě fáze. Během první z nich se vytvoří deterministický konečný automat M pro slova z S – viz Obrázek 1. Každý z jeho stavů (uzlů) odpovídá nějakému řetězci, např. ‚*head'*. Do každého uzlu vedou přechody (hrany) z uzlu, který reprezentuje nejdelší vlastní předponu příslušného řetězce – např. do uzlu odpovídajícího řetězci ‚*hea'* vede hrana z uzlu odpovídajícího ‚*he'*. Každá hrana je označena odpovídajícím vstupním symbolem, např. *a*. Počáteční stav odpovídá prázdnému řetězci, zatímco koncové stavy odpovídají vyhledávaným řetězcům. Přechody definují dopřednou funkci g (tzv. *goto*-funkci) konstruovanou Algoritmem 1. Hodnotou funkce g pro stav odpovídající např. řetězci '*he*' a vstupní symbol *a* je stav odpovídající řetězci ‚*hea'*.

Během konstrukce stavového stroje se odvozují i dvě další funkce – chybová a výstupní. Chybovou funkci konstruuje Algoritmus 2. Pro každý stav p, který odpovídá řetězci $w_p \in P(S)$ vrací chybová funkce stav q, který odpovídá řetězci $w_q \in P(S)$. Řetězec w_q je nejdelší vlastní příponou řetězce odpovídajícího stavu p, který je zároveň předponou jiného vyhledávaného řetězce. Výstupní funkce vrací množinu vyhledaných řetězců odpovídajících danému stavu. Po ukončení své konstrukce načte stroj M vstupní text. Na každé pozici, na které končí výskyt slova z S, vypíše M značku se seznamem vyhledaných slov z S. Během prohledávání automat prochází nejdelší společnou část vstupního textu a vyhledávaného řetězce pomocí funkce g. Pokud si zpracovávané znaky neodpovídají, určí následující stav stroje chybová funkce f.

Algoritmus 1: *Konstrukce dopředné funkce a částečné výstupní funkce*

VSTUP: Množina klíčových slov $S = \{ w_1 , \ldots , w_s \}$.
VÝSTUP: dopředná funkce g (reprezentovaná orientovaným *goto*-grafem, jehož uzly odpovídají stavům vyhledávacího stroje) a část výstupní funkce o.
Hodnota výstupní funkce o se pro každý nově vytvořený stav nastaví na prázdnou množinu.

Krok 1: Vytvoř počáteční stav q_0 reprezentovaný v *goto*-grafu uzlem, který odpovídá prázdnému řetězci.
Krok 2: Postupně vlož do *goto*-grafu všechna slova $w_1 , \ldots , w_s$:
- Krok 2.1: Začni v počátečním stavu q_0 a procházej vkládané slovo. Dokud odpovídá prefix zpracovávaného slova cestě některého z dříve vložených řetězců, sleduj ji.
- Krok 2.2: Pokud dojde k neshodě, vytvoř v *goto*-grafu novou větev.
- Krok 2.3: Vkládané slovo přidej do výstupní funkce stavu, v němž vkládání symbolů skončilo.

Krok 3: Pro každý symbol, který ještě nebyl použitý pro přechod z počátečního stavu, přidej smyčku pro návrat automatu do počátečního stavu q_0.

Algoritmus 2: *Konstrukce chybové funkce, dokončení konstrukce výstupní funkce*

VSTUP: Orientovaný *goto*-graf dopředné funkce g a částečná *výstupní* funkce o z **Algoritmu 1**.
VÝSTUP: Chybová funkce f a výstupní funkce o.

Algoritmus prochází *goto*-graf do šířky.

Krok 1: Pro stavy v hloubce 1 inicializuj chybovou funkci: nastav $f(s) := q_0$ pro všechny symboly a takové, že $g(q_0,a) = s$ a $s \neq q_0$.

Krok 2: Na základě chybové funkce pro stavy v nižších hloubkách iterativně urči chybovou funkci pro stavy s v hloubce d:

Krok 2.1: Pro všechna a taková, že $g(r,a) = s$, nastav $state := f(r)$.

Krok 2.2: Dokud $g(state,a) \neq q_0$, přiřaď $state := f(state)$.

Krok 2.3 Nastav $f(s) := g(state,a)$.

Krok 2.4: Uprav výstupní funkci $o(s) := o(s) \cup o(f(s))$.

Algoritmus 3: *Vyhledávání klíčových slov*

VSTUP: Slovo $x = x_1x_2...x_n$, kde x_i $(1 \leq i \leq n)$ jsou vstupní symboly, a stroj M pro rozpoznávání vzorů s dopřednou funkcí g, chybovou funkcí f a *výstupní* funkcí o.

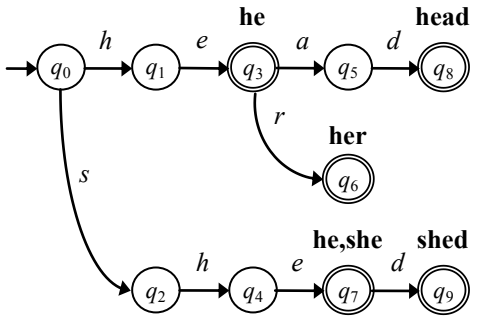
VÝSTUP: Všechny pozice v x, na nichž se vyskytují klíčová slova, spolu se seznamem jejich výskytů.

Krok 1: Začni ze stavu s = q_0 a opakuj pro $i = 1, \ldots, n$

Krok 1.1: Dokud $(g(s, x_i) = q_0$ a $s \neq q_0)$, opakuj $s := f(s)$.

Krok 1.2: $s := g(s, x_i)$.

Krok 1.3: Jestliže $o(s)$ je neprázdné, vypiš pozici i a $o(s)$.



i	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
$f(q_i)$	q_0	q_0	q_0	q_0	q_1	q_0	q_0	q_3	q_0	q_0

Obrázek 1: Příklad automatu Aho-Corasickové pro vyhledávání řetězců *'he', ,she', her, shed, head*. Dvojitým kroužkem jsou označeny stavy s neprázdným výstupem, hodnota výstupní funkce je napsaná tučně nad příslušným stavem.

3.2 Nedeterministické konečné automaty

Nedeterministické konečné automaty jsou zobecněním deterministických konečných automatů. Mají obecně několik počátečních stavů a jejich současný stav a vstupní symbol určují jen množina možných následujících stavů.

Definice 3.2.

Nedeterministický konečný automat (NKA) *je uspořádaná pětice* $A = (Q, \Sigma, \delta, I, F)$, *kde:*

- Q *je konečná neprázdná množina stavů,*

- Σ *je konečná neprázdná vstupní abeceda,*
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ *je přechodová funkce (*$\mathcal{P}(Q)$ *označuje množinu všech podmnožin množiny stavů* Q*),*
- $I \subseteq Q$ *je množina počátečních stavů,*
- $F \subseteq Q$ *je množina přijímajících stavů.*

Slovo $w = w_1 w_2 \ldots w_n$*, kde* $w_1, w_2, \ldots, w_n \in \Sigma$*, je přijímané automatem A, jestliže existuje konečná posloupnost stavů* $q_1, \ldots, q_{n+1} \in Q$ *taková, že* $q_1 \in I$, $q_{n+1} \in F$ *a pro každé* $i \in \{ 1, \ldots, n \}$ *platí, že* $q_{i+1} \in \delta(q_i, w_i)$.

Prázdné slovo je automatem přijímáno, pokud $I \cap F \neq \emptyset$.

Na druhou stranu lze ke každému nedeterministickému automatu A s m stavy sestrojit deterministický automat A' s maximálně 2^m stavy, který rozpoznává tentýž jazyk jako A. Obrovské množství stavů vzniklých při převodu nedeterministického automatu na deterministický lze poněkud omezit tzv. redukcí automatu A', která hledá k automatu A' ekvivalentní automat s minimálním počtem stavů. Výsledný minimální automat však může, bohužel, stále ještě vyžadovat exponenciální množství stavů vzhledem k m. Sestrojit odpovídající deterministický automat pak pro větší hodnoty m nemusí být prakticky možné.

Každý jazyk přijímaný (deterministickým i nedeterministickým) konečným automatem lze definovat i pomocí tzv. regulárních výrazů. Každý regulární výraz je možné převést na ekvivalentní nedeterministický automat, který bude přijímat stejný jazyk, avšak počet jeho stavů nebude příliš překračovat počet symbolů obsažených v daném regulárním výrazu.

4 Implementace automatů

Zásadní motivací pro náš výzkum v oblasti FPGA-technologií byla eventualita efektivního vyhledávání v mnohem kratším čase než pomocí standardních implementací na PC. K ověření této hypotézy jsme vytvořili jednoduché simulátory uvažovaných automatů pro PC. Deterministické automaty jsou zde reprezentovány pomocí dvourozměrných polí definujících přechodovou funkci – první dimenze je určena stavem automatu, druhá dimenze kóduje znaky vstupní abecedy. Hodnota obsažená v poli pro daný stav a symbol určuje kód nového stavu. Efektivita implementace přitom nebude příliš záviset na počtu stavů konstruovaného automatu, pokud se celý vejde do RAM počítače. Výhodou budou automaty s malým počtem stavů, protože se celé vejdou do vyrovnávací paměti (cache) CPU.

V reálných aplikacích však bude funkce automatu Aho-Corasickové silně záviset na charakteru zpracovávaného textu a na jazyce používaném v textu. Stavy bližší k počátečnímu stavu budou přirozeně navštěvované častěji než stavy bližší ke koncovým přijímajícím stavům. Automat vytvořený např. pro hledání arabského slova قال [on řekl] podle fonetického předpisu ´kála´ má několik desítek stavů. Při zpracovávání 250 MB vzorku textu extrahovaného z arabské Wikipedie strávil tento automat téměř 99.5% času pouze v prvních třech stavech (a více než 99.9% času v prvních deseti stavech). Limitujícím faktorem v případě automatů Aho-Corasickové je tedy spíše velikost dostupné operační paměti, kde je třeba uložit přechodovou tabulku.

Pro odhad doby běhu FPGA-implementací jsme použili simulátor poskytovaný výrobcem. Časová simulace přitom zohledňuje i zpoždění signálů, která mohou hrát v reálných obvodech významnou roli, a jejichž opomenutí by mohlo vést k chybám v obvodu. V případě deterministických automatů je navíc automat vždy v jediném dobře definovaném stavu, který závisí pouze na řetězci zpracovaném automatem během předchozích kroků. Při FPGA-implementaci pak lze konfiguraci automatu

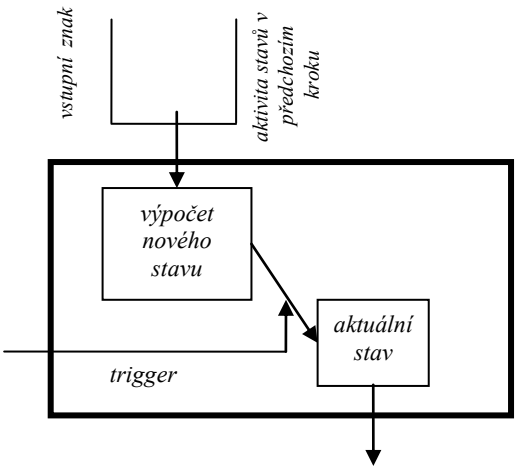
zakódovat přímo ve stavovém registru. Během stavových přechodů vyvolaných změnou vstupu lze uloženou konfiguraci samozřejmé také aktualizovat. Při odhadu času potřebného ke zpracování jednoho vstupního symbolu jsme proto měřili zpoždění mezi změnou na vstupu obvodu a odpovídající změnou na výstupu. Naměřené hodnoty tedy nezahrnují čas potřebný k načtení následujícího znaku ze vstupu.

4.1 Implementace nedeterministických automatů

Jednoduchý způsob, jak implementovat nedeterministické automaty na klasickém PC, je uložit aktivaci všech stavů do jediného pole Booleovských hodnot. V každém kroku výpočtu automat toto pole projde. Množinu aktivních stavů pro následující krok určí podle přechodů aktuálně aktivních stavů a odpovídajícího aktuálního symbolu. Pseudokód pro simulaci nedeterministických automatů na standardním PC ukazuje Algoritmus 4 dole:

Algoritmus 4: PC-simulace výpočtu nedeterministického automatu $A = (Q, \Sigma, \delta, I, F)$ nad vstupním textem $T = t_1 t_2 ... t_n$, kde $t_1, t_2, ..., t_n \in \Sigma$

```
1: Inicializuj pole stavů aktuální a následující
2: V poli aktuální aktivuj všechny počáteční stavy z I
3: for i = 1 to n do
4:   Nastav aktivitu všech stavů z pole následující na neaktivní.
5:   R ← seznam přechodů aktivovaných vstupním symbolem t_i.
6:   for all stav s do
7:     if s je aktivní then
8:       for all přechody (s → s´) ∈ R ze stavu s do
9:         Aktivuj stav s´ v poli následující.
10:      end for
11:    end if
12:  end for
13: end for
```



Obrázek 2 Schéma jednoho stavového modulu použitého při FPGA-implementacích nedeterministických automatů.

Abychom mohli využít paralelní charakter výpočtů na FPGA-hardwaru, měly by být spouštěné implementace automatů schopné aktualizovat všechny dosažitelné stavy najednou. Protože však FPGA nepodporují aktualizaci registrů z různých bloků kódu, implementovali jsme stavy automatů ve Verilogu jako jednotlivé moduly. Každý z těchto stavových modulů pak obsahuje pouze prostředky nezbytné pro uchování aktuálního stavu a pro jeho změnu v závislosti na aktuálním signálu ze vstupu.

Schéma modulu pro jeden stav ukazuje Obrázek 2. *Vstupní znak* označuje kód zpracovávaného znaku na vstupu automatu. *Aktivita stavů* v předchozím kroku kóduje aktivitu stavů, ze kterých vedou přechody do daného stavu. Pokud by např. v nějakém automatu vedly do stavu q_{15} přechody ze stavu q_7 pro přečtený symbol b, ze stavu q_4 pro přečtený symbol c a ze stavu q_{12} pro přečtený symbol b, počítá blok „výpočet nového stavu“ pro stav q_{15} formuli (($state_7$ AND in_b) OR ($state_4$ AND in_c) OR ($state_{12}$ AND in_b)), kde $state_i$ je aktivita stavu q_i v předchozím kroku a $in_x = 1$ právě tehdy, když je vstupní symbol roven x. Vstup *trigger* slouží pro synchronizaci aktualizace stavů ve všech modulech.

4.2 Vícevrstvé nedeterministické automaty

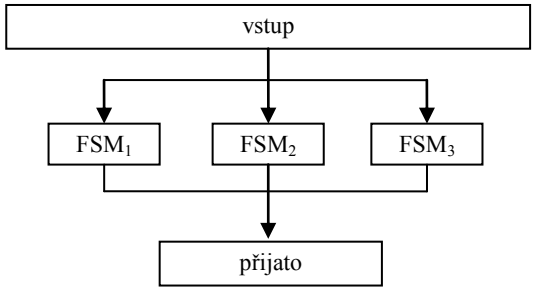
Při simulaci nedeterministických automatů je hlavním problémem synchronizace stavových přechodů. Jejím cílem je zabránit aktualizaci stavových registrů, dokud není dokončen výpočet pro nově načtený vstupní symbol. U FPGA-desek, které jsme používali, je frekvence interních hodin nastavená na 50 MHz (což odpovídá synchronizační periodě 20ns). V současné době jsou ovšem na trhu dostupné i FPGA-desky s frekvencemi dosahujícími až k 500 MHz. Pokud je ale doba potřebná pro výpočet aktivity stavů v následujícím kroku kratší než synchronizační perioda, bude doba strávená čekáním na synchronizaci vynaložená zbytečně, a to pro jakoukoliv synchronizační periodu.

Časově náročnou synchronizaci by bylo možné omezit, pokud by byla informace o aktuálně aktivních stavech uložena v odděleném vyrovnávacím obvodu (bufferu) anebo se používala jen jako výstup automatu. Kvůli efektivnějšímu zpracování několika vstupních symbolů najednou jsme se rozhodli zahrnout do modelu jednu vrstvu stavových registrů pro každý symbol z této dávky. Informace o aktivních stavech se pak přivede jako výstup z jedné vrstvy na vstup vrstvy bezprostředně následující. Po zpracování celé dávky symbolů je synchronizace nutná už pouze pro aktualizaci registrů z první vrstvy, a to v závislosti na předchozích stavech poslední vrstvy. Výměnou za větší množství stavových registrů tak získáme (lineární) úsporu času spotřebovaného při čekání na synchronizaci, a to v závislosti na počtu vstupních znaků zpracovávaných najednou v každé dávce.

S ohledem na omezení daná pro počet vstupních/výstupních pinů použité FPGA-desky jsme mohli testovat nedeterministické automaty s až 50 vrstvami. Čas potřebný na zpracování dat přirozeně klesá s počtem vrstev, přičemž nejlepších výsledků bylo dosaženo při maximálním možném počtu vrstev (50 v našem případě). Tyto výsledky už jsou porovnatelné s deterministickými automaty spouštěnými na PC.

5 FPGA-technologie ve fonetickém vyhledávání

Při fonetickém vyhledávání předpokládáme, že jak vyhledávané slovo, tak také zpracovávaný text budou uloženy v paměti (standardního) počítače. Podobně budeme předpokládat, že i vyhledaná slova budou dále zpracovávána ve (standardním) počítači. V FPGA-implementacích proto hraje důležitou roli i rychlá komunikace mezi počítačem a



Obrázek 3 Schéma paralelní implementace tří konečných automatů (FSM). Každý z těchto automatů přijímá příslušnou podmnožinu vyhledávaných slov. Vstupní slovo je tímto vícevrstvým nedeterministickým automatem přijímáno, pokud je přijímáno kterýmkoliv z těchto tří automatů.

FPGA-deskou. FPGA-desky jsou obvykle samostatné desky, které lze k počítači připojit přes USB-port. Alternativně se dodávají na rozšiřitelné kartě, kterou lze k PC připojit přes PCI Express sběrnici. Vzhledem k tomu, že je maximální přenosová rychlost USB 2.0 60 MB/s příliš nízká ([21]), nepřipadá připojení FPGA přes USB v úvahu. Nicméně FPGA-desky připojené přes PCI Express sběrnici s maximální přenosovou rychlostí 16 GB/s pro PCI Express 16x už podle [23] představují použitelné řešení. V tomto případě jsou limitujícími faktory spíš rychlost paměti, ve které je zpracovávaný text uložen, a efektivita výpočtů prováděných na FPGA-desce.

V našem případě byly programy pro FPGA generovány a překládány pomocí standardních vývojových prostředků. Po načtení kódu však požadované stavové přechody probíhaly už na FPGA-desce. Celý proces použití konečných automatů ve fonetickém vyhledávání se tak skládá z následujících čtyř kroků:

1. *Uživatel zadá vyhledávané slovo* – tak, jak ho slyší, a se stejnou transkripcí, jakou používá ve svém mateřském jazyce.
2. *Zkonstruuje se automat, který přijímá všechna slova ze slovníku podobná (s ohledem na použitá přepisovací pravidla) vyhledávanému slovu* – buď deterministický automat Aho-Corasickové nebo analogický nedeterministický (vícevrstvý) konečný automat. Struktura konstruovaného automatu (tj. jeho stavy a přechody) se zakóduje jako samostatný programu ve Verilogu. Čas potřebný pro provedení této fáze procesu sahá od několika milisekund až k desítkám minut v závislosti na délce a/nebo složitosti vyhledávaného slova a použitém slovníku – viz odhady z Tab. 1.
3. *Kód automatu napsaný ve Verilogu se přeloží a optimalizuje pro FPGA* – např. co do počtu stavů, velikosti vstupní abecedy, atd. Pro malé automaty s až několika stovkami stavů trvá kompilace obvykle méně než minutu, zatímco pro velké automaty s řádově desítkami tisíc stavů může trvat na standardním PC dokonce i několik hodin.
4. *Přeložený kód automatu se nahraje ze (standardního) PC na FPGA-desku* – a může být dále použit pro detekci vyhledávaných slov ve vstupním textu. Načtení kódu na FPGA-desku zabere obvykle několik sekund.

Kód je třeba rekompilovat pokaždé, když uživatel zadá nové vyhledávané slovo. Nicméně aktuálně dostupné FPGA-desky takovéto změny kódu podporují bez toho, že by

Tab. 1: Čas potřebný ke konstrukci automatů Aho-Corasickové (ACA) a nedeterministických konečných automatů (NKA) používaných při fonetickém vyhledávání. Uvažované automaty byly generovány na standardním PC.

Typ konstrukce	**Trvání**
ACA pro až stovky řetězců	milisekundy až sekundy
ACA pro až stovky tisíc řetězců	až minuta
ACA pro více než stovky tisíc řetězců	několik minut
NFA pomocí regulárních výrazů a normalizace	sekundy až minuty
NFA vytvářené přímo z přepisovacích pravidel	milisekundy až sekundy

bylo nutné desku zastavovat anebo restartovat. Standardní CPU-implementace naproti tomu nemusí generovat žádné programy ve Verilogu, ani je překládat či načítat.

6 Provedené experimenty

Ke generování příslušných konečných automatů (typu Aho-Corasickové i nedeterministických) se používají přepisovací pravidla závislá na zvolené dvojici jazyků. Použitá pravidla zohledňují fonetickou podobu cílového jazyka tak, jak ji slyší cizinec (v našem případě Čech). Sady pravidel, které jsme použili pro němčinu a arabštinu, jsou podobné sadám aplikovaným v [10] a lze je najít v Tabulce 2 (česko-německá přepisovací pravidla) a v Tabulce 3 (česko-arabská přepisovací pravidla).

Fonetické vyhledávání pomocí automatů Aho-Corasickové umožňuje přijímat velké množství různých slov (dokonce stovky a tisíce slov). Tato slova jsou ovšem z větší části gramaticky chybná a v (reálných) textech se téměř nevyskytují. Zřejmě proto je fonetické vyhledávání ve srovnání se Soundexovými algoritmy překvapivě přesné, a to zejména pro arabštinu. Prostřednictvím slovníku lze jeho efektivitu nicméně dále zvýšit – pro podrobnější diskuzi viz níže. Použitý německý slovník [22] obsahoval 500 000 slov, arabský slovník obsahoval 30 000 slov [16]. Alternativní nedeterministické automaty jsme vytvářeli z regulárních výrazů odpovídajících příslušnému automatu Aho-Corasickové.

Testovaná data obsahovala 124 náhodně zvolených německých a 36 arabských slov. Ke každému z těchto slov, která budeme označovat jako cílová, jsme měli k dispozici i příslušné vyhledávané slovo zakódované pomocí „české“ výslovnosti cílového slova (např. pro německé slovo „lednička“: *Kühlschrank* → *kýlšrank*). Při vyhledávání německých slov jsme používali tři různé textové soubory (10 MB, 1 GB a 100 GB) vytvořené z náhodně zvolených článků z německé Wikipedie. Ve zpracovávaných souborech byla všechna velká písmena změněna na malá a pro vícero po sobě jdoucích stejných znaků byl ponechán vždy jen jeden – např. pro slovo „talíř“: *Teller* → *teler*. Pro arabštinu jsme použili stejné množství textu extrahovaného z arabské Wikipedie a předzpracování podobného typu specifické pro arabštinu, které např. nerozlišuje délku samohlásek v českém zápisu hledaného slova ani vokalizaci arabského textu.

Hardware použitý pro vlastní implementaci představuje další důležitý faktor – v našich experimentech jsme uvažovali standardní přístup založený na CPU (spouštěný na běžném PC s procesorem Intel Core2 Duo se 2.4 GHz a 2 GB RAM; na PC běžela 32 bitová verze Ubuntu Linuxu bez jakýchkoliv úprav) i přístup založený na využití FPGA (používali jsme FPGA čip Altera Cylone II na desce Altera DE 2).

Tab. 2: Česko-německá pravidla

česky	něm.	česky	něm.	česky	něm.	česky	něm.
a, á	a	o, ó	o	c	tz	i, í, y, ý, j	i
ä, e, é	ä	ö, é, e	ö	oj	eu, äu	á	aa, ah
b	b	p	p	aj	ei, ai, ay, ey	a	aa, ah
c, k	c	q, kv	q	ai	ei, ai, ay, ey	é, e	ee, eh, oe
d, t	d	r	r	Š	sch, ch	é, e	äh, öh
e, é	e	s, z	s	šp	sp	ä	äh
g, k	g	t, th	t	št	st	ö	öh
h	h	u, ú, ů	u	č	tsch, tzch	ü, y, ý, i, í	ü
l	el	ü	üh	k	ck, ch	i, í, y, ý, j	y
j	j	v, f	v	kv	qu	ý	ie
k	k	v, w	w	ich, ik	ig	ó	oo
l	l	x, ks	x	ks	chs	o	oo, oh
m	m	í	ie	t	dt, th	u, ú, ů	uh
n	n	z, c	z	n	ng, en	u, ú, ů	uu
r	er	ss, s	ß, ss	f	pf, ph		

V praxi bude efektivita implementovaného systému záviset i na použitém modelu – v rámci našeho výzkumu jsme testovali standardní automaty Aho-Corasickové i paralelní modely nedeterministických konečných automatů, které představují vhodnou alternativu k regulárním výrazům. [10] nicméně ukazuje, že na tradičních počítačích je vyhledávání pomocí regulárních výrazů rychlejší jen pro krátká slova vyhledávaná v krátkých textových kolekcích (méně než 1 MB). Tam je totiž nevýhodou delší čas potřebný na konstrukci automatu Aho-Corasickové. Z tohoto důvodu jsme u nedeterministických automatů uvažovali dále jen jejich FPGA-implementace.

Abychom porovnali efektivitu standardních nedeterministických automatů s jejich vícevrstvou variantou, uvažovali jsme v testech i různé počty vrstev. Kvůli omezenému počtu vstupně/výstupních pinů na použité FPGA-desce jsme byli schopni testovat obvody s nanejvýš 50 vrstvami. Provedené testy potvrdily, že časové nároky klesají s počtem vrstev a nejlepších výsledků lze dosáhnout pro maximální uvažovaný počet 50 vrstev.

Pro efektivní a spolehlivé fonetické vyhledávání je však rozhodující použití slovníku. Fonetické vyhledávání totiž může vést k obrovskému množství přijímaných. Automaty vytvářené s cílem přijmout všechna uvažovaná slova by proto měly příliš mnoho stavů. Pokud bychom ale nejprve (foneticky) prohledali seznam možných slov jazyka (slovník), získali bychom typicky mnohem menší množinu gramaticky správných slov cílového jazyka, která mají výslovnost podobnou hledanému slovu. Prostřednictvím slovníku tak lze eliminovat mnoho zbytečných stavů vyhledávacího automatu a zároveň zvýšit jeho spolehlivost. Výjimku představují např. cizí jména s mnoha různými přepisy, které jsou sice přípustné, ale nejsou obsažené ve slovníku.

Relativně menší vliv na efektivitu použité implementace by mohla mít délka transkripce vyhledávaného slova a zvolený jazyk. V našich experimentech se délka testovaných transkripcí pohybovala mezi 4 a 18, přičemž většina z nich spadala do kategorie mezi 6 a 12. Velký počet přepisovacích pravidel však může vést především u jazyků s nejednoznačnou výslovností k exponenciálnímu nárůstu počtu vyhledávaných řetězců. To

Tab. 3: Česko-arabská přepisovací pravidla (symbol ‘◌‘odpovídá vokalizované souhlásce).

česky	arabsky	česky	arabsky	česky	arabsky
á, a, i, áj	ا	s	س	n	ن
á, a, aj, ajá, já, íja, ijá	يا	š, sh, s	ش	h	ه
b, p	ب	s	ص	w, ů, ú	و
t	ت	d	ض	y, i, j, í, ý, íj	ي
th	ث	t	ط	x	كس
j, g, ž, č, dž	ج	z	ظ	a, e, i, o, u, y, ý, í	*(prázdné)*
h, ch	ح	r	ع	č	تش
k, kh, x, ch	خ	gh, g, h, r	غ	un	◌ٌ
d	د	f	ف	a	◌َ
dh, th, d	ذ	q, k	ق	u	◌ُ
r	ر	k	ك	i, y	◌ِ
z	ز	l	ل	a	آ
á, j, ji	ي	m	م	h, t, a, atun, tun	ة
á, í, ji, jí, í, ý	ى				

by mohlo znamenat vážný problém obzvlášť pro dlouhá slova vyhledávaná pomocí automatů Aho-Corasickové bez slovníku.

Dalším významným faktorem je také délka zpracovávaného textu. Pro každé vyhledávané slovo, které uživatel zadá, je totiž třeba vygenerovat nejdřív příslušný vyhledávací automat. V případě FPGA-implementací může celý proces vytvoření a zavedení příslušného vyhledávacího modelu trvat poměrně dlouhou dobu. Zřetelné výhody tohoto typu paralelizace se tak projeví teprve při zpracovávání rozsáhlých textů.

6.1 Dosažené výsledky

Ze získaných výsledků (viz [11] a Tabulky 4–6) je zřejmé, že konstrukce standardních automatů Aho-Corasickové je časově náročnější, pokud je vytváříme se slovníkem. Celý slovník totiž musíme nejprve načíst. Poté je třeba projít slovník vždy, když chceme do vytvářeného automatu zakódovat nové slovo. Výsledné automaty mají na druhou stranu méně stavů i přechodů a přitom dávají přesnější výsledky. Bez slovníku totiž počet generovaných řetězců (a tedy i počet stavů vytvářeného automatu Aho-Corasickové) exponenciálně narůstá vzhledem k délce zadané transkripce a počtu použitých přepisovacích pravidel – viz Obrázky 5 a 7. Především pro dlouhé transkripce tak může systému velice snadno dojít paměť. V takovém případě představují FPGA-implementace vícevrstvých nedeterministických konečných automatů jistě vhodnou i efektivní alternativu.

Pokud tedy není možné použít slovník (buď proto, že není k dispozici, anebo proto že chceme vyhledávat i slova, která v něm nejsou obsažena – např. jména) a máme prohledávat velmi dlouhé texty (100 GB a více), je patrně nejvýhodnější použít FPGA-

Tab. 4: Doba běhu porovnávaných typů automatů na vzorku textu délky 10 MB. V této a následujících dvou tabulkách označují tučně zvýrazněné hodnoty nejlepší dosaženou hodnotu v experimentech se slovníkem, resp. bez slovníku. Kurzívou jsou zapsané výsledky testů, v nichž některé z experimentů nedoběhly (příslušný automat se nepodařilo vytvořit nebo se ho nepodařilo přeložit). '*selhání*' odpovídá případům, kdy žádný z experimentů nedoběhl.

Délka zápisu	Se slovníkem		Bez slovníku			
	AC-A CPU	AC-A FPGA	AC-A CPU	AC-A FPGA	NKA/FPGA	ML NKA/FPGA
Arabština						
≤ 6	**0.28** (±0.01)	20.66 (±2.03)	**0.07** (±0.03)	450.24 (±570.87)	19.33 (±0.35)	70 (±5.62)
7	**0.28** (±0.01)	20.04 (± 0.99)	**0.15** (±0.05)	*11093.06 (±12968.17)*	19.85 (±0.29)	85.24 (±3.67)
8	**0.29** (±0.01)	21.22 (± 4.46)	**0.36** (±0.12)	*6749.65 (±0)*	19.94 (±0.23)	87.86 (±4.27)
9	**0.28** (±0.01)	23.98 (± 5.35)	**1.88** (±0.97)	*3332.01 (±5989.13)*	20.53 (±0.33)	99.51 (±6.36)
10	**0.28** (±0.01)	19.13 (± 0.92)	**9.52** (±6.22)	*selhání*	20.75 (±0.3)	104.43 (±4.46)
11… 14	**0.27** (±0.00)	25.15 (±13.16)	**15.9** (±9.57)	*selhání*	20.8 (±0.73)	117.31 (±7.08)
≥ 15	**0.32** (±0.04)	19.4 (± 1.93)	*selhání*	*selhání*	**21.19** (±1)	147.11 (±8.76)
Němčina						
≤ 6	**2.76** (±0.01)	19.72 (±0.11)	**0.03** (±0)	60.06 (±39.09)	18.28 (±0.11)	50.2 (±1.39)
7	**2.77** (±0.01)	19.65 (±0.29)	**0.09** (±0.03)	*9584.31 (±10887.8)*	18.91 (±0.33)	65.62 (±3.42)
8	**2.77** (±0.02)	19.72 (±0.3)	**0.1** (±0.05)	*4013.86 (±6906.95)*	18.68 (±0.35)	65.7 (±5.11)
9	**2.76** (±0.02)	19.66 (±0.49)	**0.18** (±0.11)	*37.76 (±0)*	18.58 (±0)	66.51 (±7.68)
10	**2.78** (±0.04)	19.79 (±0.7)	**0.51** (±0.43)	*selhání*	19.25 (±0.65)	84.84 (±12.41)
11… 14	**2.78** (±0.08)	19.95 (±0.93)	**3.96** (±5.8)	*selhání*	20.08 (±0.98)	90.01 (±12.74)

implementaci vícevrstvých nedeterministických automatů, která je mnohem rychlejší než standardní CPU-implementace automatů Aho-Corasickové – viz Tabulky 4–6. Při vlastním rozpoznávání vyžaduje standardní implementace automatů Aho-Corasickové zhruba stejnou dobu ať už se slovníkem anebo bez něj. Důvodem je zřejmě fakt, že při zpracování prohledávaných textových souborů zůstává automat Aho-Corasickové většinou v některém z prvních tří stavů. Výsledky prezentované v [10] (viz Tabulky 4–6) a v Tabulce 7 tedy naznačují, že se při fonetickém vyhledávání pomocí standardních automatů Aho-Corasickové vytvářených se slovníkem nevyplatí využívat FPGA-technologie.

Pro arabštinu mají vytvářené automaty typicky větší počet stavů než automaty vytvářené pro německá slova, protože arabská písmena neodpovídají jednoznačně symbolům latinky. Pro oba jazyky (němčinu i arabštinu) však zůstává počet stavů

Tab. 5: Doba běhu zkoumaných typů automatů na vzorku textu délky 1 GB

Délka zápisu	Se slovníkem		Bez slovníku			
	AC-A CPU	AC-A FPGA	AC-A CPU	AC-A FPGA	NKA/FPGA	ML NKA/FPGA
Arabština						
≤ 6	**2.62** (±0.04)	43.68 (±3.42)	**2.43** (±0.04)	477.58 (±573.91)	27.57 (±0.47)	70.46 (±5.66)
7	**2.62** (±0.02)	42.94 (±1.95)	**2.49** (±0.05)	*11128.93 (±12972.93)*	28.1 (±0.46)	85.77 (±3.69)
8	**2.63** (±0.03)	42.59 (±5.21)	**2.72** (±0.12)	*6792.69 (±0)*	28.14 (±0.39)	88.42 (±4.29)
9	**2.63** (±0.03)	46.14 (±6.05)	**4.24** (±0.96)	*3369.77 (±6001.78)*	29.05 (±0.47)	100.11 (±6.39)
10	**2.6** (±0.03)	40.83 (±1.97)	**11.88** (±6.21)	*selhání*	29.92 (±0.52)	105.05 (±4.48)
11… 14	**2.6** (±0.05)	45.51 (±15.4)	**18.2** (±9.57)	*selhání*	30.3 (±0.79)	118 (±7.15)
≥ 15	**2.68** (±0.07)	39.69 (±4.25)	*selhání*	*selhání*	**30.5** (±1.12)	147.98 (±8.8)
Němčina						
≤ 6	**5.09** (±0.01)	37.3 (±0.35)	**2.37** (±0.01)	83.62 (±39.4)	25.91 (±0.14)	50.55 (±1.4)
7	**5.11** (±0.03)	36.95 (±0.85)	**2.4** (±0.04)	*9621.36 (±10894.19)*	26.75 (±0.37)	66.09 (±3.44)
8	**5.14** (±0.03)	37.34 (±0.77)	**2.42** (±0.06)	*4046.61 (±6916)*	26.65 (±0.48)	66.18 (±5.13)
9	**5.11** (±0.05)	37.55 (±1.01)	**2.51** (±0.12)	*61.93 (±0)*	26.35 (±0.21)	67.01 (±7.73)
10	**5.14** (±0.1)	39.26 (±1.67)	**2.88** (±0.45)	*selhání*	27.32 (±0.47)	85.44 (±12.48)
11… 14	**5.11** (±0.21)	38.69 (±3.65)	**6.24** (±5.84)	*selhání*	27.96 (±0.81)	90.67 (±12.78)

Tab. 7: Doba běhu zkoumaných typů automatů na vzorku textu délky 100 GB

Délka zápisu	Se slovníkem		Bez slovníku			
	AC-A CPU	AC-A FPGA	AC-A CPU	AC-A FPGA	NFA/FPGA	ML NFA/FPGA
Arabština						
≤ 6	**236.3** (±2.69)	2345.53 (±160.66)	237.28 (±3.16)	3212.01 (±888)	850.83(±42.26)	**115.72** (±9.45)
7	**235.99** (±2.58)	2332.69 (±108.7)	236.62 (±2.43)	*14715.82 (±13455.41)*	853.75 (±30.25)	**138.94** (±5.39)
8	**233.93** (±2.48)	2180.21 (±142.65)	237.81 (±2.54)	*11096.72* (±0)	848.89 (±27.61)	**144.5** (±6.87)
9	**235.31** (±2.59)	2262.44 (±146.49)	239.48 (±2.27)	*7144.98* (*±7266.63*)	880.93 (±35.4)	**160.17** (±8.84)
10	**233.56** (±2.75)	2211.47 (±119.38)	246.19 (±8.5)	*selhání*	947.21 (±41.74)	**167.11** (±6.18)
11… 14	**235.11** (±5.91)	2081.68 (±332.89)	252.51 (±13.83)	*selhání*	980.56 (±9.62)	**187.62** (±15.36)
≥ 15	**236.87** (±4)	2068.1 (±241.76)	*selhání*	*selhání*	961.44 (±67.02)	**234.92** (±13.67)
Němčina						
≤ 6	**236.5** (±0.92)	1795.32 (±32.35)	234.06 (±0.95)	2440.1 (±89.42)	789.46 (±6.21)	**85.63** (±2.33)
7	**236.12** (±3.41)	1767.67 (±87.48)	235.84 (±3.69)	*13326.08 (±11547.79)*	810.6 (±21.64)	**113.73** (±5.89)
8	**236.17** (±2.74)	1798.47 (±59.42)	237.06 (±2.87)	*7321.55* (*±7824.58*)	823.93 (±17.17)	**113.49** (±7.71)
9	**240.36** (±4.99)	1826.01 (±105.65)	238 (±5.6)	*2479.01* (±0)	803.58 (±21.27)	**117.19** (±13.2)
10	**237.85** (±6.78)	1986.63 (±98.66)	232.92 (±4.08)	*selhání*	834.9 (±30.31)	**145.99** (±19.82)
11… 14	**239.09** (±8.7)	1913.31 (±275.01)	239.03 (±5.32)	*selhání*	816.4 (±15.88)	**156.63** (±17.18)

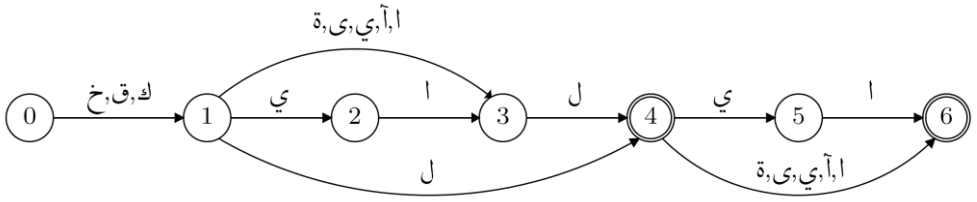
deterministických automatů Aho-Corasickové vytvářených se slovníkem omezený zhruba na stovku. To odpovídá počtu stavů ekvivalentních nedeterministických konečných automatů, který je výrazně menší než pro automaty Aho-Corasickové bez slovníku.

Pokud nelze při generování automatů Aho-Corasickové použít slovník, bude jejich velikost záviset především na délce vyhledávaného vstupního slova a na složitosti použitých přepisovacích pravidel. Vzhledem k extrémně velkému počtu potřebných stavů (viz Obrázky 5 a 7) bylo v některých případech dokonce nemožné příslušný automat Aho-Corasickové sestrojit, přeložit anebo načíst na FPGA-desku – viz Tabulka 7.

Alternativní nedeterministické automaty zůstávají naproti tomu relativně malé dokonce pro poměrně dlouhá slova – německá i arabská. Obrázek 4 ilustruje příklad takového nedeterministického automatu zkonstruovaného pro arabské slovo ´kála´ (قال *[on řekl]*). Počet logických bloků potřebných pro FPGA-implementace nedeterministických automatů navíc roste jen lineárně s počtem jejich stavů – viz Obrázek 6 a 8. FPGA-implementace nedeterministických automatů tedy vyžadují nanejvýš několik málo set logických bloků. Referenční deska, kterou jsme používali při našich experimentech, má k dispozici zhruba 33 000 logických bloků. Na jediné FPGA-desce by tak mohlo paralelně běžet až několik set nedeterministických automatů vytvořených pro různá slova najednou.

Tab. 6: Procentuální podíl automatů, které se nepodařilo vytvořit nebo přeložit

Délka zápisu	AC-A CPU	AC-A FPGA
≤ 6	0 %	0 %
7	0 %	63.2 %
8	0 %	88.2 %
9	0 %	88.9 %
10	15.4 %	100 %
11..14	20 %	100 %
≥ 15	100 %	100 %



Obrázek 4: Nedeterministický konečný automat přijímající arabská slova, která odpovídají fonetickému zápisu 'kála'.

7 Závěr

Fonetické vyhledávání představuje poměrně novou oblast ve vyhledávání informací, která by mohla mnoha uživatelům usnadnit vyhledávání v cizojazyčných textech. V tomto příspěvku jsme proto představili nejpodstatnější implementační otázky, jejichž vhodné řešení by mohlo zvýšit efektivitu fonetického vyhledávání. Při analýze technických prostředků jsme se zaměřili na uživatele, kteří nemusí cílovému jazyku rozumět a nemusí ani znát jeho abecedu. V podstatě ani nepředpokládáme, že by byli schopni správně rozlišovat všechny jeho hlásky. Různé testy [10] potvrdily, že zvolený přístup je co do přesnosti porovnatelný s nejlepšími německými fonetickými algoritmy (Kolínská fonetika a PHONEM) a je dokonce více než čtyřikrát přesnější než arabský Soundex a Phonix. Stejnou strategii lze navíc aplikovat pro libovolný pár alfabetických *jazyků.*

Provedené testy dále ukázaly, že fonetické vyhledávání pomocí automatů Aho-Corasickové je mimořádně efektivní především pro implementace využívající klasické PC. Pro velké indexy anebo kolekce textů je tato technika v průměru čtyřikrát rychlejší než použití regulárních výrazů. Přitom se vyhledají všechny možné zápisy hledaného slova a na rozdíl od např. Soundexových algoritmů je možné vyhledávat i jeho podřetězce. I to přispívá k vyšší přesnosti vyhledávání u obou testovaných jazyků.

Generované nedeterministické konečné automaty ani jejich vícevrstvé varianty naproti tomu nemívají mnoho stavů (okolo 100 i pro delší slova). Vícevrstvé nedeterministické konečné automaty navíc šetří podstatné množství požadovaných synchronizačních kroků. V případě deterministických automatů ale, bohužel, dochází k exponenciálnímu nárůstu počtu stavů. Ten lze sice redukovat pomocí slovníku, to však není možné u speciálních slov, která nejsou obsažena ve slovníku – např. jména.

Experimentálně získané výsledky tedy naznačují, že pokud chceme vyhledávat ve velmi dlouhých textech (100 GB a více) a bez slovníku, je nejlepší volbou použít FPGA-implementace vícevrstvých nedeterministických automatů. Standardní CPU-implementace automatů Aho-Corasickové mají zhruba stejné časové nároky ať už se slovníkem anebo bez něj. Pokud bychom pro vyhledávání používali standardní automat Aho-Corasickové vytvářený se slovníkem, využití FPGA-desek se vůbec nevyplatí.

Výjimkou by snad mohlo být tzv. bit-splitové uspořádání implementovaného automatu Aho-Corasickové [20], které efektivněji využívá jinak omezené zdroje RAM použitých FPGA-desek. Např. při vyhledávání peptidových struktur [5], byla bit-splitová FPGA-implementace automatu Aho-Corasickové zhruba 20-krát rychlejší než jejich klasická implementace spouštěná na standardním PC. K finálnímu rozhodnutí, který model použít, by však mělo dojít i na základě praktických testů provedených pro daný jazyk, přepisovací pravidla a použitý hardware podobným způsobem, který sumarizují Tabulky 4–6.

V rámci dalšího výzkumu se chceme soustředit na problematiku aproximativního prohledávání, které by umožňovalo vyhledat např. i seznam slov podobných zadanému slovu. Následně by tato slova mohla být uživateli nabídnuta jako možné alternativy pro další, tentokrát už exaktní vyhledávání. V případě velkých množství prohledávaných dat by mohlo být výhodné rozdělit zpracovávaná data do vícero bloků a vyhledávat paralelně ve vícero blocích dat najednou. FPGA-implementace fonetického vyhledávání by zřejmě bylo možné využít i při návrhu zařízení pro monitorování a analýzu provozu v sítích. Kromě FPGA-desek ovšem připadají v úvahu i další platformy, např. grafický hardware optimalizovaný pro paralelní blokové operace.

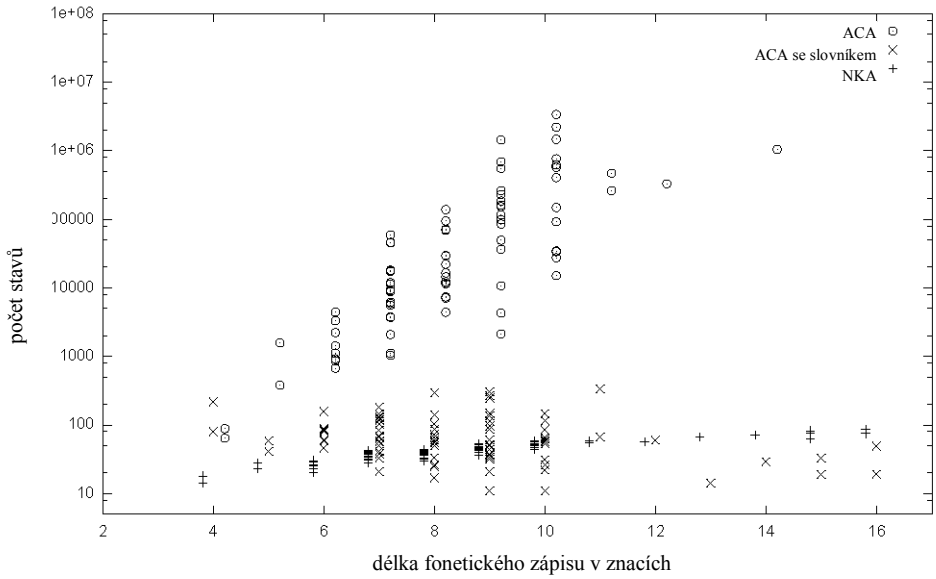
PODĚKOVÁNÍ

Tento výzkum byl podporován Grantovou agenturou České republiky, a to grantem č. P103/10/0783 a grantem č. P202/10/1333.

Poděkování autorů dále patří zejména Prof. Jiřímu Fleissigovi z Oddělení orientálních jazyků Státní jazykové školy v Praze za jeho neutuchající podporu při studiu arabštiny a problematiky strojového zpracování přirozených jazyků.

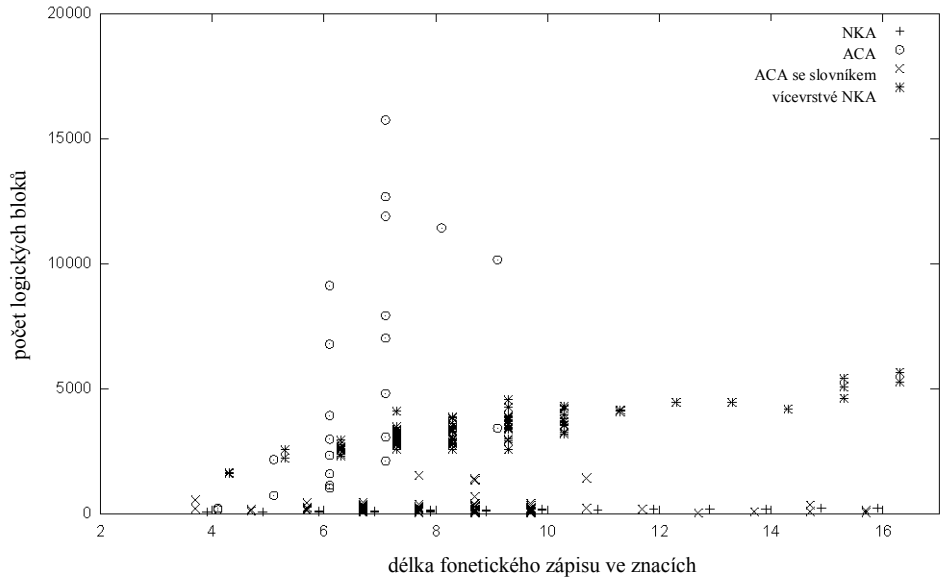
Literatura

1. Aho, A.V., Corasick, M.J.: Efficient String Matching: An Aid to Bibliographic Search. *Communications of the ACM* 18 (1975) 333-340.
2. Al-Shamaa K.: *ArSoundex*. http://www.ar-php.com, 2008.
3. Altera: *PCI Express, PCI, PCIe, PCI-XP, and PCI-SIG Solutions.* [Online], [cit. 2011-8-20], dostupné z <http://www.altera.com/technology/high_speed/protocols/pci_exp/pro-pci_exp.html>.



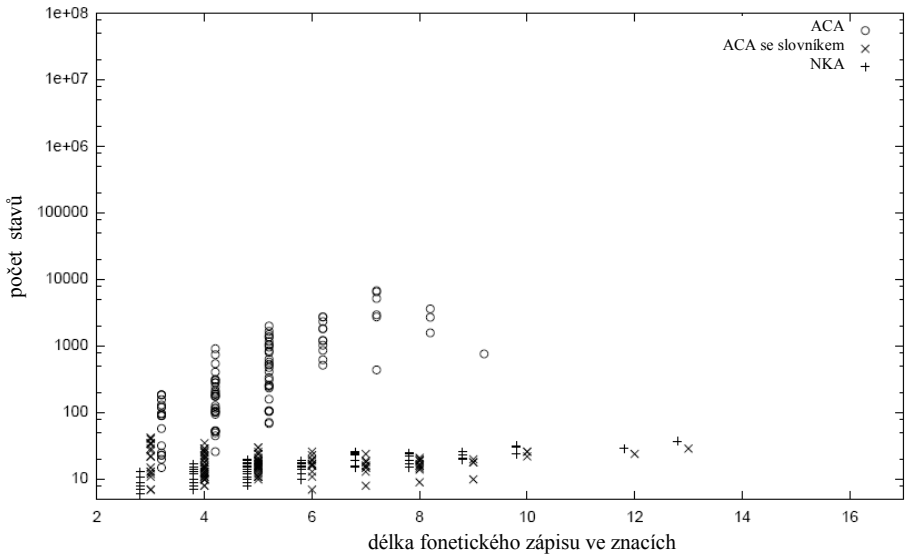
Obrázek 5: Počet stavů zkonstruovaných automatů v závislosti na délce fonetického zápisu v češtině pro arabská slova.

4. Baker, Z.K., Prasanna, V.K.: Time and area efficient pattern matching on FPGAs. In: *Proc. of FPGA'2004*, New York, USA, ACM (2004), 223-232.
5. Dandass, Y. S., Burgess, S. C., Lawrence, M., Bridges, S. M.: Accelerating string set matching in FPGA hardware for bioinformatics research. *BMC Bioinformatics* 9:197 (2008) 11 pp.
6. Dimopoulos, V., Papaefstathiou, I., Pnevmatikatos, D.: A Memory-Effcient Reconfigurable Aho-Corasick FSM Implementation for Intrusion Detection Systems. *Proc. of IC-SAMOS 2007*, IEEE (2007) 186-193.
7. Goldhammer, A.: PCI Express and FPGAs: Why FPGAs are the best platform for building PCI Express endpoint devices. *Xcell Journal* (2007) 14-17.
8. Jung, H.-J., Baker, Z. K., Prasanna, V. K.: Performance of FPGA Implementation of Bit-split Architecture for Intrusion Detection Systems. In: *Proc. of the Reconfigurable Arch. Workshop at IPDPS (RAW '06),* Rhodes Island, Greece, IEEE (2006) 8 pp.
9. Mitra, A., Najjar, W., Bhuyan, L.: Compiling PCRE to FPGA for accelerating Snort IDs. In: *Proc. of ANCS'07*, New York, USA, ACM (2007) 127-136.
10. Mrázová, I., Mráz, F., Petříček, M., Reitermanová, Z.: Phonetic search in foreign texts. In: *Proc. of ANNIE* 2008, ASME Press Series (2008) 533-540.
11. Mrázová, I., Sýkora, O.: When Is Phonetic Search with FPGAs Efficient? In: A. Pulka, T. Golonek, editors: *Proceedings of ICSES'10 (International Conference on Signals and Electronic Systems)*, Gliwice, Poland, IEEE (2010), 359-362.
12. Omondi, A. R., Rajapakse, J. C. (eds): *FPGA Implementations of Neural Networks*. Springer, Berlin (2006).
13. Postel, H.J.: Die Kölner Phonetik: ein Verfahren zur Identi_zierung von Personennamen auf der Grundlage der Gestaltanalyse. *IBM-Nachrichten* 19 (1969) 925-931.

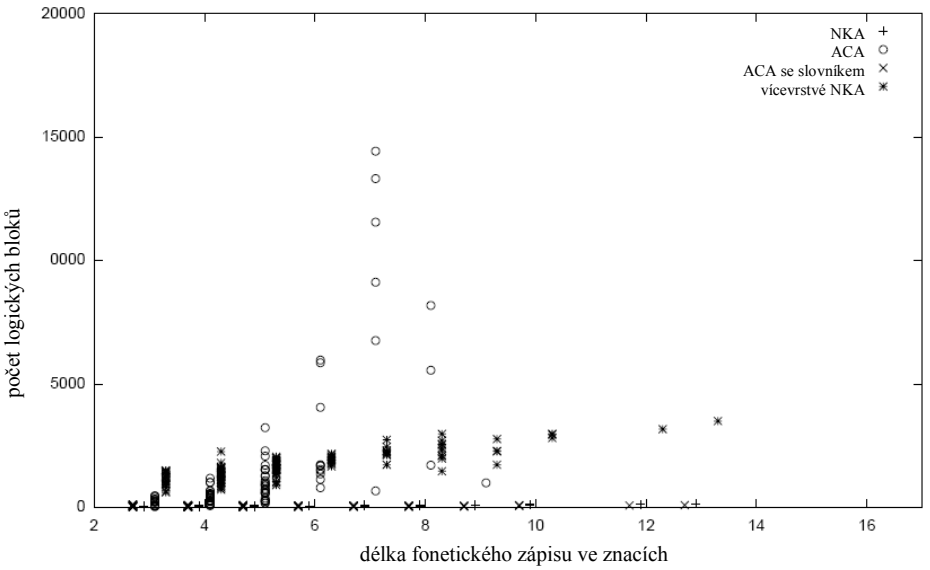


Obrázek 6: Počet logických bloků potřebných pro implementaci automatů na FPGA-desce v závislosti na délce fonetického zápisu arabských slov v češtině.

14. Roesch, M.: Snort - lightweight intrusion detection for networks. In: *Proc. of LISA '99*, Berkeley, USA, USENIX Association (1999) 229-238.
15. Russell, R.: U.S. patent No. 1,261.167. United States Patent Office (1918).
16. Sameer, M.: Myspell arabic spellchecking wordlist. [online], [cit 2007-9-2], dostupé z <ftp://foolab.org/pub/software/arspell/20060329/ar.zip>.
17. Scarpazza, D.P., Villa, O., Petrini, F.: Exact multi-pattern string matching on the cell/b.e. processor. In: *Proc. of CF '08*, New York, USA, ACM (2008) 33-42.
18. Scarpazza, D.P., Villa, O., Petrini, F.: High-speed string searching against large dictionaries on the cell/b.e. processor. In: *Proc. of IPDPS 2008*, IEEE (2008) 1-12.
19. Sipser, M.: *Introduction to the Theory of Computation*. PWS publishing (1997).
20. Tan, L., Sherwood, T.: A high throughput string matching architecture for intrusion detection and prevention. In: *Proc. of ISCA '05*, Washington, USA, IEEE (2005) 112-122.
21. *USB 2.0 and hi-speed USB FAQ*, [online], [cit. 2011-8-20], dostupné z <http://www.everythingusb.com/usb2/faq.htm>.
22. Vierheilig, J.: German spelling wordlist for WinEdt (or compatible) spellcheckers. [online], [cit. 2011-8-20], dostupné z <http://ftp.cvut.cz/tex-archive/systems/win32/winedt/dict/de_neu.zip>.
23. Wikipedia contributors: PCI Express. Wikipedia, The Free Encyclopedia, [online], [cit. 2011-8-20], dostupné z <http://en.wikipedia.org/wiki/PCI_Express>.
24. Wilde, G., Meyer, C.: Doppelgänger gesucht – ein Programm für kontextsensitive phonetische Textumwandlung. *c't Magazin für Computer und Technik* 25 (1999) 252.
25. *Xilinx*. [online], [cit. 2011-8-20], dostupné z <http://www.xilinx.com/products/design_resources/conn_central/solution_kits/pci/index.htm>.



Obrázek 7: Počet stavů zkonstruovaných automatů v závislosti na délce fonetického zápisu německých slov v češtině.



Obrázek 8: Počet logických bloků potřebných pro implementaci automatů na FPGA-desce v závislosti na délce fonetického zápisu v češtině pro německá slova.

Phonetic search in foreign texts - how to find relevant information about "as-sauratu fee misra" without understanding Arabic

Turbulent developments taking place in various areas of our increasingly globalized world pose a great challenge to traditional ways of storing, providing and searching for useful information. Among other requirements e.g. for image and video retrieval, competitive web-mining technologies are also expected to support efficient searching of resources available in foreign languages. On the other hand, it might be quite tricky to formulate the right query for somebody who is not fluent in the respective language – in particular if the chosen language uses a different alphabet (e.g. Arabic) and machine translation is not available.

For many users, a viable option would be to search for relevant information just by typing the words or phrases as he or she *hears* them. Phonetic search represents thus a new area in information retrieval. Its goal is to search texts for all words that have the same pronunciation as the word heard and written by the user. The user is assumed to be a foreigner who uses in general a different alphabet and different transcription rules. Yet when hearing on TV or on the street e.g. the phrase „as-sauratu fee misra," he or she would be able to find information relevant to „the revolution in Egypt."

In this lecture, we will discuss both the traditional phonetic algorithms applicable to the solution of this problem and modern approaches based on finite-state machines that are extremely efficient especially in the case of large text collections. Further, we will assess possible benefits of programmable hardware (FPGA) in phonetic search based on the given hardware, search string, language, availability of a dictionary and length of the searched text.y and length of the searched text.

Základní registry ČR

Petr TILLER

AQUASOFT spol. s r.o.
Rubeška 215/1, 190 00 Praha 9
p.tiller@aquasoft.eu

Abstrakt. Příspěvek představuje systém Základních registrů české republiky. Autor je architektem Informačního systému základních registrů a spoluautorem Globální architektury základních registrů. Bude diskutována funkcionalita, datové procesy a bezpečnost celého systému

Klíčová slova: Základní registry, veřejná správa, IS veřejné správy, datový fond

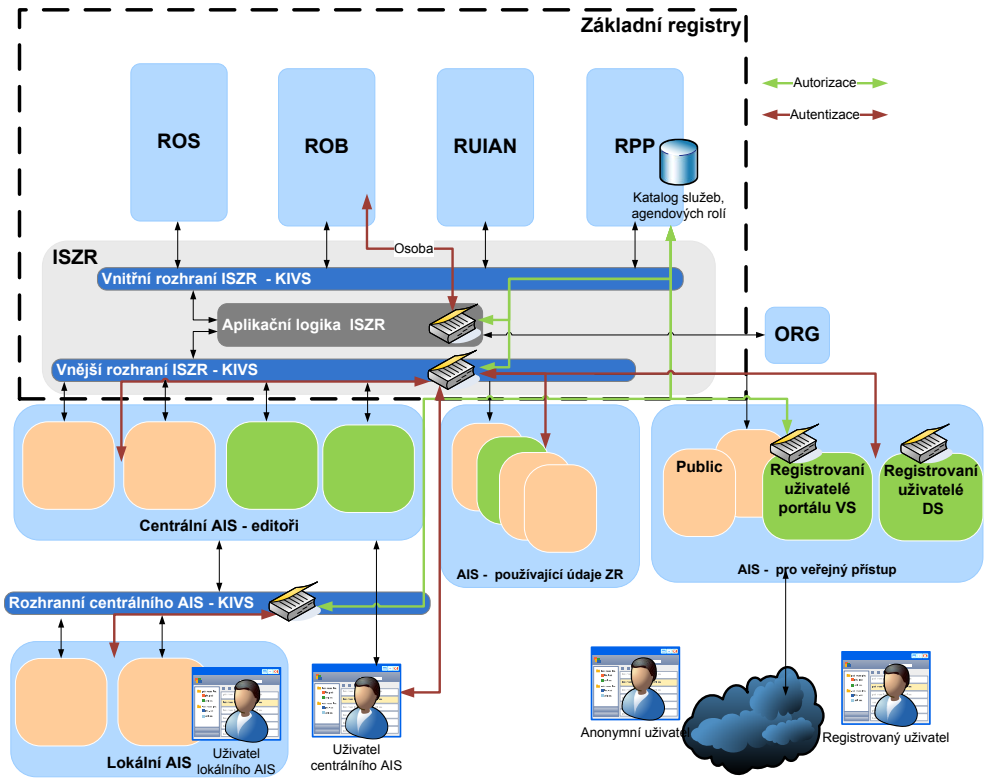
1 Úvod

V současné době neexistuje centrální autorita pro údaje udržované v rámci informačních systémů veřejné správy o občanech, právnických osobách a adresách. Tyto informace jsou udržovány odděleně a nekoordinovaně. Důsledkem tohoto stavu je že

- Občan/právnická osoba musí státním orgánům a a orgánům samosprávy opakovaně dokládat správnost údajů o své osobě/firmě
- V případě změny údajů (např. přestěhování) existuje ohlašovací povinnost
- Údaje v informačních systémech veřejné správy mohou být neaktuální nebo zkreslené a ve výsledku vést k chybnému rozhodnutí nebo ke škodám
- Ve všech informačních systémech veřejné správy a mnoha soukromých je u osob udržováno rodné číslo jako jednoznačný identifikátor, což umožňuje následné hromadné spojování dat

Tento stav se snaží napravit koncepce základních registrů, která označuje skupinu údajů jako "referenční". Tyto údaje vyjmenované podle zákona budou uloženy v základních registrech a poskytovány jednotlivým informačním systémům veřejné správy. Současně je definována role spolupracujícího agendového informačního systému, který poskytuje ostatním agendovým informačním systémům své uložené informace prostřednictvím informačního systému základních registrů.

Základní registry se tak stávají základním kamenem informačních systémů státní a veřejné správy.



Obr.1. Globální architektura základních registrů ČR

2 Údržba dat základních registrů

2.1 Zajištění jednoznačnosti a aktualizace dat

Základním postulátem, který je nutné vzít v úvahu, je, že v základních registrech žádná data nevznikají. Data jsou do základních registrů ukládána zásadně ze zdrojových agendových informačních systémů (AIS), které nazýváme editační.

Za aktuálnost referenčních údajů v základních registrech jsou zodpovědné agendy pracující v roli editorů. Těmto agendám je přikázán dle zákona zápis aktuálního referenčního údaje do registrů bez zbytečného odkladu, nejpozději však do 3 pracovních dnů ode dne, kdy se o vzniku nebo o změně skutečnosti, kterou referenční údaj popisuje, dozví.

Současně může správnost referenčního údaje zpochybnit kdokoliv, kdo je oprávněn daný referenční údaj získat ze základního registru a to v případě, že je schopen prokázat jeho nesprávnost, nebo má jiné oprávněné pochybnosti o jeho správnosti. Referenční údaj však může měnit pouze a jenom editor a ten také zodpovídá za jeho správnost dle zákona. Základní pravidla zpochybnění údaje jsou:

- Referenční údaj může opravit pouze editor

- Proces může spustit kdokoliv, kdo má přístup k danému referenčnímu údaji a to prostřednictvím svého agendového informačního systému, kontaktního místa veřejné správy (CzechPoint), nebo datové schránky.
- Z hlediska Základních registrů dochází k předání procesu editorovi zasláním formuláře zpochybňujícího referenční údaj editorovi (pomocí datové schránky nebo eGON služby) a zasláním potvrzení iniciátorovi procesu
- Pokud je iniciátorem procesu občan a proces byl zahájen prostřednictvím CzechPoint nebo jiného kontaktního místa veřejné správy, pak z hlediska výše popsaného procesu je iniciátorem CzechPoint nebo jiné kontaktní místo veřejné správy a o průběhu a výsledku procesu je informováno právě toto kontaktní místo veřejné správy. Při příjmu žádosti od občana je tedy nutné pořídit kontaktní informace na tohoto občana a o výstupu procesu předat občanovi informaci pomocí standardních postupů veřejné správy.

Editační AIS vytváří a udržuje data v rámci své agendy a pouze část těchto dat ukládá jednoznačným způsobem do základních registrů. Podmnožina ukládaných dat je upravena zákonem. V tomo okamžiku vzniká oprávněná obava, jestli může dojít k nejednoznačnosti dat uložených v základních registrech. Tato nejednoznačnost může teoreticky pocházet z následujících zdrojů:

- Stejný údaj je editován více editory
- Údaj v základních registrech nesouhlasí s údajem ve zdrojovém AIS
- Během ukládání dat nebyla celá transakce správně uzavřena a uložená data jsou nekonzistentní

Jednotlivá rizika jsou ošetřena v rámci návrhu architektury následujícícm způsobem:

2.1.1 Stejný údaj je editován více editory

V současné době existuje situace, kdy jeden a ten samý údaj o subjektu (osoba, firma atd.) je vytvářen v rámci různých agend opakovaně a ne vždy konzistentně (kolikrát například autorovi bylo zkomoleno jméno ani nejde spočítat). Reálně je však vždy právě jedna agenda zodpovědná za vytváření údaje a ostatní agendy by měly tento údaj pouze přejímat.

Tato situace je tedy v rámci základních registrů ošetřena tak, že je zavedena role sekundárního a primárního editora následujícím způsobem:

- **Primární editor** je oprávněn vytvářen nový subjekt/objekt a je následně také oprávněn tento objekt odstranit. Pro každý záznam (datovou větu) existuje vždy právě jeden primární editor
- **Sekundární editor –** je oprávněn ke změně jednoho nebo více údajů u již existujícího objektu. Není oprávněn zakládat nebo odstraňovat objekt. Pro každý údaj existuje nejvýše jeden sekundární editor

Situace je složitější tím, že dle platného právního řádu může být pro různé typy objektů a jejich atributy různý primární i sekundární editor. Nelze tedy stanovit jednoznačný algoritmus, jak určit primárního a sekundárního editora objektu/subjektu nebo údaje. V rámci registru je tedy vždy u každého objektu/subjektu a údaje uloženo, kdo je primárním/sekundárním editorem údaje. Jeden údaj tedy nemůže být editován více editory

a při pokusu o tento postup neoprávněným editorem dojde k chybovému stavu a ke změně nedojde.

Jednotlivý základní registr musí vnitřními procesy zajistit konzistenci dat vzhledem k editorům tohoto registru, a pokud má definovány role primárních a sekundárních editorů, pak musí zajistit izolaci záznamů mezi primárními i sekundárními editory.

2.1.2 Údaj v základních registrech nesouhlasí s údajem ve zdrojovém AIS

K této situaci dochází zdánlivě paradoxně velmi často. Zdrojový AIS pracuje s údaji a na základě ověření správnosti údaje a rozhodnutí o jeho platnosti zapisuje údaj do základních registrů. Základní registry tedy nejsou real-time zrcadlem databází zdrojových systémů, ale obsahují údaje vždy s určitým zpožděním, daným úředním postupem.

Jiná situace však vzniká v okamžiku, kdy údaj v základních registrech nebyl aktualizován, ačkoli zdrojový systém tuto aktualizaci provedl. Tato možnost je eliminována důsledným dodržováním transakčních pravidel při zpracování zápisu dat a zavedením podmínky, že jeden požadavek na editaci může měnit údaje pouze v jediném registru.

Každá změna údajů je navíc zaznamenána v registru práv a povinností s odkazem na zákonné ustanovení a číslo úředního rozhodnutí, které ke změně údaje vedlo.

2.1.3 Během ukládání dat nebyla celá transakce správně uzavřena a uložená data jsou nekonzistentní

Celý systém základních registrů je z vnějšího hlediska jedním datovým zdrojem a je tedy zdánlivě jednoduché požadovat transakční zpracování včetně obecných požadavků ACID:

- **Atomičnost** – Transakce provede buď všechny operace nad databází, nebo žádnou
- **Konzistence** – Izolovaná transakce zachovává konzistenci databáze.
- **Izolovanost** – Dvě a více současně probíhajících transakcí nesmí být vzájemně ovlivněno ani na úrovni mezivýsledků.
- **Trvalost** – Změny v databázi provedené úspěšným dokončením transakce jsou uchovány i při případném výpadku systému

Základní registry jsou vnitřně tvořeny více databázemi a procesy, který zajišťují konzistenci dat mezi jednotlivými registry. V principu platí, že každý jednotlivý základní registr vnitřně splňuje požadavky ACID. Uživatelem základního registru je z tohoto pohledu informační systém základních registrů (ISZR). Pokud tedy ISZR při provádění eGON služby použije službu jednotlivého základního registru publikovanou na vnitřním rozhraní, pak provedení této služby musí zcela splňovat požadavky ACID.

Celková transakce je dále opět rozdělena na příjem a zpracování požadavku. Oproti specifikaci ACID může být editorem navíc požadováno serializované zpracování požadavků. ISZR pak musí zajistit, že do jednotlivých základních registrů jsou tyto požadavky předávány ve stejném pořadí, v jakém byly přijaty a to až po vyřízení předchozího požadavku jednotlivým základním registrem. V případě neúspěšně provedené transakce zápisu jsou všechny následující požadavky v dané sekvenci označeny za neúspěšné.

Požadavek serializace může být zadán pouze v rámci jednoho editora. Jeden editor tedy nemůže vyžadovat řazení požadavků v závislosti na požadavcích druhého editora.

2.2 Zajištění aktualizace dat jednotlivých agendových systémů

Agendové informační systémy pracují typicky s údaji, vedenými v rámci agendového informačního systému. Tyto údaje se skládají jak z údajů, které se v AIS nevytváří , tak z údajů, které se v rámci AIS vytváří.

Údaje, které se v AIS nevytváří mohou být zároveň referenčními údaji v některém základním registru (nikoliv nutně v plném rozsahu). Cílem je tedy takový stav, kdy údaje, které jsou v základních registrech uloženy jako referenční, jsou aktualizovány v AIS tak, aby osoba pracující s AIS mohla pracovat v dobré víře, že stav údaje v AIS odpovídá referenčnímu údaji v základních registrech.

Není prakticky proveditelná situace, kdy jsou všechny údaje vždy v reálném čase přejímány ze základních registrů. V současné době je aktualizace údajů v jednotlivých AIS velice zdlouhavá a většinou bez pevných procesů. Cílem základních registrů je náprava tohoto stavu, nikoliv snaha o neustálé odebírání údajů z jednoho místa.

Pro aktualizaci údajů v AIS budou k dispozici následující procesy:

- **notifikační proces** – zajišťuje pravidelnou informovanost AIS o tom, u jakých subjektů a jejich údajů v základních registrech došlo ke změně
- **čtení v reálném čase** - při dotazu odevzdá aktuální hodnotu referenčních údajů
- **update** – v případě veřejných informací je poskytován pravidelný výdej změnových vět jednotlivých registrů

Cílem těchto procesů je zajištění souladu údajů vedených v AIS s odpovídajícími referenčními údaji v základních registrech.

Ideálním stavem je, když AIS provádí pravidelnou aktualizaci vedených údajů pomocí notifikačního systému, popsaného v následující kapitole. Notifikační systém zaručuje, že údaje v AIS odpovídají stavu referenčních údajů z předchozího dne.

Update pomocí změnových vět je prováděn z veřejných údajů uložených v základních registrech (celý obsah registru RUIAN, části registrů ROS a RPP). Tento mechanismus vždy jednou měsíčně poskytne celý veřejný obsah a dále pravidelně denně poskytuje změnové věty za uplynulý den. Takto vydané údaje jsou již dále informační a jejich výdej ze základních registrů není logován.

2.3 Notifikační systém

Jak bylo popsáno výše, tento systém je základním nástrojem pro aktualizaci údajů v jednotlivých agendových informačních systémech a poskytuje **referenční** údaje ze základních registrů. V rámci notifikačního systému je vždy jednou denně vytvářen seznam objektů/subjektů, u kterých za uplynulý den došlo ke změně:

- ROB – seznam AIFO
- ROS - seznam ICO
- RUIAN – seznam identifikátorů adresního místa

Tyto seznamy jsou pak poskytnuty jednotlivým agendovým informačním systémům, které provádí aktualizaci těch objektů/subjektů se kterými pracují v rozsahu údajů, se kterými pracují. Současně jsou při vzniku subjektu/objektu informovány ty agendové systémy, u kterých existuje ohlašovací povinnost. Tato povinnost je doposud určena všem obyvatelům (například informování zdravotní pojišťovny o narození člověka).

Notifikační systém lze samozřejmě inciovat ad-hoc pro jakýkoliv agendový systém. Celý systém je nastaven tak, aby při dodržení tohoto postupu získal uživatel jistotu, že pracuje s údaji, které souhlasí s referenčními údaji v základních registrech při minimálních požadavcích na komunikaci se základními registry.

3 Bezpečnost dat v základních registrech

Vzhledem k tomu, že v základních registrech jsou uloženy základní informace o všech občanech a právnických osobách v České republice je nutné, aby tato data byla uložena bezpečně a byla zajištěna maximální kontrola užití těchto dat. Konkretní architektura a realizace bezpečnostích postupů základních registrů není veřejná s tím, že bezpečnostní perimetr na vnějším rozhraní základních registrů je zabezpečen veřejně známým provozním předpisem. Vnitřní mechanismy kontroly a ochrany však nejsou zveřejněny. Tímto postupem je ztížena činnost potencionálního útočníka, který neví, jaké kontorly jsou prováděny uvnitř systému a má tedy ztíženu případnou penetraci.

Veřejné kontrole bezpečnosti bude tedy podrobováno vnější rozhraní základních registrů (eGON rozhraní).

Pro zajištění bezpečnosti dat je nutné uvažovat následující aspekty:

- Zajištění bezpečnosti předávaných dat
- Zajištění bezpečnosti ukládaných dat
- Ochrana auditních postupů proti zneužití

Důsledkem porušení bezpečnosti může být situace, kdy jsou zveřejněny nebo odcizeny osobní údaje osob, nebo kdy je neoprávněně nahlíženo do historie práce jednotlivých orgánů veřejné moci s daty občana. Tato nebezpečí lze shrnout do základních rizik:

- Neoprávněný přístup k datům
- Podvržení dat
- Sledování historie („Velký bratr“)

Zajištění neoprávněného přístupu k datům a zamezení neoprávněné manipulace s daty jsou standardní úlohy při správě dat a ani v případě základních registrů nejsou použity žádné převratné metody. Tyto standardní úlohy jsou řešeny standardními postupy:

- **Ochrana komunikace –** využívání kryptované komunikace SSL s identifikací klienta. Klient se musí při komunikaci identifikovat vydaným klientským certifikátem a kanál je tedy založen na vzájemné důvěře mezi serverem a klientem s identifikací serveru i klienta

- **Ochrana uložených dat –** využívání kryptování uložených dat a kontroly přístupu k datům jak na aplikační, tak na systémové úrovni. Obecně musí být oddělena schopnost správy dat od schopnosti data číst.

Speciálním problémem je zamezení „Velkého bratra“. Tato komplexní podmínka požaduje, aby nikdo za žádných okolností nemohl svévolně získávat data o občanech a právnických osobách a nemohl ani sledovat práci jednotlivých agend s těmito daty. Z tohoto hlediska je zvláště citlivým bodem ukládání logů o jednotlivých transakcích, protože právě z těchto logů by bylo možné potencionálně rekonstruovat citlivé informace.

Ochrana logů je prováděna na základě využívání násobné identity (ZIFO/AIFO – popsáno v následující kapitole) a rozdělení logů dle následujících principů:

- Pro rekonstrukci logu je vždy nutná spolupráce minimálně třech částí základních registrů – převodník ORG, ISZR a cílový registr
- V žádné části základních registrů nikdy není uložen log obsahující kompletní transakce včetně osobních údajů a identifikace osoby
- Výdej informací z logů je vždy evidován a není umožněn bezdůvodně

4 Ochrana identity ZIFO/AIFO

Základním nástrojem ochrany dat je koncepce Základního identifikátoru fyzické osoby (ZIFO) a Agendového identifikátoru fyzické osoby (AIFO). Tento koncept násobné identity je vybudován následovně:

- Převodník ORG, který je spravován Úřadem pro ochranu osobních údajů přiděluje pro každou osobu ZIFO. ZIFO nikdy neopustí databázi ORG. Z druhé strany ORG “neví” komu bylo ZIFO přiděleno – v rámci ORG nejsou uloženy žádné osobní údaje.
- Existuje jednoznačný převod mezi ZIFO a AIFO pro cílový agendový systém. Tento převod je jednocestný, není tedy možné na základě znalosti AIFO pro danou agendu odvodit ZIFO
- Pro každou agendu je využíváno unikátní AIFO

V tomto konceptu je zaručeno, že není možné spojovat data mezi jednotlivými agendami na základě znalosti jednoho univerzálního klíče (jakým je dnes rodné číslo). Pro vzájemné provázání dat mezi jednotlivými agendami tedy vždy bude nutný překlad AIFO zdrojové a cílové agendy, což zajišťuje pouze převodník ORG.

5 Závěr

Zavedení základních registrů České republiky má význam zvláště v oblasti definice referenčního údaje (a jeho právních důsledků) a zavední násobné identity občana v agendách veřejné správy.

Zavedením existence refererenčního údaje dochází poprvé v historii České republiky k situaci, kdy občan nebo fyzická osoba prokazují jakoukoliv skutečnost, obsaženou v referenčních údajích, vůči orgánům veřejné moci pouze jednou a dále je tento údaj disttribuován všem relevantním agendám. Současně bude možné dále pokračovat k omezování ohlašovacích povinností občana.

Násobná identita chrání soukromí občana a soustředěním všech aktivit do jednoho kontrolovaného místa dává občanovi podstatně vyšší záruky ochrany jeho osobích údajů, než dnešní stav. Občan poprvé v hostorii dostane do ruky nástroj, kterým se může dovědět, které agendy státní a veřejné moci pracují s jeho údaji.

Předložený článek pak předkládá popis postupů pro zajištění konzistence a ochrany těchto údajů.

Ucelený pohled na správu služeb IT.
Projekt centrálního monitoringu IT v České pojišťovně.

Ing. Aleš SMETANA[1], Jindřich ŠTUMPF[2]

[1] ***GC System a.s.***
Na Strži 3/342, 140 00 Praha
asmetana@gcsystem.cz
[2] ***GALEOS a.s.***
Michelská 300/60, 140 00 Praha
jindrich.stumpf@galeos.cz

Abstrakt. Příspěvek popisuje principy správy služeb IT (IT Service Management) řešené portfoliem produktů IBM a Progress Software. Jde o ucelenou sadu nástrojů na řízení služeb a jejich kvality doplněnou o monitoring a vizualizaci SOA. Text příspěvku je rozdělen na dvě části: v teoretické definuje obecné pojmy a logické vrstvy z oblasti ICT monitoringu a principy řízení poskytování IT služeb vycházející z ITIL a governance SOA.

V druhé části autoři představí implementační projekt centrálního monitoringu IT v České pojišťovně, který patří mezi nejrozsáhlejší projekty svého druhu v ČR a Evropě. Je vyjmenována škála užitých SW technologií, popsána komplexita heterogenního prostředí zákazníka a uvedeny vybrané zajímavé statistiky a zjištění z tohoto projektu.

Klíčová slova: IT Service Management, ITIL, SOA, SOA governance, SOA repozitory, UDDI registry, Service Desk, Change and Configuration Management Database, Tivoli, Actional

1 Základní vlastnosti a položky IT Service Managementu

Definice základních pojmů ITILV3

Služba (Service)	Prostředek dodávání hodnoty zákazníkovi tím, že zprostředkovává výstupy, jichž chce zákazník dosáhnout, aniž by vlastnil specifické náklady a rizika.
IT Služba (IT Service)	Služba poskytovaná jednomu nebo více zákazníkům poskytovatelem služeb IT. IT služba je založena na využití informační technologie a podporuje podnikové procesy zákazníka.
Správa služeb (Service Management)	Množina specifických organizačních schopností pro dodávání hodnoty zákazníkům ve formě služeb.
Databáze konfigurací a změn (CMDB)	Databáze využívaná pro správu konfiguračních záznamů během jejich životního cyklu. Systém správy konfigurací obsahuje jednu nebo více konfiguračních databází. V každé konfigurační databázi jsou zaznamenány atributy konfiguračních položek a vazby s dalšími konfiguračními položkami. [ITSMF1]

Konfigurační položka	Jakákoliv komponenta, která by měla být spravována za účelem dodávky služby IT. Informace o všech konfiguračních položkách jsou zaznamenány v konfiguračním záznamu v systému správy konfigurací (CMS) a jsou udržovány během jejich životního cyklu Správou konfigurací. Konfigurační položky jsou řízeny Správou změn. Konfigurační položky typicky zahrnují hardware, software, stavby, lidi a formální dokumentaci, jako dokumentaci procesů a SLA. [ITSMF1]
Service Desk	Jediné kontaktní místo mezi poskytovatelem služeb a uživateli. Typický Service Desk spravuje incidenty a požadavky na službu a obstarává komunikaci s uživateli. [ITSMF1]
Střední doba obnovy (MTTR)	Středni doba opravy je průměrný čas potřebný pro opravu konfigurační položky nebo služby IT po selhání. MTTR je měřena od okamžiku výpadku konfigurační položky nebo služby IT do okamžiku, kdy jsou konfigurační položka nebo služba zcela obnoveny a poskytují svoji plnou funkčnost. MTTR nezahrnuje čas obnovy. MTTR je někdy nesprávně používaná jako střední doba obnovy služby. [ITSMF1]

2 Konsolidace nástrojů pro IT Service Management – pohled zespoda vzhůru

Služby IT jsou podstatnými, strategickými a organizačními aktivy. Proto organizace do nich musí investovat odpovídající kapacity zdrojů do podpory, dodávky a správy těchto kritických služeb IT a systémů IT, které je podporují [ITILV3].

“Poskytovatelé IT služeb si již nemohou dovolit zaměřovat se na technologie a jejich vzájemné uspořádání; primárním cílem je kvalita poskytovaných služeb a budování vztahu se zákazníkem. [BONVAN]

Výše uvedené vystihuje fakt, že pro správné a efektivní řízení výkonnosti organizace s podporou ICT je potřebné používat portfolio nástrojů, které vnímá heterogenní prostředí ICT jako celek – umí ICT rozpoznávat, analyzovat, učit se a následně vyhodnocovat a navíc identifikovat dopad do samotných obchodních procesů a činností organizací. Koncept celistvého pohledu na řízení IT služeb (ITSM) začíná u myšlenky “co neumíš měřit, neumíš řídit” a končí u efektivity služeb ICT jako platformy významně přispívající ke spokojenosti a úspěšnosti zákazníka IT služeb a jeho zákazníků. Je to právě dostupnost a efektivita byznys služeb, z které vycházejí potřeby monitorovat prvky ICT.

Pro ucelené řízení ITSM má IBM připravené portfolio produktů. Česká pojišťovna se rozhodla pro jeho využití v kombinaci s modulem Progress Actional pro monitoring a vizualizaci SOA. IBM portfolio se skládá z pěti základních stavebních kamenů, které v dalším textu dále rozvedeme. Tyto stavební kameny jsou:

1. Zjišťování a řízení závislostí (Dependencies)
2. Sběr, monitoring a správa událostí (Monitoring & Events)
3. Monitoring uživatelských aplikací (User Experience)
4. Řízení metrik byznys služeb (Business Metrics)
5. Agregovaný pohled na řízení ICT (Views)

2.1 ICT – klíčový majetek organizace

Předtím, než budeme dále diskutovat jednotlivé stavební prvky IBM portfolia pro ITSM, je nutné začít u samotného fundamentu „proč se organizace zabývají ITSM a z čeho vychází veškeré snažení týkající se řízení dostupnosti a efektivity ICT“, jeho významu pro obchodní služby organizací atd. Jde především o fakt, že všechny části ICT (konkrétně HW, SW i SW licence) jsou v prvé řadě aktiva organizace, na která je vynakládáno velké množství finančních zdrojů, které je nutné, v rámci požadované návratnosti, získat zpět obchodními činnostmi.

Nejde tedy o pouhé řízení ICT, jde o komplexní proces řízení aktiv (Asset Management), kdy každé aktivum organizace má svůj životní cyklus, jehož významnou částí je právě ITSM.

Životní cyklus aktiva je proto možné zobecnit do několika fází.

1. Plánování – první fáze, ve které organizace přemýšlejí o samotné potřebě a přínosech pořízení aktiva a jeho nasazení pro podporu obchodních činností společně se všemi požadavky (finančními i technickými) na jeho provoz;
2. Ohodnocení a návrh – tato fáze zahrnuje samotný návrh jakým způsobem bude aktivum nasazeno do existujícího nebo plánovaného funkčního celku, kalkulaci přínosů a rizik a návrh následné realizace;
3. Pořízení – jde o fázi realizace nákupu, včetně uzavření smluv o pořízení i podpoře aktiv a nasazení do prostředí organizace;
4. Provoz a sledování, údržba a správa změn – fáze, kdy aktivum produkuje přidanou hodnotu, pro kterou bylo pořízeno, v průběhu které je nutné také sledovat její funkčnost a vazby vůči ostatním aktivům, provádět údržbu a změny vyplývající z obchodních činností organizace a jejich kontinuálního rozvoje;
5. Vyřazení a recyklace – ukončení životního cyklu tak, aby byl co nejmenší dopad na kontinuitu byznys služby, včetně životního prostředí, ve kterém organizace existuje apod.

Proto je důležité, aby informace a vztah mezi řízením z pohledu ITSM bylo propojeno s pohledem finančním a pevně spojeno s ekonomickými cíli organizace.

2.2 Zjišťování a řízení závislostí (Dependencies)

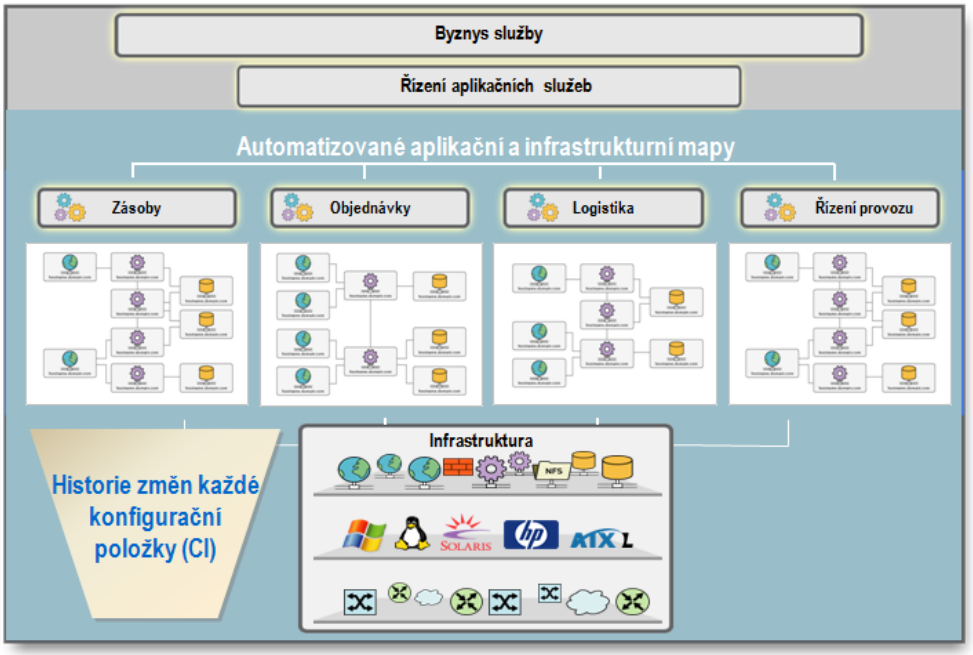
V předchozí kapitole bylo uvedeno, že životní cyklus aktiv organizace se rozděluje do několika fází. ITSM jistě v menší či větší míře souvisí se všemi z těchto fází, nejvýznamněji ale pravděpodobně s provozem a sledováním, údržbou a správou změn.

Prvním krokem pro plnění tohoto účelu je nutné evidovat požadované informace – a to pro všechny části ICT – síťovou infrastrukturou počínaje a uživatelskými aplikacemi konče. Pro snazší uchopení a identifikaci se proto zavádí pojem „konfigurační položka“ (CI z anglického originál Configuration Item), která představuje základní stavební jednotku systému konfiguračního managementu a je dle [ITSMF1] jakoukoli komponentou, která by měla být spravována za účelem dodávky služby IT

Díky takovémuto systému jedinečné identifikace je možné k jednotlivým konfiguračním položkám přiřazovat nejenom základní informace, ale také evidovat další vlastnosti a závislosti vytvořené mezi ostatními konfiguračními položkami.

V rámci zjišťování a řízení závislostí prvním kroku je tedy cílem zajistit odpovědi na otázky:

- Jaké konfigurační položky máme pořízené, včetně evidence jejich základních atributů a informací o konfiguraci, včetně informací jaké mají vazby s dalšími konfiguračními položkami, tzn. jakým způsobem se ovlivňují.
- Kde jsou konfigurační položky umístěné, kdo je za jejich provoz a správu odpovědný, jakým způsobem a s jakými znalostmi tuto správu vykonává.
- Jakým způsobem se provádí nad těmito prvky změny a jaké tyto změny mají dopady na ostatní konfigurační položky, včetně byznys služeb na nejvyšší úrovni hierarchie. Jaká rizika znamenají změny do prostředí, včetně nároků (finančních i organizačních) na deployment nebo rollback nových aplikací.
- Jak víme které byznys služby jsou provozovány nad jakými prvky a zda tyto prvky splňují požadavky na realizaci těchto byznys služeb.
- Jak je možné zajistit a garantovat, smlouvami o dostupnosti služeb např. OLA nebo SLA, dostupnost byznys služeb a řídit rizika s tím spojená.



Obr. 1: Přehled oblastí závislostí mezi jednotlivými prvky systémů IT

Proto, aby mohla být organizace schopna na tyto body reagovat, nabízí IBM nástroje na automatické vyhledávání (discovery) jednotlivých konfiguračních položek a procesy nad jednotlivými konfiguračními položkami, ať už jde o konfigurační databázi (CMDB) nebo procesy Change a Release Managementu.

Průzkumy organizací po celém světě ukazují, že zhruba pouze 40% IT infrastruktury je discoverováno a plně porozuměno – např. ve smyslu jakou mají prvky mezi sebou souvislost, což je významná motivace pro zlepšení. IBM proto nabízí řešení, které provádí discovery IT prvků v heterogenním prostředí do tří úrovní – discovery prvků v rámci sítě, discovery konfigurací IT prvků a aplikačních závislostí, discovery konfigurací samotných

aplikací, včetně auditování a reportování změn konfiguračních položek a compliance např. s interními bezpečnostními pravidly organizace.

Jako zdroj pro ukládání těchto informací a nástroj nabízející další přidanou hodnotu je nabízen nástroj Tivoli CCMDB, systém konfigurační databáze. Ten krom evidence konfiguračních položek, podpořené možnostmi implementací nativních procesů organizace jakékoli velikosti, nabízí také přehledný nástroj pro zobrazení závislostí mezi konfiguračními položkami a analýzu dopadu (včetně pohledu byznys služeb, nikoli tedy pouze IT), auditování změn a nativní napojení na procesy Change a Release Managementu, které využívají uložených informací.

Propojením takového nástroje pro discovery s konfigurační databází vzniká opravdový „single source of truth“ (přeloženo jako jediný zdroj pravdivých informací pro informace o konfiguračních položkách), který vytváří silný, komplexní a plně automatizovaný základ, který je použitelný pro rozhodování na všech úrovních řízení IT a integrovatelný s dalšími vrstvami popisovanými v dalších kapitolách.

2.3 Sběr, monitoring a správa událostí (Monitoring & Events) a Monitoring uživatelských aplikací (User Experience)

Po kroku identifikace a evidence jednotlivých prvků (konfiguračních položek) v daném prostředí je nutné tyto prvky monitorovat a nad informacemi poskytnutými z monitoringu provádět efektivní a inteligentní rozhodování.

Pro oblast monitoringu je klíčovou metrikou tzv. Střední doba do obnovy (anglicky Mean Time To Recovery - MTTR), tedy dle [WIKI1] průměrná doba, po které se měřený subjekt navrátí do běžného stavu. Tento úsek dále rozdělujeme na dobu identifikace problému, notifikaci (operátorům Service Desku, resp. vlastníkovi), diagnostiku a izolaci problému a poté zafixování a zprovoznění. Díky tomu, že nástroje z portfolia IBM jsou vzájemně integrované a mají jednotnou základnu, přinášejí zákazníkovi výrazné zjednodušení pracnosti a tudíž snížení MTTR.

Klíčem ke zvýšené efektivitě je jednoznačně snaha poskytovat uživateli proaktivní přístup k monitoringu a vyhodnocení daných událostí – propojením současných a historických dat z monitoringu, obohacování událostí z více zdrojů, vytvářením trendů, využitím fixních nebo dynamických prahových hodnot (tzn. thresholdy) nebo jednotnou integrační platformou, jejímž úkolem je konsolidace, korelace a automatizace nasbíraných událostí.

IBM ve svém portfoliu pro monitoring (produkty IBM Tivoli Monitoring) nabízí možnost sledovat rozličné prvky využité v prostředí zákazníka, fyzickými a virtuálními HW prvky počínaje (včetně síťové infrastruktury), přes monitoring middleware nebo aplikační vrstvu a bezpečnostními oblastmi konče. Všechny monitorované informace jsou poté poskytnuty na jednotnou komunikační vrstvu a také ukládány do jednotného datového skladu (anglicky data warehouse). Monitorovat lze agentním nebo bezagentním způsobem, v závislosti na prioritách a potřebách zákazníka. Díky datovému skladu lze poté s uloženými daty dále pracovat a využívat je v rámci analýzy problému nebo proaktivnímu plánování kapacit HW.

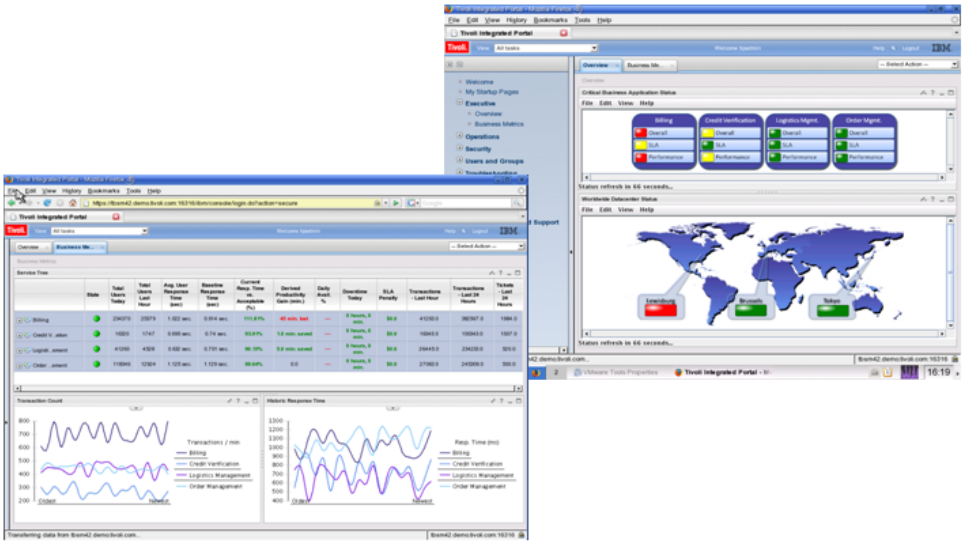
Nadstavbou nad monitoringem je jednotná vrstva Tivoli Netcool Omnibus, který funguje jako jediný bod, do kterého jsou směrovány všechny události. V této částí je rovněž možné napojení některých stávajících systémů. Tento centrální bod je velmi důležitý pro udržení jednotného dohledu nad všemi částmi prostředí. Korelace událostí pomocí progresivních algoritmů je velmi důležitou částí, aby ICT provoz nebyl zahlcen chybovými událostmi následků, ale omezeným počtem chybových událostí, které ukazují na skutečnou příčinu problému. Tímto pokročilým způsobem se mnohonásobně redukuje počet událostí.

2.4 Řízení metrik byznys služeb (Business Metrics)

Informace z komponent portfolia IBM zmíněných v předchozích dvou kapitolách mohou být vstupem do vrstvy, která vizualizuje a transformuje ICT na obchodní procesy. Tivoli Business Service Manager (TBSM) vizualizuje dopad aktuálního stavu provozu ICT na klíčové procesy zákazníka, které zajišťují příjem finančních prostředků (např. klíčové aplikace pro analýzu klientů, podnikový informační systém, prodejní portál, apod.).

Systém tak poskytuje odpovědi na otázky typu:

- Jakým způsobem propojím byznys kontext s provozem ICT, např. zdroji nebo transakcemi byznys aplikací?
- Na základě jakých pravidel mohu prioritizovat řešení incidentů v případě že se vyskytnou ve stejnou dobu?
- Jakým způsobem mohu kalkulovat smysluplné KPI pro byznys?



Obr. 2: Custom dashboardy pro řízení byznys služeb

2.5 Agregovaný pohled (Views)

Poslední vrstva reprezentuje jednotný přístup ke správě celého řešení a jednotnému reportingu z libovolné komponenty řešení. Tato část je velmi důležitá pro efektivitu provozu samotného řešení.

Na základě potřeb zákazníka je možné z celkového pohledu získat detailní – tedy od byznys služby až po ICT prvek nebo naopak.

3 ITSM na míru společnosti každé velikosti

Portfolio produktů a metodických principů pro ITSM je závislé na obchodních potřebách zákazníka, které jsou zpravidla přímo úměrné velikosti dané organizace. Současně jeho pořízení závisí na způsobu a schopnostech jeho využití. Uvažujme tři velikostní profily organizací.

3.1 Velký dodavatel služeb

Velký dodavatel služeb podporuje komplexní heterogenní prostředí, poskytuje služby postavené na uznávaných principech řízení IT (ITIL, COBIT) – např. poskytovatel ICT služeb dodávající služby interním nebo externím zákazníkům (operátoři služeb, velké finanční a státní instituce).

Používá nástroje ze všech šesti výše popsaných oblastí.

Benefity:

- Real-time vizualizace a kalkulace SLA/OLA.
- Konsolidace a integrace ITIL procesů, automatizace ICT prostředí.
- Monitoring a vizualizace SOA.
- Konsolidace aplikací monitoringu

3.2 Středně velký dodavatel služeb

Středně velký dodavatel služeb chápe ICT jako pomocný nástroj pro úspěch byznysu svého zákazníka, implementuje části uznávaných principů řízení IT – např. velký nebo středně velký výrobní obchodní podnik.

Používá: TADDM, TSRM, CMDB.

Benefity:

- Konsolidace a integrace vybraných ITIL procesů, automatizace.
- Monitoring a vizualizace SOA.
- Konsolidace aplikací monitoringu.

3.3 Malý dodavatel služeb

Malý dodavatel služeb chápe ICT jako pomocný nástroj pro úspěch byznysu svého zákazníka, vytváří interní principy dle osvědčených nejlepších postupů (best-practices). Typicky používá open-source produkty.

Nástroje (včetně specifikace): ITM

Benefity:

- Jednotné rozhraní pro uživatele (Service Desk).
- Konsolidace aplikací monitoringu

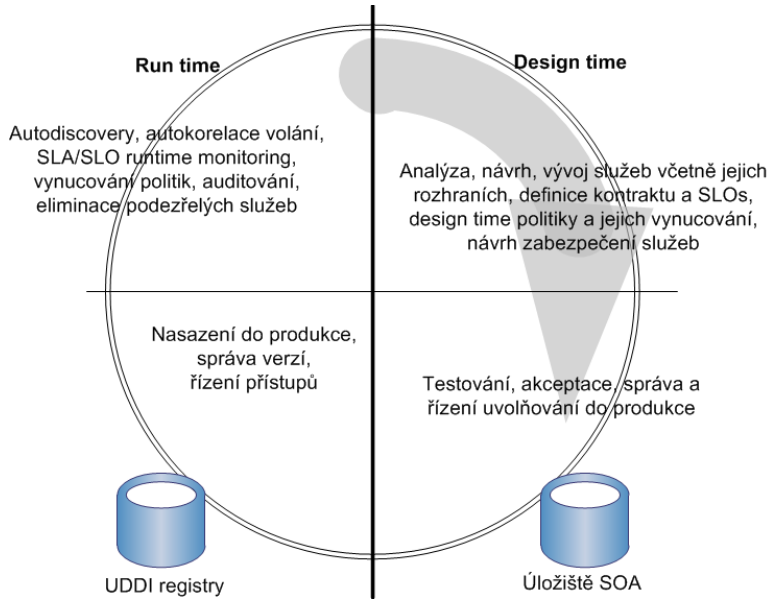
4 Jaké činnosti by měla řešit governance SOA?

Softwarové služby jako stavební kameny SOA jsou nedílnou součástí ITSM. Pod pojmem (softwarová) služba budeme v kontextu SOA rozumět smluvní interface do softwarové logiky. Pod pojmem governance SOA budeme mít na mysli činnosti zajišťující správu, řízení, vizualizaci, reporting a systém varování (alerting) servisně orientované architektury.

Na základě vlastních zkušeností rozdělujeme governance SOA na dvě základní fáze a sedm níže uvedených činností:

- Předprodukční fáze (design time)
    - Vytvoření a správa datového modelu SOA
    - Správa kontraktů a předprodukčních politik
    - Správa a poskytování informací
    - UDDI registry
- Produkční fáze (run time)

- Definice a správa přístupových bodů
- Autodiscovery, autokorelace a vizualizace síťové mapy volání služeb
- Vyhodnocování měření a vynucování provozních politik SLA



Obr. 3: Nástin rozdělení fází governance SOA.

4.1 Předprodukční fáze

Vytvoření a správa datového modelu SOA

To co platí pro běžné aplikace, zde platí dvojnásob: dobře navržený datový model je klíčovým stavebním kamenem celé servisní architektury. Musí umožňovat uchovávat jak všechna data, tak i metadata o službách. Kromě obvyklých údajů, jako kdo je vlastníkem služby, jakými procesy (životním cyklem) musí procházet, kde má být služba umístěna, jakou má úroveň zabezpečení, jaký soubor pravidel je na ní kde aplikován apod., je nutné uchovávat i informace o vzájemných vztazích a závislostech služby a jejích konzumentů. Tyto informace slouží pro analýzu základních příčin problémů, pro kapacitní plánování, pro verzování služby a plánování údržby.

Správa kontraktů a předprodukčních politik

Zde se definují a povolují vazby mezi konzumenty a poskytovateli a nastavují se očekávaná chování služby v podobě tzv. cílů služby (Service Level Objectives). Pokud takový kontrakt není definován, konzument by neměl mít možnost využít danou službou.

Správa a poskytování informací

Úlohou těchto činností je doručení odpovídajících informací všem vlastníkům v podobě pro ně vhodné. Je zřejmé, že pracovníci v IT mají na infrastrukturu SOA jiný pohled s odlišnou mírou abstrakce a požadavků než vedoucí IT pracovníci nebo pracovníci obchodu.

Každé této skupině je nutné prezentovat potřebné informace v pro ně odpovídajícím obsahu i formě.

UDDI registry

Tato úložiště představují vhodný doplněk k repozitory SOA. Uchovávají podmnožinu dat a metadat o službách. UDDI registry obsahují mechanizmus pro klasifikaci, katalogizaci a dynamické náhledy na služby. Takto kategorizované služby pak mohou zjišťovat a využívat přes definované rozhraní i jiné aplikace. Typicky jde o vývojová prostředí, což přispívá k větší produktivitě vývojářů, lepšímu zviditelnění služeb a jejich větší znovupoužitelnosti jak v rámci organizace tak mezi nimi.

4.2 Produkční fáze

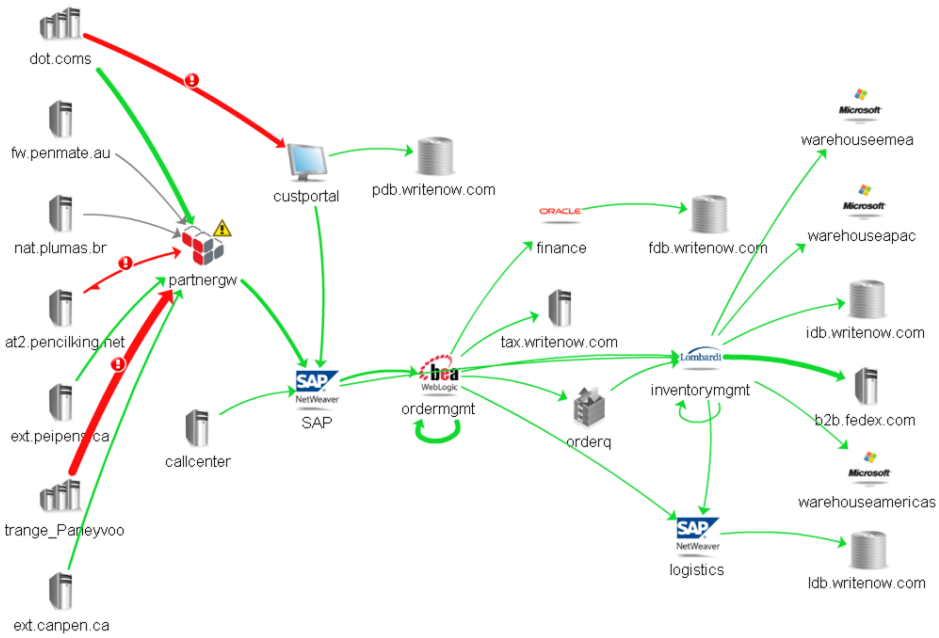
Definice a správa přístupových bodů

Pokud jsou správně nastaveny kroky životního cyklu služby, jsou přístupové body (access points, endpoints) a jejich konfigurace automaticky generovány z repozitory SOA. Tím je elegantně zajištěno, že všechny nasazené služby a jejich konzumenti odpovídají předem definovaným politikám a kontraktům. Zákazníci, kteří repozitory SOA nevyužívají, je musejí tyto přístupové body pracně konfigurovat.

Autodiscovery, autokorelace a vizualizace síťové mapy volání služeb

Jedná se o skupinu činností automaticky zajišťující detekci a vizualizaci SOA v reálném čase resp., zjišťující, jaké služby/operace se v provozu právě teď nacházejí nebo nacházeli. Pokud je zjištěna „podezřelá“ (rogue) služba/operace, např. ta která nemá validní kontrakt nebo nemá kontrakt definován vůbec, není této službě např. umožněna komunikace.

Zde doslova platí: „Co nevidím, nemůžu účinně spravovat, natož řídit“. Vizualizace veškeré komunikace služeb spolu se systematickým kontinuálním monitorováním a zjišťování příčin porušení SLA pravidel pak umožňuje zavést takový režim, v kterém jsou tato porušení okamžitě v reálném čase vidět, je možné zjistit jejich příčinu a vhodně na ni reagovat.



Obr. 4: Ukázka autodiscovery volání služeb na úrovni uzlů.

Vyhodnocování měření a vynucování provozních politik SLA

Skupina činností automaticky zajišťující v reálném čase:

- Sběr a vyhodnocování parametrů komunikace, jako jsou např. počet volání, doba odezvy, velikost zpráv, velikost průtoku dat, chyby nebo výjimky apod.
- Vynucování provozních politik SLA, jako např. „pokud počet volání dané služby za posledních 5 minut je větší než 1000, potom nepřijímej další volání, dokud počet volání neklesne pod tuto hranici.

5 Projekt centrálního monitoringu IT v České pojišťovně

5.1 Shrnutí

Při realizaci komplexního monitorovacího řešení bylo vycházeno z principů ISM – IBM Service Management. Tato metodika a doporučení založená na dlouhodobých zkušenostech zákazníků z celého světa navrhuje základní principy pro komplexní sledování ICT infrastruktury s nedílnými vazbami na obchodní procesy.

Jednou ze základních myšlenek ISM je zajištění znalostí a možnost doručit konkrétní informace přesně v době kdy jsou potřeba ve slíbeném čase a nákladech. Tyto informace jsou dále využity ve vazbě na klíčové obchodní procesy.

Proto bylo jedním z hlavních cílů posunout oblast monitorování infrastruktury od vzájemně nesourodých celků, které si mezi sebou vyměňují minimum informací k jedinému centralizovanému a propojenému řešení.

Transformací na výslednou dynamickou infrastrukturu, která se neustále přizpůsobuje potřebám zákazníka, bylo docíleno:

- Kompletní IT konsolidaci všech vrstev (HW, middleware, aplikace, SOA)
- Důležitou vazbu na obchodní procesy
- Zajištění služeb a aplikací

Základní principy a přístup při řešení takto komplexního projektu byla ochrana investic a parametry TCO a ROI. Bylo tedy dílčím cíle se snažit využít některých stávajících řešení a integrovat je do navrhovaného propojeného celku. Míra propojeních existujících řešení závisela na kvalitě poskytovaných informací a možnosti napojení.

Dalším hlavním cílem, který projekt sledoval, bylo reálné a hmatatelné zvýšení efektivity a snížení nákladů na provoz. Jádro celého systému je obecně připraveno pro spolupráci s libovolnou aplikací třetích stran. Posouzení možnosti integrace bylo provedeno ve studii proveditelnosti a toto posouzení bylo vždy vedeno s cílem maximálně zefektivnit daný proces, nebo operaci.

Dodané řešení bylo navrhováno tak, aby zákazníkovi poskytlo širokou škálu přínosů, jejichž část je uvedena níže:

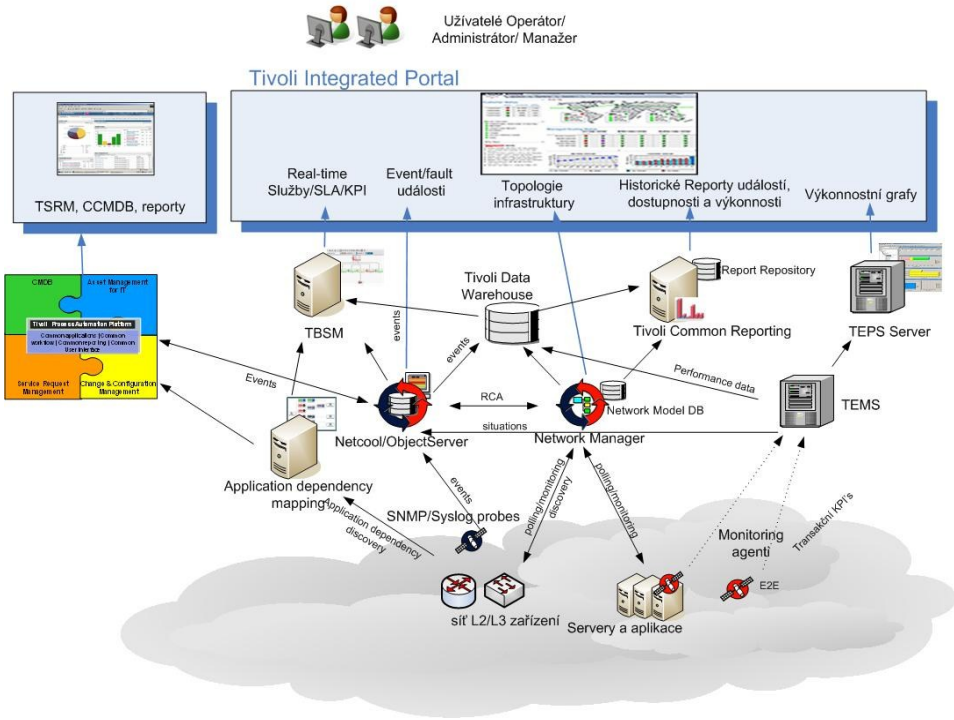
- Úsporu nákladů na provoz IT služeb;
- Lepší kvalitu a spolehlivost IT služeb;
- Optimalizaci využívání ICT zdrojů;
- Menší počet výpadků ICT systémů proaktivním přístupem monitorování a vyhodnocováním politik SLA;
- Zlepší komunikace mezi ICT útvarem a koncovými uživateli
- Zastřešení a sledování komplexního heterogenního ICT prostředí;
- Konsolidace a centralizace dohledových systémů;
- Kapacitní plánování a sledování trendů konfiguračních položek;
- Jediný společný bod pro centralizaci a korelaci událostí;
- Pomoc řídit IT a vizuálně navázat IT na obchodní služby.

Řešení je složeno ze vzájemně propojených komunikujících komponent. Díky úzké vzájemné integraci je řešení schopno poskytnout vysokou dostupnost a flexibilitu při běžném provozu. Obsahuje jediné a jednotné společné rozhraní pro správu a reportování, které šetří práci administrátorům, operátorům i řešitelům. Informace jsou jednoduše přístupné z jednoho místa a vytvoření manažerských reportů a souhrnů je otázka několika málo minut.

Důležitou součástí celého řešení je komponenta pro automatické plnění dat konfigurační databáze, které pravidelně aktualizuje vazby a závislosti aplikací na prvcích ICT infrastruktury. Díky tomu má zákazník okamžitý přehled o provedených změnách (i automatických) a jejich dopadu na prostředí.

Jednotliví agenti a nebo bez-agentní sondy pro získávání informací z infrastruktury pokrývají libovolné prvky a celé komplexní heterogenní prostředí bez ohledu na výrobce, vč. zcela univerzálních agentů nebo naopak agentů vyvinutých pro zcela specifické požadavky zákazníka. Sondy pro tzv. end-to-end aplikační monitoring sbírají reálné hodnoty od uživatelů nebo provádějí robotické spouštění aplikací a skriptů. Na to navazující transakční monitoring sleduje průběh celé transakce a chování požadavků v jednotlivých částech infrastruktury (klient, síť, front-end, back-end, …).

Komponenta Business Service Manager modeluje obchodní služby, navazuje je na ICT zdroje a real-time sleduje aktuální parametry SLA. V návaznosti na data z ERP systému je schopna i měření finančních ztrát v případě výpadku ICT provozu.



Obr. 5: Architektura ITSM řešení IBM Tivoli produkty v České pojišťovně

5.2 Vybrané klíčové vlastnosti governance SOA v ČP

Proč Progress Actional?

ČP v rámci výběrového řízení vyhodnocovala několik srovnatelných technologií. Progress Actional byl vybrán zejména z následujících důvodů:

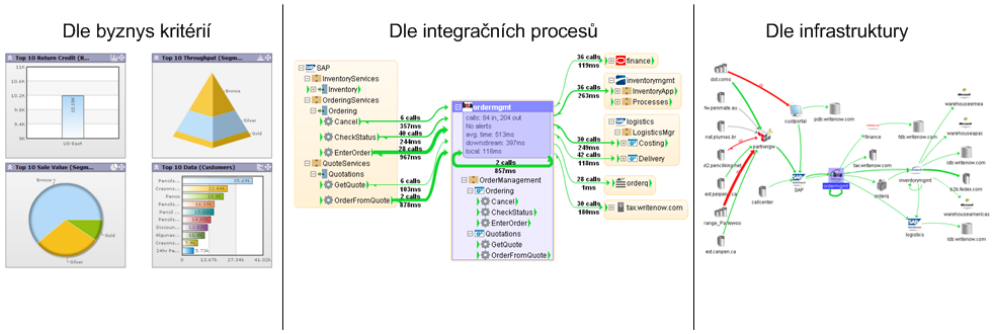
- Rychlost, neinvazivnost a relativní snadnost nasazení.
- Existence sond pro vizualizaci SOA pro všechny hlavní platformy ČP.
- Vhodné API pro integraci s Help Desk (SNMP, portlety, log4j).
- Jednoduchá údržba.
- Rozumná cena.

V následujícím textu uvádíme vybrané vlastnosti Progress Actional, které se v ČP ukázaly jako klíčové pro úspěch projektu.

Automatické zjišťování konzumentů a poskytovatelů

Pro toto automatické zjišťování využívá Actional neinvazivní softwarové agenty s platformově závislými sondami (interceptors). Neinvazivnost tohoto přístupu spočívá v tom, že není nutné zasahovat do zdrojových kódů stávajících aplikací pro to, aby se činnosti governance SOA mohly v plném rozsahu realizovat.

Softwaroví agenti sledují na dané platformě provoz mezi jednotlivými službami (příchozí a odchozí volání, JMS messaging apod.) a ukládají tyto informace do společné databáze. Tato data je pak možné zobrazit jak z pohledu infrastrukturního (node, služba, operace) tak z pohledu byznysového, jak nastiňuje obrázek níže. Tato data zároveň v reálném čase vstupují do politik SLA a slouží k vyhodnocování jejich parametrů a prosazování nastavených pravidel.



Obr. 6: Systém pro správu a řízení SOA nabízí různé pohledy pro různé uživatele.

To poskytuje ČP pohled na SOA„tak jak ve skutečnosti vypadá“ a ne „tak jak ji někdo navrhl a myslí se, že vypadá“.

Je až překvapivé, jak obtížně zákazníci zjišťují, kteří konzumenti právě teď využívají nebo využívali jaké služby, jakých SLA bylo při této konzumaci dosaženo, nebo zda k určité službě nepřistupují neautorizovaní uživatelé.

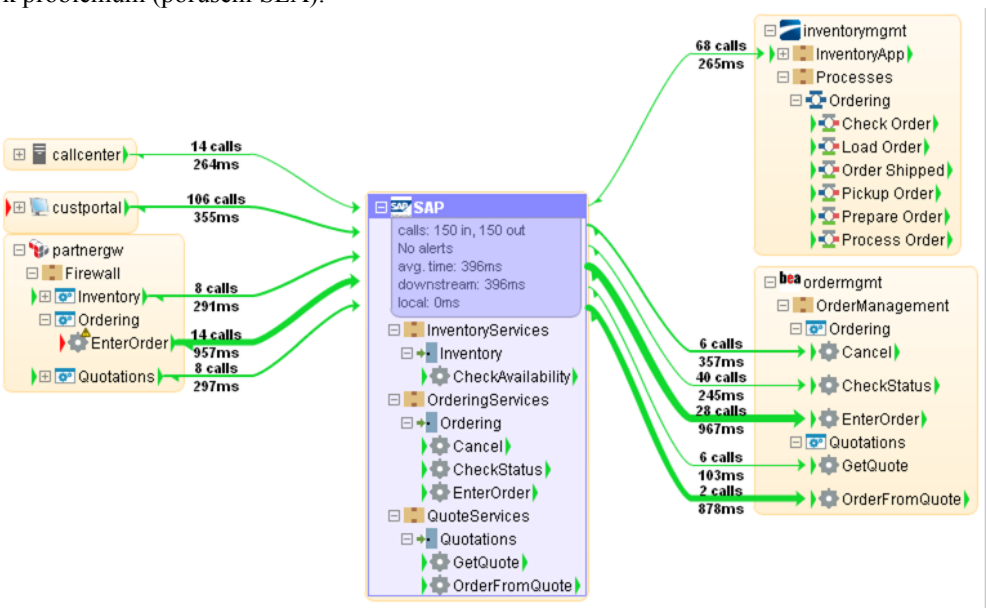
Zpětná vazba do repozitory SOA

Informace zjištěné během provozu služeb jsou sdíleny s repozitory SOA. Architekti, integrační tým, vývojáři i pracovníci provozu ČP mohou například využít aktuální statistiky výkonnosti nebo informace o současné úrovni poskytování služeb, které se do SOA úložiště ukládají k jemnějšímu doladění SLA. Reálná zpětná vazba umožní zlepšit rozhodovací procesy, zvýšit návratnost SOA a uspořit náklady spojené se správou a využíváním služeb. Důležité jsou i sekundární efekty, jako jsou zlepšená morálka uživatelů i pracovníků v IT - všichni vědí, že je využívání služeb monitorováno, vyhodnocováno a účtováno v reálném čase, proto se všichni snaží dodržovat nastavená pravidla.

Automatická korelace volání – síťová mapa volání služeb

Systém governance SOA automaticky svazuje volání konzumentů a služeb a vytvářet tak síťovou mapu těchto volání. Tato mapa pak slouží nejenom pro vizualizaci řetězce volání, ale i pro sledování integračních nebo i obchodních procesů. Toto sledování umožňuje aktivovat příslušné politiky v případě, kdy dojde k určité události nebo naopak k určité události nedojde.

Mapování komunikačních toků vytváří interaktivní topologickou mapu, která ukazuje, kudy (mezi kterými službami nebo systémy) zprávy fyzicky skutečně proudí. Mapování toku volání je důležité i z hlediska způsobu, jakým jsou aplikovány politiky SLA. Je pak možné automaticky vyhodnotit, kde na cestě po směru či proti směru volání dochází k problémům (porušení SLA).



Obr. 7: Ukázka autokorelace volání služeb.

Eliminace podezřelých poskytovatelů

Podezřelá a tím potenciálně nebezpečná služba (rogue service) je každá služba, která neprošla definovaným životním cyklem a byla např. vystavena manuálně administrátorem. Taková služba je vždy rizikem, neboť:

- Může zveřejnit citlivá data a vystavit tak společnost riziku.
- Může využívat kapacitu systému bez možnosti účtování zdrojů.
- Může se vyhnout dodržování zákonů nebo podnikových předpisů.
- Snižuje motivaci dodržovat SLA runtime politiky SLA, protože tyto politiky na ni nelze uplatnit.

Odhalení podezřelé služby probíhá samočinně v reálném čase pomocí funkce pro automatické zjišťování služeb. Nezbytným předpokladem pro její odhalení je využití informací z registrů nebo repozitory SOA, kde je definováno, zda se v daném prostředí vůbec může daná služba nacházet. Systém governance SOA pak dokáže takovou službu okamžitě detekovat a eliminovat do doby, než jsou na ni uplatněna příslušná pravidla a politiky SLA.

Politiky SLA

Politiky SLA se obvykle mění nezávisle na vývoji, nasazení a změnách aplikací. Je proto důležité, aby se zavádění a prosazování politik mohlo uskutečňovat zcela odděleně.

Systém SOA governance v ČP umožňuje vytvořit příslušnou politiku bez toho, aby byla provázána s určitou službou. Taková politika pak může být široce aplikována na všechny služby, skupiny služeb nebo servery v celé síti. A to včetně právě zjištěných podezřelých služeb nebo služeb, které teprve budou do infrastruktury přidány někdy později.

Výstrahy a varování

Každý konzument může mít nastaveny odlišné SLA, využívat jiné služby a být jinak ošetřován. Systém governance SOA umožňuje detailní pohled na jednotlivé konzumenty a na to, s jakou rychlostí či dobou odezvy pro ně příslušné služby pracují. SLA jsou dvouprahová a lze tak definovat například měkký práh pro 80% porušení a tvrdý práh pro 100% porušení. Při porušení daného SLA systém přesně určí, která služba či vazba je za toto porušení odpovědná (například kde dochází k nepřípustným prodlevám a jaká je jejich příčina).

Integrace governance SOA s centrálním monitoringem

Pokud se ČP rozhodla v projektu centrálního monitoring využít technologie různých softwarových dodavatelů, nutné musí čelit otázce integrace těchto technologií. Současné technologie již ale disponují širokou škálou integračních rozhraní, takže zpravidla stačí vybrat ty nejvhodnější. Pro začlenění Progress Actional do IBM Tivoli portálu byly zvoleny následující dva způsoby:

- SNMP trapy generované při každém porušení SLA.
- Portlety zobrazující základní provozní statistiky služeb/operací.

5.3 Zajímavá fakta projektu

Prostředí zákazníka České pojišťovny se vyznačuje svou heterogenností a komplexností, což dokládají tabulky č.1 a č.2. Z tohoto důvodu byl projekt rozdělen do celkem šesti logických modulů, resp. fází projektu, které byly realizovány nezávisle na sobě.

Jednotlivé fáze projektu a jejich cíle zle shrnout následovně:

- INFR – fáze implementace monitoringu infrastruktury, konkrétně serverů (OS, webových a aplikačních serverů), databází (Oracle, MS SQL, DB2), síťové infrastruktury apod.
- EUM – fáze implementace monitoringu vybraných aplikací
- CMDB – fáze implementace konfigurační databáze s nativní podporou ITIL i zakomponováním best practices zákazníka
- SD – fáze implementace konfigurační databáze s nativní podporou ITIL i zakomponováním best practices zákazníka
- SLM – fáze implementace nástroje pro vizualizaci a řízení byznys služeb, SLA i OLA, založených na modulech monitoringu
- SOA – fáze implementace nového monitorovacího nástroje SOA platformy, integrované do jednotného dohledového centra

Implementační fáze projektu se realizovaly v průběhu 17 měsíců s šestiměsíční před implementační a čtyřměsíční post implementační fází. Projekt splnil všechny očekávané cíle s potenciálem dalšího rozšiřování nejenom směrem na IT, ale také byznys komponenty a služby.

Uživatelé IT	Celkový počet uživatelů IT	+12 000
	Celkový počet interních uživatelů	+4 000
	Celkový počet externích uživatelů	+8 000
	Počet zákazníků Service Desku za rok	+120 000
Infrastrukturní portfolio	Celkový počet serverů	+800
	Celková disková kapacita (TB)	+600
	Poboček připojených na WAN	+200
Aplikační portfolio	Celkový počet aplikací	+400
	Celkový počet batch jobů	+13 000
	Celkový počet print jobů za rok	+14 000

Tab. 1: IT fakta prostředí v České pojišťovně (zdroj ČP)

Stav před implementací nástrojů Tivoli	Stav po implementaci nástrojů Tivoli
18 monitorovacích nástrojů zaměřených především na infrastrukturu	3 monitorovací nástroje zaměřené na monitoring infrastruktury i aplikací s funkcionalitou pro řízení SLA i OLA, integrované s nástroji pro řízení a vizualizaci byznys služeb
End User Monitoring monitorující několik klíčových aplikací bez možnosti řízení SLA a OLA	
2 nezávislé konfigurační databáze (CMDB) pokrývající infrastrukturní i aplikační portfolio	1 konfigurační databáze (CMDB) pokrývající infrastrukturní i aplikační portfolio, která je součástí jednotného systému integrující dále Service Desk (SD) a další ITIL procesy jako např. Change a Release Management, správu licencí apod.
1 service desk (SD) vytvořený z nástroje HP PPM, který musel být pro účely SD upraven	
Neexistence SOA monitoringu	1 systém pro SOA monitoring

Tab. 2: Porovnání stavu před a po implementaci projektu (zdroj ČP)

Literatura

1. [BONVAN] Bon van, J.: *IT Service Management: An Introduction*. Van Haren Publishing, 2002.
2. [ITILV3] http://www.itsmf.cz/uws/include/download.asp?file=/uws_files/publikace_ke_stazeni/uvodni_prehled_itil_v3.pdf
3. [ITSMF1] http://www.itsmf.cz/uws/include/download.asp?file=/uws_files/publikace_ke_stazeni/slovnicek_itil_v3.doc
4. [WIKI1] - http://cs.wikipedia.org/wiki/MTTR

Annotation:
The paper describes IT Service Management (ITSM) principles based on IBM and Progress Software product principles. It is a set of technologies for service and its quality management supplemented by SOA monitoring and visualization. The paper is divided into two parts: in theoretic one are introduce

generic terms and logical layers from ICT monitoring including ITSM coming from ITIL standard and SOA governance principles.
In the second part there is a description of an implementation project of IT central monitoring in Česká pojišťovna, one of the largest monitoring projects in Czech Republic as well as Europe. Used software technologies and modules are specified, described a complexity of heterogeneous IT environment and interesting facts are named.

Standardní příspěvky

NoSQL databáze

Jaroslav POKORNÝ

Katedra softwarového inženýrství, MFF UK Praha
Malostranské nám. 25, 118 00 Praha
jaroslav.pokorny@mff.cuni.cz

Abstrakt. Cloud computing je většinou založeno na použití distribuované výpočetní kapacity včetně distribuce dat způsobem, který podporuje dynamické škálování. Databázová platforma v pozadí není vždy možná s použitím klasických databázových prostředků založených na transakčním zpracování s ACID vlastnostmi a plným SQL. Zavedení horizontálního rozdělení dat spolu s uvolněním požadavků na konzistenci databáze vedlo ke vzniku NoSQL databází. Článek popisuje jejich rysy, jako je datový model a způsob dotazování a přístupy k implementaci. Uveden je rovněž přehled některých jejich nejznámějších představitelů NoSQL databází.

Klíčová slova: cloud computing, NoSQL databáze, CAP teorém, slabá konzistence, horizontální škálování, vertikální škálování, horizontální rozdělení dat.

1 Úvod

S rozvojem cloud computingu vystupuje v poslední době do popředí problém služeb využívajících Internet a vyžadujících velká data (angl. *data-intensive services*). Cloud computing se zdá dokonce být budoucí architekturou podporující právě takové aplikace. Přechod od tradičního middleware šitého na míru podnikovým aplikacím ke cloud computing má na životní cyklus aplikací zásadní vliv. Na druhé straně však existují jisté požadavky uvažovaných aplikací, které zatím cloud computing splňuje nedostatečně.

Po mnoho let se ve vývoji informačních systémů spoléhalo na *vertikální škálování* (také zvané angl. *scale-up*), tj. investovalo se do nových a drahých velkých serverů. Bohužel, tento přístup použití architektury sdílení-ničeho vyžaduje vyšší úroveň dovedností a není v některých případech spolehlivý. Po přerozdělení dat za provozu může např. klesnout výkon systému. Rozdělování databáze mezi více (levných) strojů přidávaných dynamicky, tzv. *horizontální škálování* (též zvané angl. *scale-out*), může patrně zajistit škálovatelnost efektivněji a levněji. Než přizpůsobovat běžné SŘBD pro horizontální škálování, zdá se, že dnešní hojně citované NoSQL databáze navržené pro levný hardware a využívající rovněž architekturu sdílení-ničeho mohou být v některých případech řešením ještě lepším. Kromě cloud computingu se NoSQL databáze uplatňují v aplikacích Web. 2.0 a v sociálních sítích, kde horizontální škálování zahrnuje tisíce uzlů. Není náhoda, že NoSQL databáze, které měly na vývoj této kategorie software největší vliv, pocházejí z vývojových dílen firem Google a Amazon.

Aby bylo možné docílit horizontálního škálování, musely ale NoSQL databáze slevit z některých obvyklých databázových charakteristik. To se týká např. omezení relačního datového modelu a obvyklých požadavků na zpracování transakcí. Cílem článku je diskutovat omezení bránící škálování dnešních databází a pokusit se rozšířit úvahy o

trendech v databázích popsané v [13] a [9]. Zaměříme se také na nové architektury SŘBD, kde škálovatelnost má prioritu, a popíšeme omezenou funkcionalitu takových (NoSQL) databází. Otázkou je, jak právě tento software může zajistit pružný rozvoj cloud computingu.

V sekci 2 nejprve uvedeme základní charakteristiky cloud computingu, tak jak jsou dnes všeobecně přijímány. Sekce 3 sumarizuje transakční problémy, které jsou pro škálovatelné databáze kritické. Škálovatelnosti je věnována sekce 4. Sekce 5 pak zmiňuje specializovaná datová úložiště, tak jak se vyskytla v poslední dekádě, a věnuje se jedné jejich variantě - NoSQL databázím. Závěrem zmíníme další vývoj škálovatelných SŘBD, který pokračuje v linii tradičních relačních SŘBD.

2 Cloud computing

Jakkoliv je na cloud computing mnoho různých názorů, je užitečné z nějaké definice vycházet. Uvedeme tu, kterou stanovil National Institute of Standards and Technology (NIST) v roce 2009: *Cloud computing je model umožňující pohodlný síťový přístup na vyžádání ke sdílenému zásobníku konfigurovatelných výpočetních zdrojů (např. sítí, serverů, pamětí, aplikací a služeb), které mohou být rychle dodány a uvolněny s minimálním úsilím investovaným pro řídící činnosti či interakce s poskytovatelem služby*.

Často se uvádí pět charakteristik cloud computingu, které jsou jeho dodavatelům společné:

- samoobslužně a na vyžádání – množství zdrojů je dodáváno bez interakce s uživatelem,
- široký přístup k síti (typicky k Internetu),
- shromažďování zdrojů (jejich velikost, umístění a struktura jsou uživatelům utajeny),
- rychlost pružnosti služby (příděl a uvolňování zdrojů v jakémkoliv množství jsou rychlé a vytváří iluzi neomezené škálovatelnosti),
- měřené služby (provoz a využití zdrojů jsou automaticky monitorovány, měřeny a optimalizovány).

S cloud computingem jsou nedílně svázány další technologie, jako jsou počítání v gridu, SOA a virtualizace, které se běžně vyskytují i samostatně. V tomto článku nás zajímají především technologické problémy cloud computingu vztažené k poskytovateli a/nebo k uživateli. Poskytovatele se týkají zejména:

- konzistence dat,
- dostupnost,
- predikovatelný výkon,
- škálovatelná paměť s vysokým výkonem.

Autoři [10] uvádějí ještě vícenásobný pronájem zdrojů, silné a stabilní API a licencování software.

Z hlediska uživatele je dobré se zmínit o modelu dat. Na rozdíl od podnikových systémů, kde je datový model relativně dobře definován, v cloud computingu se setkáme s více modely, např. pro strukturovaná a nestrukturovaná data, multimedia, metadata apod., navíc pak s modely přicházejícími z použitých zdrojů. Architektura cloud computingu by měla poskytnout možnosti kombinovat tyto modely. Další problémy na uživatelské straně zahrnují bezpečnost a kódování, interoperabilitu a konzistenci zaručovanou používáním transakcí. Tomuto ryze databázovému problému se budeme věnovat v sekci 3.

Z hlediska databází lze uvažovat cloudy poskytující platformu, tj. typ služeb PaaS (Platform-as-a-Service), kde v podpůrné infrastruktuře je poskytnuta databáze. Tyto databáze je možné uvažovat a využívat samostatně (viz Tabulku 1 v sekci 5), nebo jsou součástí širší služby. Známým příkladem v této oblasti je zřejmě AppEngine Google[1].

3 Transakční zpracování

Jedním ze základních rysů databázové technologie je několik vlastností souvisejících s transakčním zpracováním. Jde o *atomicitu* (atomicity - A), *konzistenci* (consistency - C), *izolaci* (isolation – I) a *trvanlivost* (durability – D). Velmi stručně lze tyto vlastnosti po řadě charakterizovat jako „všechno nebo nic“, „výsledkem každé transakce jsou tabulky se správnými daty“, „transakce jsou nezávislé“, „databáze přežije chybu systému“. Konzistenci v uvedeném smyslu budeme nazývat *silnou konzistencí*.

Databázová praxe ukazuje, že ACID transakce jsou požadovány pouze v jistých případech použití. Např. databáze v bankách a obchodu musí vždy obsahovat správná data. V důsledku toho aplikace v byznysu požadují, aby i cloud databáze zaručovala ACID vlastnosti. V souvislosti s cloud computingem pak v tomto případě hovoříme o *korporátní cloud databázi*.

Databáze, které neimplementují ACID plně, mohou být pouze *případně konzistentní*. V principu jde o to, že jestliže se vzdáme silné konzistence, můžeme získat více dostupnosti, což vysoce zlepší škálovatelnost databáze. Takový přístup může být vhodný spíše pro *zákaznické cloud databáze*. Připomíná to datová úložiště pro dokumenty z 90. let, kde ne příliš časté výskyty konfliktů při aktualizacích převažovaly a konzistence nebyla to nejdůležitější.

Na rozdíl od ACID vlastností uvažujme nyní trojici požadavků zahrnujících konzistenci (consistency (C), dostupnost (availability (A)) a toleranci k rozdělení (partitioning tolerance (P)), zkráceně CAP.

- *Konzistence* znamená, že kdykoliv jsou data zapsána, každý, kdo čte z databáze, bude vždy vidět poslední verzi dat.
- *Dostupnost* znamená záruku očekávání, že každá operace skončí v zamýšlené odezvě. Vysoká dostupnost je obvykle dosahována pomocí velkého množství propojených fyzických serverů působících jako jedna databáze s rozdělením dat mezi různé databázové uzly a využitím replik dat.
- *Tolerance k rozdělení* znamená, že z databáze se může stále číst a zapisovat do ní, i když jsou její části zcela nepřístupné. Situací, která to může zapříčinit, je přerušení síťového spojení mezi významným počtem databázových uzlů. Tolerance k rozdělení může být dosažena mechanismy, kterými jsou zápisy místo na určených nedosažitelných uzlech poslány uzlům, které jsou ještě přístupné. Po té, kdy se nedostupné uzly navrátí zpět, obdrží zápisy, které chyběly.

Existuje CAP teorém, také nazývaný Brewerův teorém, navržený Brewerem v [2] a formálně dokázaný v [11]. CAP teorém tvrdí, že pro jakýkoliv systém sdílení dat je nemožné *garantovat* současně všechny tři uvedené vlastnosti. Speciálně ve webových aplikacích založených na strategii horizontálního škálování, je nutné se rozhodnout mezi C a A. Běžné SŘBD upřednostňují C nad A a P.

[1] http://code.google.com/intl/cs/appengine/docs/whatisgoogleappengine.html

Existují dva směry v řešení, jestli C nebo A. Jeden požaduje silnou konzistenci jako klíčovou vlastnost a zkouší maximalizovat dostupnost. Výhodou tohoto postupu je, že ACID vlastnosti umožňují jednodušeji vytvářet aplikace a řídit datové služby. Jinak musí být implementována složitá aplikační logika, která detekuje a kompenzuje nekonzistence. Druhý směr upřednostňuje dostupnost a zkouší maximalizovat konzistenci. Priorita dostupnosti má spíše ekonomické zdůvodnění. Nedostupnost služby může znamenat finanční ztráty.

Připomeňme skutečnost, že existence 2PC protokolu (dvoufázový potvrzovací protokol) zajišťuje konzistenci a atomicitu z ACID. Pak, založeno na CAP teorému, dostupnost nemůže být vždy zaručena a k jejímu zvýšení je nutné slevit z konzistence. Takový transakční model používá například vlastnosti BASE (Basically Available, Soft-state, Eventually Consistent) [14]. Databáze bez silné konzistence znamená, že když jsou data zapsána, ne každý, kdo čte z databáze, bude vidět nová data vždy správně; toto je obvykle nazýváno *případnou konzistencí* nebo *slabou konzistencí*. Dostupnost v BASE je dosažena povolením dílčích chyb tak, aniž by došlo k chybě celého systému.

4 Škálovatelnost databází

Dynamická škálovatelnost jako jeden z klíčových principů cloud computingu se ukázala být pro databáze zvlášť podstatným problémem. Webová místa na vrcholové úrovni vynikají masivní škálovatelností, nízkou latencí, schopností nárůstu kapacity databáze na vyžádání a jednodušším programovacím modelem. Tyto a další rysy současné SŘBD neposkytují cenově právě efektivním způsobem. Relační databáze (tradičně) obsazují jeden server, který může být škálován přidáváním více procesorů, více vnitřní a vnější paměti. Relační databáze umístěná na vícenásobných serverech obvykle používá k udržení synchronizace replikace.

Jeden z fundamentálních požadavků pro zpracování aplikace v cloudu vyžadující masivní zpracování dat je platforma pro podporu škálovatelnosti databáze. Populární relační databáze jako Oracle mají velkou vyjadřovací sílu, je však obtížné ji zvyšovat velkým počtem počítačů místo jednotlivého databázového serveru. Oracle 11g staví myšlenku škálovatelnosti pomocí gridu jednak na vlastním souborovém systému ASM (Automatic Storage Management), kde mřížku tvoří ASM diskové grupy, jednak na mřížce databázových a aplikačních serverů.

Často je nutné jít dokonce níže, tj. až k operačnímu systému. Příkladem může být na Linuxu založený operační systém XtreemOS[2] pro grid.

V poslední dekádě byla vyvinuta nová kategorie škálovatelných SŘBD, sice NoSQL databáze diskutované v sekci 5. Tyto systémy se škálují skoro lineárně s počtem použitých serverů. To je možné díky rozdělování dat. Technicky jde často o metodu *distribuovaných hašovacích tabulek* (Distributed Hash Table - DHT), ve kterých se hašují dvojice (klíč, hodnota) do *kapes* (buckets) – dílčích paměťových prostorů, z nichž každý je umístěn na jednom uzlu sítě.

Horizontální rozdělování dat umožňuje výpočet rozdělit na paralelně zpracovávané úlohy. To samozřejmě není snadno realizovatelné pro jakýkoliv algoritmus a jakýkoliv programovací jazyk. Složitost úlohy pro zpracování dat je minimalizována použitím specializovaných programovacích jazyků, např. MapReduce [7] vyvinutý v Google,

[2] http://www.xtreemos.eu/

objevujících se zejména s použitím NoSQL databází. Připomeňme, že počítání v takovém jazyku např. neumožňuje efektivní implementaci relační operace spojení. Takové architektury jsou vhodné spíše pro zákaznický cloud computing.

Korporátní cloud computing vyžaduje další přístupy, tj. nejenom NoSQL databáze. Rezervy jsou i v relačních SŘBD samotných. Argument, že takové SŘBD nejsou škálovatelné, není vždy zcela pravdivý. Největší instalace SŘBD pracují v intenzivním provozu a s petabajty dat. Takové databáze vyžadují hodně paměti a výpočetní kapacitu pro zpracování. Po dlouhý čas byla omezujícím faktorem tradiční rotující disková zařízení. Dnes může být řešením technologie SSD (solid-state drive). SSD paměti jsou 100× rychlejší v náhodných operacích čtení/zápis než nejlepší disky na trhu (až 50,000 zápisů/s). Takové paměti zlepšují databázovou architekturu sdílení-disků, což může být pro korporátní cloud databáze ideální. Taková architektura mnohdy eliminuje potřebu rozdělovat data.

5 NoSQL databáze

Termín *NoSQL databáze* byl zvolen pro volně specifikovanou třídu nerelačních datových úložišť. Takové databáze nepoužívají (většinou) SQL jako svůj dotazovací jazyk. Termín NoSQL je proto zavádějící a v databázové komunitě je vykládán spíše jako „not only SQL“. Někdy se pro tato datová úložiště používá i termín *postrelační*. V širším smyslu se sem zahrnují i XML databáze, grafové databáze či databáze dokumentů nebo objektové databáze. Někteří představitelé těchto softwarových prostředků nejsou dokonce databázemi v tradičním pojetí vůbec.

Část NoSQL databází obvykle zjednodušuje nebo omezuje režii vyskytující se v relačních databázích s plnou funkčností. Na druhé straně jsou jejich data často organizována do tabulek a přistupována pouze prostřednictvím primárního klíče.

NoSQL databáze také nepodporují operace spojení a `ORDER BY`. Důvodem je, že distribuce řádkových dat je dělána horizontálně. Ztráta při horizontálním rozdělení dat je aktuální také v případě, kdy je použit v každém uzlu i plně funkční SŘBD. Je-li třeba, pro NoSQL databázi může být operace spojení implementována na straně klienta. Bez ohledu na tato omezení umožňují NoSQL databáze vyvíjet užitečné aplikace. Pro úplnost uveďme, že data mohou být rozdělena i vertikálně, tj. např. dvě části záznamu jsou na různých uzlech. I tak lze podporovat horizontální škálování. Další možností je použití replikací

Jak se v těchto databázích uplatňují rysy známé z tradičních přístupů ke zpracování dat, ukážeme v následujících odstavcích.

5.1. Datový model

To, co je v klasických přístupech k databázím nejzákladnější - (logický) datový model, je v NoSQL databázích popsáno spíše intuitivně, bez jakýchkoliv formálních základů. I terminologie je podle toho velice různorodá. Současně se stírá rozdíl mezi konceptuálním a databázovým pohledem na data.

5.1.1. Druhy datových modelů

NoSQL databáze ukládají nejčastěji kombinace dvojic (klíč, hodnota), kde klíč je to, co se v relačních databázích nazývá jméno atributu nebo v SQL databázích jméno sloupce. Alternativně se hovoří přímo o atributech a o sloupcích a tím pádem o *sloupcových NoSQL*

databázích. Některé jsou složeny z kolekcí dvojic (`klíč`, `hodnota`) nebo jde o obecněji *semistrukturované dokumenty* vybavené indexy. Jsou rovněž nazývány (poněkud nevhodně) *dokumentově orientované* NoSQL databáze. Příkladem takového dokumentu může být

```
Jméno:"Jaroslav",
Adresa:"Malostranské nám. 25, 118 00 Praha 1"
Vnoučata:{Klára: "7", Barbora: "6", "Magdalena: "3",
   "Kristina: "1", "Otakar: "3", Richard: "1"}
```

Hodnotou např. `Vnoučata:Barbora` je 6 (s lepší interpretací např. 6 let).

K zápisu datové struktury se v těchto případech obvykle využívá JSON formát[3]. Zde použijeme intuitivní značení vycházející z předchozího příkladu a pouze připomínající JSON.

Jednodušší NoSQL databáze, zvané *úložiště typu klíč-hodnota* (nebo *velké hašovací tabulky*), obsahují pouze množinu dvojic (klíč, hodnota). Jinými slovy jde o množinu pojmenovaných hodnot. Klíč jednoznačně identifikuje hodnotu (typicky řetězec, ale i ukazatel, kde je umístěna hodnota) a tato hodnota může být strukturovaná nebo zcela nestrukturovaná (typicky BLOB). Přístup klíč-hodnota připomíná jednoduché abstrakce, jako jsou souborové systémy nebo hašovací tabulky (DHT), které umožňují rychlé vyhledávání. Podstatné zde ovšem je, že dvojice (klíč, hodnota) mohou být různých typů. V řeči relačního modelu dat – nemusí „pocházet" ze stejné tabulky. Jakkoliv jde o velmi rychlé a škálovatelné databáze, jejich nevýhodou je příliš jednoduchý datový model. Na druhé straně nejsou potřeba hodnoty `NULL`, protože ve všech případech jde o databáze bez schématu.

Grafové databáze jsou vlastně síťovými databázemi, jejich hrany a uzly reprezentují nějaká uživatelská data strukturovaná jako množiny dvojic (klíč, hodnota).

5.1.2. Příklady z praxe

V databázi CASSANDRA[4] se trojici (*jméno*, *hodnota*, *časové_razítko*) říká *sloupec*. Např.

```
{Jméno:"Jaroslav", Adresa:"Malostranské nám. 25, 118 00 Praha
1"}
```

reprezentuje dva takové sloupce (časová razítka nejsou uvedena). *Supersloupce* (nemají časové razítko) obsahují několik sloupců a tvoří vyšší pojmenovanou jednotku, např.

```
Kdo: "Osoba1", {Jméno:"Jaroslav", Adresa:"Malostranské nám.
25, 118 00 Praha 1"}
```

Skupinou sloupců (column family) je (překvapivě) pojmenovaná struktura obsahující neomezené množství řádků. Každý má klíč (jméno záznamu – řádku). Řádky jsou buď tvořeny sloupci nebo supersloupci. Ještě vyšší jednotkou obsahující předchozí struktury je *prostor klíčů* (key space), který je obvykle pojmenován jménem aplikace. Zajímavým rysem je, že lze specifikovat setříděnost v rámci řádku (podle jmen sloupců a rovněž sloupců v supersloupcích)

V BigTable [4] lze datový model charakterizovat jako jistou třírozměrnou mřížku (*tabulku*) jejíž *buňky* (obsahují *hodnotu*) jsou adresovány trojicemi <*klíč_řádku*, *sloupec*, *časové_razítko*>. Na API úrovni slouží trojice pro vyhledávání i pro operace `INSERT` a `DELETE`. Jeden nebo několik sloupců se v BigTable sdružují do pojmenovaných *skupin sloupců* (jiný pojem než v CASSANDRA). Sloupec se pak adresuje pomocí

[3] http://www.json.org/

[4] http://cassandra.apache.org

jméno_skupiny:kvalifikátor, např. `Vnoučata:Barbora`. Skupin sloupců je pevný počet, ve skupině může být pro každý řádek různý počet sloupců.

Časová razítka modelují čas a slouží rovněž k rozlišování verzí hodnoty. Dokumenty obsažené v řádcích tabulky jsou různě velké, lze do nich přidávat další data. Řádky jsou lexikograficky setříděny podle klíčů řádků. Databáze BigTable může obsahovat více takových tabulek.

klíč_řádku	*časové razítko*	*sloupec Jméno*	*skupina sloupců Vnoučata*		
http://ksi....	t1	"Jaroslav"	"Klára" 7		
	t2	"Jaroslav"	"Klára" 7	"Barbora" 6	
	t3	"Jaroslav"	"Klára" 7	"Barbora" 6	"Magdalena" 3

Tab.1. Tabulková reprezentace jednoho řádku v BigTable

Není těžké si představit dvojrozměrnou reprezentaci takové mřížky (viz tabulka 1 vycházející z příkladu v sekci 5.1.1). Každému řádku bude odpovídat tabulka s tolika řádky, kolik je pro řádek použito časových razítek. Sloupců bude mít tabulka tolik, kolik je skupin sloupců. Vzhledem k tomu že skupiny sloupců jsou pro každý řádek různě velké, lze se celkově na taková data dívat jako na tabulku řídkých dat. Na fyzické úrovni je použito vertikální rozdělení dat, kdy skupiny sloupců jedné tabulky se nacházejí na různých uzlech. Např. skupiny sloupců *Zákazník*, *Účty_zákazníka*, *Login_informace* mohou být na třech uzlech chápány jako tři tabulky propojitelné přes *ID_zákazníka*.

Tabulka v BigTable a prostor klíčů u CASSANDRA představují v principu totéž.

Výhodou těchto a podobných databází je bohatší datový model oproti jednouchému přístupu (klíč, hodnota). Data s takovým modelem spadají již spíše do kategorie semistrukturovaných dat. Jména sloupců vlastně představují značky přiřazené hodnotám. Pro aplikace tohoto přístupu se hodí např. uživatelské profily, informace o výrobku, webový obsah (blogy, wiki, zprávy) apod.

5.2. Dotazování

Dotazování v NoSQL databázích je tou nejméně propracovanou složkou. Jednou z možností dotazování nad NoSQL databázemi je (možná pro někoho paradoxně) omezený SQL. Např. v systému SimpleDB[5] má `SELECT` následující syntaxi:

```
SELECT výstupní_seznam
FROM jméno_domény
[WHERE výraz] [třídění] [limit limit]
```

kde `výstupní_seznam` může být: *, `itemName()`, `count(*)`, `seznam atributů`, přičemž `itemName()` se používá pro získání jména prvku. `jméno_domény` určuje doménu, odkud se vybírá. Ve `výraz` lze použít =, <=, <, > =, `LIKE`, `NOT LIKE`, `BETWEEN`, `IS NULL`, `IS NOT NULL` apod. `třídění` je pro jednotlivý atribut vzestupné nebo sestupné, `limit` omezuje velikost výstupu (implicitně 100, maximálně 2500). Operace spojení, agregace a zanořování poddotazů nejsou podporovány.

[5] http://aws.amazon.com/simpledb/

Trochu širší podmnožinu SQL zvanou GQL (Goole Query Language) poskytuje již zmiňovaný AppEngine. Další velmi omezenou variantu SQL používá Hypertable[6]. Jeho jazyk včetně aktualizačních a dalších příkazů se nazývá HQL (Hypertext Query Language).

Typické API pro NoSQL databáze obsahuje operace jako `get(klíč)`, tj. extrakce hodnoty daného klíče. `put(klíč, hodnota)` (vytvoření nebo aktualizace hodnoty daného klíče), `delete(klíč)` (odstranění klíče a jeho hodnoty), `execute(klíč, operace, parametry)` (vyvolá operaci na hodnotě dané klíčem, která je speciální datovou strukturou, např. seznam, množina apod.). Procedurální přístup k dotazování je vlastní např. CouchDB[7].

Díky horizontálnímu rozdělení dat nepodporují NoSQL databáze operace spojení (`JOIN`) a `ORDER BY`. Toto omezení je aktuální také v případě, kdy je v každém uzlu použit plně funkční SŘBD. Je-li třeba, může být operace spojení implementována na straně klienta. Operace selekce je v NoSQL databázích rovněž často popsána na úrovni API, ale i kódu.

Dotazování (a aktualizační operace) se tedy většinou redukuje na přístup prostřednictvím klíče přes jednoduché API (např. hašováním klíče). Celkově se zdá, že rozvoj dotazovacích možností je ponecháno na klientovi, např. přidáním vyhledávání podle klíčových slov, nebo dokonce využití relační databáze pro ukládání metadat o objektech v NoSQL databázi. Takový přístup neznamená nic jiného než ruční programování dotazů, což může být vhodné pro jednoduché úlohy a naopak velmi časově náročné pro jiné.

5.3. Uložení dat

Relační databáze jsou většinou ukládány na disk nebo v nějaké oblasti paměti na síti. Množiny databázových řádků jsou přenášeny do paměti pomocí příkazu `SELECT` jazyka SQL nebo operací uložené procedury.

Některé (ale ne všechny) NoSQL databáze jsou navrženy tak, že pro zvýšení rychlosti jsou jejich data umisťována v paměti s uložením dat na disk při zastavení práce s databází nebo při zálohování dat (např. Redis[8]). NoSQL databáze mohou být umístěny na jednom serveru, ale častěji jsou navrženy k práci v oblaku serverů. Využívány jsou rovněž distribuované indexy. Protože zátěž lze rozložit na více počítačů, můžeme NoSQL databáze chápat jako speciální typ nerelačních distribuovaných SŘBD.

Fyzický datový model je opět víceúrovňový. Databáze se fyzicky jeví jako množina provázaných tabulek (např. v hierarchii) a teprve ty jsou uloženy do souborového prostředí na disku. Využívají se i techniky sloupcově orientovaných databází, které s klíčem asociují množinu skupin sloupců. Takové skupiny jsou ukládány na různých strojích. Sloupcový přístup navíc umožňuje snadné přidávání informací (vertikální škálování) a kompresi dat.

Příklad typického hierarchického uložení nabízí fyzická úroveň v BigTable. Tabulka je rozdělena do tzv. *tabletů*, z nichž každý obsahuje řádky z nějakého intervalu (v souladu s daným uspořádáním). Tablet je identifikován jménem tabulky a koncovým klíčem intervalu. Stejné struktuře se v HBase[9] říká *region* identifikovaný jménem tabulky a počátečním klíčem. Řádky v regionu jsou uspořádány podle klíče od nějakého počátečního

[6] http://hypertable.org
[7] http://couchdb.apache.org
[8] http://redis.io/
[9] http://hbase.apache.org/

do koncového. V CouchDB je pro ukládání dvojic (`klíč`, `hodnota`) použit B-strom tak, že je podporováno setřídění podle klíče.

CASSANDRA používá pro rozdělení dat na jednotlivé servery v prostoru klíčů DHT. Tyto DHT jsou např. organizovány v kruhu s možností dynamizace vkládáním nového uzlu mezi dva uzly resp. sléváním sousedních uzlů. V uživatelském API se manipuluje s DHT opět pomocí operací `put(klíč, hodnota)` a `get(klíč)` → `hodnota`. Uvedeme, jak je DHT řešena u úložiště Valdemort[10]. Data jsou rozdělena do kruhu uzlů, přičemž data z uzlu K, jsou replikována do uzlů K+1,…,K+*n*, pro nějaké dané *n* (tzv. *konzistentní hašování*).

5.4. Vlastnosti transakčního zpracování

Relační databáze podporují v praxi vždy plně vlastnosti ACID. Být relační a ACID není ovšem pro některé případy použití nutné, navíc může přidávat ne nutnou režii. Dosáhnout silné konzistence dokonce nemusí být pro tyto databáze možné. Fixace tolerance rozdělování (P) tedy vyžaduje slabší formy konzistence (C) nebo nižší dostupnost (A) (viz BASE vlastnosti). V případě, že se databáze zaměří na A a P, musí slevit z C. Místo silné konzistence tedy NoSQL databáze implementují případnou konzistenci, přičemž jakékoliv změny jsou replikovány do celé databáze případně, avšak vždy v nějakém daném čase. To znamená, že jednotlivý uzel nebo skupina uzlů nemusí obsahovat posledně aktualizovaná data. Takové databáze pak dosahují nízké latence, vysoké průchodnosti, což činí nějaké participující webové místo více responzivní pro uživatele.

5.5. Architektury NoSQL databází

Jako příklad datového úložiště použitelného pro „vyšší“ systémy zmiňme velmi populární aktivitu Amazonu realizovanou v Single Storage Service (S3)[11]. Na S3 je založen již zmiňovaný SimpleDB. S3 dovoluje zapisovat, číst a odstraňovat objekty velikosti do 5 terrabajtů pomocí jedinečného uživatelsky orientovaného klíče. S3 je nejúspěšnější pro multimediální objekty a zálohování. Takové objekty jsou typicky velké a zřídka aktualizované. Architektura úložiště S3 dovoluje neomezenou škálovatelnost a byla rovněž použita pro vybudování plně funkčního databázového systému s malými objekty a častými aktualizacemi [1] a další NoSQL databáze jako je Dynamo [6].

Jednotlivé architektury používají různé možnosti distribuce, zajištění dostupnosti a přístupu k replikaci dat. Mnohé podporují ACID, jiné případnou konzistenci (CASSANDRA, Dynamo), některé, jako SimpleDB, nepodporují transakce vůbec.

V tabulce 2 podáváme vlastní výběr NoSQL databází spolu s jejich základními charakteristikami zaměřenými na datový model a způsob dotazování. Výraz {hodnota} znamená množinu hodnot.

Některé z NoSQL projektů jsou vyspělejší než ostatní, každý z nich se však pokouší řešit podobné problémy. Seznam různých open a closed source NoSQL databází lze nalézt v [3] a [12], dobře udržovanou a strukturovanou stránkou je http://nosql-database.org/. Vzhledem k odlišnosti jednotlivých NoSQL databází se nezdá reálné, že bude vyvinut nějaký unifikovaný dotazovací standard nebo datový model.

[10] http://project-voldemort.com
[11] http://aws.amazon.com/s3/

Jméno	Výrobce	Datový model	Dotazovací jazyk
BigTable[12]	Google	množina dvojic (klíč, {hodnota})	čtení libovolné buňky tabulky, lze omezit rozsah prohledávaných řádků, sloupců, skupin sloupců
HBase	Apache	skupiny sloupců (je klonem BigTable)	JRUBY (IRB-based Shell - podobný SQL)
PNUTS [5]	Yahoo	(hašované nebo uspořádané) tabulky, typovaná pole, flexibilní schéma	nalezení objektu, rozsahové dotazy, složité predikáty, upořádání, top-k
SimpleDB	Amazon	množina dvojic (klíč, {hodnota})	omezený SQL: select, delete, operace GetAttributes, PutAttributes
Dynamo	Amazon	jako S3	jednoduché čtení a zápis
CASSANDRA	Apache (původně Facebook)	sloupce, skupiny sloupců korespondující k nějakému klíči	jednoduchá selekce přes klíč, rozsahový dotaz, řezy přes vyjmenované sloupce nebo rozsahy sloupců
Voldemort	LinkeId	založen na modelu Dynamo	podobné Dynamo
CouchDB	Apache	dokument jako seznam pojmenovaných (strukturovaných) položek	jednoduchá selekce přes klíč, rozsahový dotaz, pohledy pomocí Javascript a MapReduce
Hypertable	Hypertable	jako BigTable	HQL (Hypertext Query Language)
MongoDB[13]	10gen	objektově strukturované dokumenty ukládané v kolekcích	manipulace s objekty v kolekcích (nalezení objektu či objektů, odstranění, aktualizace, jednoduché selekce)
Redis	Salvatore Sanfilippo	množina dvojic (klíč, hodnota), kde hodnota je jednoduchá typovaná, seznam, uspořádaná (podle ohodnocení) nebo neupořádaná množina, hašovaná hodnota	primitivní příkazy pro každý typ hodnoty

Tab.2. NoSQL databáze - přehled

Úložiště typu klíč-hodnota používané v NoSQL databázích pracují pro jisté druhy dat, ale vůbec nepracují dobře pro jiné druhy. V důsledku toho se firmy, které mají a používají

[12] http://www.google.com/base/
[13] http://www.mongodb.org

NoSQL databáze, nevzdávají SŘBD s SQL. Pokračují v jejich používání pro některé aplikace. Mírný problém se týká programování aplikací s NoSQL databázemi. Aplikace využívající možnost jisté nekonzistence musí být navrženy odlišně než ty, které mohou spoléhat na konzistenci plně zajištěnou SŘBD.

6 Závěr

Ukázali jsme různé přístupy k NoSQL databázím, zejména rysy jejich datových modelů a možnosti dotazování s důrazem na jejich použití v cloud architekturách. NoSQL databáze zatím mají daleko k vyspělým databázovým technologiím a svou koncepcí nemůžou nahradit tradiční relační SŘBD. V práci [8] uvádí Leavitt názory různých šéfů IT významných softwarových firem. Ty se shodují na budoucnosti NoSQL v kontextu využití rozmanitých databázových prostředků aplikačně orientovaným způsobem, jejich širokém využití ve specializovaných projektech zejména s nestrukturovanými daty a vysokými požadavky na škálování. Na druhé straně adopce NoSQL nebude příliš konkurovat využití relačních databází, které reprezentují velké investice a hlavně spolehlivost a vyspělou technologii.

Ukázali jsme, že díky horizontálnímu škálování není možné dosáhnout jednoduše naplnění ACID vlastností. To ovšem neznamená, že jakýkoliv cloud computing je ochotno vzdát se zachování ACID vlastností. Další architektury pro cloud computing, kde se bude využívat horizontálního škálování, zachování ACID a fault-tolerant databázi, budou zřejmě vyžadovat další výzkum. V praxi se již takové systémy dokonce vyskytují. Dobrý úvod do škálovatelných databázových systémů založených na tradičních architekturách poskytuje práce [3]. Patří mezi ně např. relační SŘBD MySQL Cluster[14], VoltDB[15] a Clustrix[16].

Za budoucí trend se považují hybridní systémy s více datovými úložišti založenými obecně na odlišných principech. Hybridním je i již rovněž zmiňovaný Valdemort s MySQL jako jedním z úložišť.

Poděkování Tento výzkum byl částečně podporován grantem GAČR No. 201/09/0990.

Literatura

[1] Brantner, M., Florescu, D., Graf, D., Kossman, D., Kraska, T.: Building a Database na S3. In: *Proc. SIGMOD '08*, June 9-12, Vencouver, Canada (2008), 251-263.

[2] Brewer, E.A.: Towards Robust Distributed Systems. Invited talk on PODC 2000, July 16-19 2000, Portland, Oregon (2000).

[3] Cattell, R.: Scalable SQL and NoSQL Data Stores, 2011, last updated January 28, 2011.

[4] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., Gruber, R. E.: Bigtable: A Distributed Storage System For Structured Data. In: Proc. of 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 06), http://www.usenix.org/event/osdi06/tech/

[14] http://www.mysql.com/products/cluster/

[15] http://voltdb.com/

[16] http://www.clustrix.com/

[5] Cooper, B. F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.-A., Puz, N., Weaver, D., Yerneni, R.: PNUTS: Yahoo!'s hosted data serving platform. PVLDB, 1(2):1277–1288 (2008).
[6] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W.: Dynamo: Amazon's Highly Available Key-value Store. In: SOSP'07, October 14–17, 2007, Stevenson, Washington, USA, ACM, pp. 205-220.
[7] Dean, D., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Communications the ACM, January 2008/Vol. 51, No. 1 (2008) 107-113.
[8] Leavitt, N.: Will NoSQL Databases Live Up to Their Promise? Computer, Vol. 43, No. 2 (2010) 12-14.
[9] Feuerlicht, G., Pokorny, J.: Can Relational DBMS Scale-up to the Cloud? In: Proc. of ISD 2011, (2011), submitted.
[10] Frischbier, S., Petrov, I.: Aspects of Data-Intensive Cloud Computing. In: From Active Data Management to Event-Based Systems and More, LNCS 6462, Springer 2010, pp. 57-77.
[11] Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility consistent, available, partition-tolerant web services. Newsletter ACM SIGACT News, Vol. 33, Issue 2 (2002) 51-59.
[12] Intersimone, D.: The end of SQL and relational database? (Part 2 of 3). Computerworld, February 10, 2010, http://blogs.computerworld.com/15556/the_end__sql_a_relační_databáze_part_2__3.
[13] Pokorný, J.: Databases in the 3rd Millennium: Trends and Research Directions. Journal of Systems Integration, Vol 1, No 1-2 (2010) 3-15.
[14] Pritchett, D.: BASE: An Acid Alternative. ACM Queue, May/June (2008) 48-55.

Annotation:
The paper is focused on so called NoSQL databases. In context of cloud computing, architectures and basic features of these databases are studied, particularly their horizontal scalability and concurrency model, that is mostly weaker than ACID transactions in relational SQL-like database systems. Some characteristics like a data model and querying capabilities are discussed in more detail. The paper also contains an overview of some representatives of NoSQL databases.

Adaptívny pripomienkovač: vplyv kontextu na vzory správania

Dušan ZELENÍK, Mária BIELIKOVÁ

Ústav informatiky a softvérového inžinierstva, Fakulta informatiky a informačných technológií, Slovenská technická univerzita , Ilkovičova 3, 812 19 Bratislava
[zelenik, bielik]@fiit.stuba.sk

Abstrakt. Veľa činností, ktoré vykonávame významne závisí od kontextu, v ktorom sa dejú. Smerovanie k personalizácii, teda k uvažovaniu aspektov individuálnych používateľov, však nezahŕňa všetky aspekty. V tomto článku prezentujeme metódu pre adaptívne pripomienkovanie udalostí na základe vplyvu kontextu. Navrhnutá metóda pre pripomienkovanie automaticky plánuje a organizuje udalosti podľa aktuálnych vlastností prostredia, v ktorom sa používateľ nachádza. Takto adaptívne plánujeme a upozorňujeme používateľa na vhodný čas odchodu z jedného miesta, tak aby načas stihol príchod na plánovanú udalosť. Vytvorili sme mobilnú aplikáciu, ktorá spĺňa úlohu monitorovacieho zariadenia najmä polohy používateľa v čase. Záznamy o pohybe následne analyzujeme s účelom objavenia vzorov správania s ohľadom na kontext. Na základe tejto predikcie odporúčame plán, ktorý zabezpečí, že používateľ bude informovaný o nadchádzajúcich udalostiach so zohľadnením jeho kontextu. Takto ho vieme napr. skôr zobudiť ak zistíme, že dážď zdrží jeho cestu do práce a preto potrebuje odísť z domu skôr.

Kľúčové slova: kontexty, monitorovanie používateľa, adaptívne pripomienkovanie, predikcia správania, odhaľovanie vzorov správania.

1 Úvod

V súčasnosti takmer každý používa mobilný telefón, či podobné zariadenie, najmä s účelom podpory komunikácie alebo organizovania času. Mobilné zariadenia sú s nami neustále a pritom i najlacnejšie z nich dnes už môžeme využiť na monitorovanie našich aktivít. Vždy môžeme byť s určitou presnosťou lokalizovaní operátorom využitím telestriálnych GSM veží a sily signálu alebo GPS pomocou GPS modulov. Spoznávanie aktuálnej polohy nám pomáha pri navigácii, ale i v iných situáciách, napr. keď sa potrebujeme rozhodnúť kedy sa presunúť z jedného miesta na druhé, či kedy sa ráno zobudiť.

Monitorovanie prostredia, v ktorom sa nachádzame a hľadanie atribútov prostredia (kontext prostredia [7]) nás privádza k možnostiam pokročilej analýzy a následne k prispôsobovaniu. Prispôsobovanie v konečnom dôsledku vplýva na pohodlie a kvalitu nášho života. V prípade mobilných telefónov s pokročilejšou funkcionalitou vieme vytvárať ďalšie možnosti ako sa prispôsobiť v prospech používateľa.

V tomto článku sa venujeme metóde pre pripomienkovanie nadchádzajúcich udalostí, ktoré čakajú používateľa. Udalosťami rozumieme aktivity, ktoré používateľa čakajú. Môže to byť začiatok pracovnej doby alebo i dohodnutý obed v konkrétnej reštaurácii. Metóda, ktorú sme navrhli umožní plánovanie odchodu z jednej lokality tak, aby používateľ stihol plánovanú udalosť v druhej lokalite. Príkladom je adaptívne zobúdzame používateľa v ranných hodinách podľa dopravnej situácie a podľa obvyklých časových potrieb tohto

používateľa na prípravu a transport. V prípade, že začne v noci snežiť, predpovedáme predĺženie času odhadovaného na transport a upravíme čas zobúdzania. Na druhej strane, pripomienky posunieme, či úplne potlačíme, ak sa používateľ už nachádza na danom mieste, prípadne už plní plán prepravy.

Rôzne atribúty prostredia (kontexty) vplývajú na predikciu, ktorá pomáha odhaľovať znalosti aplikované v takomto pripomienkovači. Kontexty môžeme rozdeliť na kontexty minulosti, prítomnosti a budúcnosti podľa spôsobu, akým kontexty využívame [11]. Náš používateľ nevie, čo sa stane budúce ráno, ale z jeho minulosti poznáme reakcie na stavy z minulosti. V aktuálnej situácii sme preto pripravení vykonať akciu podľa pozorovaní histórie a predikcie nastávajúcich javov. Hneď ako je známy súčasný kontext prostredia, v ktorom sa používateľ nachádza, musíme reagovať. V prípade ranného budenia zobúdzame používateľa o pár minút skôr v prípade, že sneží.

V našom prípade uvažujeme kontexty prostredia, ku ktorým hľadáme vhodné riešenia v podobe reakcií. Sústredíme sa na tieto kontexty a k ním príslušné pozorovania:

- poloha (aktuálna, plánovaná),
- čas (aktuálny, potrebný na prepravu, čas konania udalosti),
- typ prepravy,
- počasie.

Tieto kontexty pozorujeme dlhodobo, aby sme mohli vytvoriť čo najpresnejšie odhady a tým aj vhodné prispôsobenie. Zmeny vzorov správania sa samozrejme v čase menia, preto najaktuálnejšie pozorovania získavajú väčšiu váhu. Lokalita je monitorovaná neustále pričom je spojená s časom, kedy bola zaznamenaná.

Vzory opakovania hľadáme objavovaním vzorov v záznamoch z monitorovania. To umožňuje robiť odhady na základe toho, ako používateľ riešil aktuálne situácie v minulosti (dĺžka prepravy z miesta na miesto, vplyv počasia na prepravu, cesta do práce, domov a tiež obvyklé časy, kedy používateľ tieto lokality navštevuje). Lokalita a s ňou spojený čas sú jediné vstupy, ktoré získavame priamo od používateľa, ktorý samozrejme súhlasí s použitím týchto údajov v jeho prospech. Ostatné informácie vieme získať:

- využitím dodatočných zdrojov informácií z Internetu (počasie využitím času a lokality) alebo
- výpočtom (typ prepravy využitím pozorovania pohybu).

Naša metóda pritom nepotrebuje presné informácie o polohe. Nepotrebujeme poznať presnú lokalitu v podobne GPS súradníc. To nás oslobodzuje od využívania GPS modulov, ktoré nadmerne ovplyvňujú výdrž batérie a teda dĺžku prevádzky zariadenia. GPS súradnice pritom poskytujú presnú lokalitu, zobraziteľnú na mape. Avšak pre potreby zohľadnenia kontextu potrebujeme iba oddeliť jedinečné lokality, k čomu postačuje GSM pripojenie, ktoré je i tak neustále aktívne. Výsledkom monitorovania je teda lokalita v podobe identifikátora GSM veže s najsilnejším signálom v danom momente.

Celé riešenie staviame do možného využívania sily davu, ktorý metódu využíva. Dav ľudí je schopný vytvoriť presnú mapu s časmi potrebnými na prepravu, ktorá môže byť použitá v prípade, ak nie sme z pozorovania jednotlivca schopní predpovedať koľko potrvá preprava doposiaľ používateľom nepoznanej trasy.

2 Existujúce riešenia

Pod termínom *kontext* rozumieme súbor atribútov situácie alebo podmienky [11]. Atribúty môžu opisovať prostredie používateľa od jednoduchších vlastností ako je čas alebo lokalita až po komplexnejšie ako je spoločnosť, ktorá používateľa obklopuje, kde patria aj emócie. Prostredie zahŕňa aj zariadenia, s ktorými pracujeme a ich nastavenia.

Ciglan a ďalší v práci [5] predstavili udalosť ako dianie, ktoré je dôležité pre väčšinu. V spomenutej práci rozumejú pod udalosťou napríklad Vianoce, Veľkú noc alebo Nový rok. V práci využívajú štatistiky z prezerania stránky Wikipeadia. Zobrazenia jednotlivých stránok napomáha k zisteniu aktuálneho diania alebo významných udalostí v určenom časovom intervale. Autori na analýzu využili nadpisy a prepojenia medzi udalosťami.

Podobne aj my analyzujeme správanie používateľov. My sa však venujeme jednotlivcom a ich správaniu na podstatne väčšej úrovni granularity, a nie masy, ktorá určuje niekoľko dôležitých udalostí. Teda neuvažujeme pod termínom udalosť iba populárne dianie relevantné pre mnohých, uvažujeme jednotlivca a udalosti, ktoré sú relevantné najmä pre tohto jednotlivca. Pracujeme so vzormi správania vo forme opakovania sa udalostí pre jednotlivca skôr v kratších intervaloch. Namiesto ročne opakovanej udalosti Vianoce, hľadáme udalosti ako príchod do práce či školy. Napr. pozorujeme používateľa, ktorý býva vždy v pracovných dňoch v práci od 8 hodina rána do 4 večer. Táto udalosť je dôležitá pre jedného, pritom však nesúvisí s návykmi ostatných.

Analyzujeme správanie jednotlivcov, aby sme mohli objavovať udalosti, ktoré sa spájajú s potrebami presunu z miesta na miesto. Podobne ako v práci venujúcej sa predpovedaniu a adaptovaniu podľa vykonávanej úlohy [12] monitorujeme používateľa prostredníctvom mobilného telefónu. Úloha v spomenutej práci je definovaná ako nadchádzajúca aktivita používateľa. V práci autori prezentujú prístup, ktorý môže poslúžiť napr. ako automatický vypínač zvonenia na mobilnom telefóne. Metóda predpovedá napr. stretnutie v práci, a preto automaticky stíši alebo vypne zvonenie mobilného telefónu. Riešenie prezentované v spomenutej práci je navrhnuté tak, aby predpovedanú úlohu aplikácia ponúkla používateľovi. My sa v našej aplikácii realizujeme iba jeden druh úlohy – prepravy z jedného miesta na druhé. Túto úlohu taktiež predpovedáme využívaním naučených vzorov v správaní používateľa.

Predikcia je v našom prípade založená na rituáloch používateľa podobne ako prezentuje Bamis v [2]. V práci navrhuje používanie množstva dát získaných z rôznych senzorov. Príkladom môže byť použitie teplomeru na zistenie informácie o tom, či používateľ spí alebo nie (s predpokladom, že telesná teplota je vyššia ak sa pohybuje a nespí). V našej práci používame pozorovania s vyššou abstrakciou. Nepoužívame nízkoúrovňové merania atribútov prostredia, ale skôr informácie v podobe pozorovania lokality, času, prípadne počasia. Z týchto kontextov a aktivity podobne ako v spomenutej práci odvodzujeme rituály používateľa. Postupne napr. zisťujeme v akom čase býva doma, kedy býva v práci alebo kedy a kde obeduje. Taktiež uvažujeme pohodlie používateľa a teda sa sústredíme iba na mobilný telefón ako zdroj týchto informácií. V práci, ktorú prezentuje Bamis [2] využívajú veľmi presné a užitočné dáta, ktoré sa však získavajú ťažko v prípade obyčajných ľudí.

Ďalšou možnosťou ako získať kontexty a teda informácie o prostredí je priama otázka, ktorú možno formulovať používateľovi. Takto explicitne vieme získať pomerne kvalitné informácie (v prípade, že používateľ je ochotný spolupracovať). Odporúčanie hudby, ktoré prezentuje vo svojej práci Baltrunas [1] využíva zložité používateľské rozhranie, cez ktoré takéto informácie o prítomných kontextoch získavajú. Používateľ rozhoduje o tom, kedy je vhodné počúvať zvolenú hudbu, a tak pomáha ostaným používateľom. V práci [9] autori využívajú dokonca špeciálne upravené diaľkové ovládanie, ktoré umožňuje priamu explicitnú formu vyjadrenia aktuálnych pocitov používateľa. Divák, ktorý sleduje televíziu

tak môže označiť svoju aktuálnu náladu za smutnú, či veselú. Tento emocionálny kontext ďalej využívajú pri odporúčaní programov. Alternatíva je opísaná v práci [4]. Navrhuje metódu, ktorá využívaním štatistických prístupov nahrádza explicitné získavanie kontextov. Odporúčania tu generujú využívaním najmä kontextu času a s tým spojeným správaním sa používateľov. Masa používateľov a v dôsledku vzniknutý pravdepodobnostný model je hlavným pôvodcom odporúčaní.

Mei [10] tiež navrhuje využiť silu masy používateľov s cieľom pomoci jednotlivcovi. V práci sa uvádza metóda pre navrhovanie dopytov podľa populárnosti u väčšiny iných používateľov. V aktuálnom čase s podobnou úlohou tak používatelia akoby spoločnými silami vyhľadávali informácie na webe. Kontexty využívané v tejto práci su krátkodobé. Opierajú sa o predpoklad, že používatelia majú záujem o podobné informácie v rovnakom čase. Toto samozrejme platí najmä pre trendy a populárne informácie a môže spôsobiť veľké nepresnosti pri veľmi špecifických používateľoch. My podobne navrhujeme riešenie spoločného dosiahnutia cieľu pomocou skupiny používateľov. Pracujeme však aj s takou alternatívou, kde predpokladáme uzavretosť sveta jednotlivca a neprístupnosť znalostí objavených pomocou masy. Takto zabezpečujeme, že i dnešná väčšina používateľov bez neustáleho mobilného pripojenia do internetu bude môcť využívať prínosy našej metódy.

Keďže pracujeme s lokalitami, uvažujeme aj vedľajší produkt, ktorý vzniká. Používatelia s mobilnými telefónmi postupne prechádzajú cez oblasti v mestách a takto postupne vytvárajú mapu tohto mesta. V konečnom stave tak môžeme získať vzdialenosti medzi rôznymi lokalitami v tomto meste, či oblasti. Mapa vzniká nie ako analógia geografickej mapy, ale ako mapa časových vzdialeností medzi lokalitami. Clarke [6] vo svojej práci spomína hru, kde veľmi podobne používatelia s cieľom hľadať artefakty v teréne postupne preskúmavajú oblasť. Podobne vzniká mapa vzdialeností, ktorá sa neskôr môže použiť na plánovanie presunov, či navigáciu. V našom prípade tvorby mapy sa sústreďujeme na časové vzdialenosti medzi jednotlivými lokalitami. Každý používateľ vytvára takúto mapu najmä pre seba a svoje návyky na prepravu a lokalitu. Kolaboratívne však vieme poskytnúť používateľovi aj mapu vzdialeností medzi lokalitami, ktoré doposiaľ nenavštívil. Okrem toho je koncept otvorený na zapracovanie ďalších atribútov ako počasie a jeho vplyv na prepravu.

3 Kontexty a prispôsobovanie

Metóda pre pripomienkovanie udalostí, ktorú sme navrhli, je založená na vzoroch správania a objavovaní kontextov. Získavame niekoľko kontextov, konkrétne napríklad lokalitu, čas alebo počasie. Pracujeme s mobilným zariadením, ktoré je neustále zapnuté a monitoruje používateľa. V tejto časti najprv ukážeme ako získavame kontexty a akým spôsobom monitorujeme používateľa. Ďalej opisujeme objavovanie vzorov správania, alebo inak aj rituálov vzhľadom na získané kontexty. Nie všetky kontexty považujeme za dôležité, lokalita a čas sú najvýznamnejšie kontexty, ktoré v tejto práci využívame.

3.1 Poloha

Dnes už mnohé mobilné zariadenia umožňujú získavať informácie o polohe pomocou GPS. GPS moduly sú už dnes veľmi presné a získanie lokality a jej presné mapovanie na geografickú polohu dosahuje vynikajúce výsledky. Táto technológia má však aj viaceré nedostatky ako je energetická spotreba a dlhší čas inicializácie. Okrem toho je takmer nepoužiteľná vo vnútorných priestoroch, či prikrytých miestach. Napokon, nie každé zariadenie je vybavené GPS modulom a používatelia nenechávajú tento modul aktívny.

Pre uvedené dôvody sme sa rozhodli používať GSM signál napriek nepresnostiam a faktu, že nie je určený na lokalizáciu. Lokalita sa v tomto prípade dá získavať i použitím GSM veži a signálu. Každá veža má svoje vlastné identifikátory a pri pripojení telefónu na tieto veže vieme identifikovať aj veže. Jedinečnosť identifikátorov zaručuje, že lokalita môže byť odlíšená na takej úrovni podrobnosti, aká je pre prispôsobovanie postačujúca.

Pochopiteľne vieme určiť GPS súradnice týchto veží. Súradnice sú presné, ale nehovoria o exaktnej polohe používateľa, keďže hovoríme o aproximácii na stovky metrov (v urbanizovanej oblasti). Presne miesto teda získať nevieme, ale vieme odhadovať polohu dokonca i bez GPS súradníc. Dôležité je totiž odlíšiť lokality od seba, nie hľadať ich na geografickej mape. Z rovnakého dôvodu nepotrebujeme prekladať identifikátory veží na GPS súradnice a preto netreba ani pripojenie na internet, či prenášanie a aktualizovanie databázy týchto veží na mobilné zariadenie. Jedinečné veže používame na odhadovanie časovej vzdialenosti medzi lokalitami, ktoré sú potom základom pre pripomienkovanie.

Každá udalosť, ktorú plánujeme pripomienkovať a začleniť do plánu používateľa je spojená s lokalitou, kde sa odohráva. Náš pripomienkovač sa učí deň po dni pomocou záznamov o pohybe používateľa. Každá udalosť takto vlastne zaznamenaná a môže byť označená ako rituál, ak sa opakuje. Presne vieme kedy a kde sa udalosť u používateľa vyskytovala, čo slúži ako základ na odhaľovanie rituálov. Po dlhšom pozorovaní sme tak schopní predpovedať udalosti a teda ich rovno plánovať pre používateľa automaticky. Ak sa udalosť blíži, potom sa stane lokalita tejto udalosti dočasným cieľom a čas konania znížený o čas potrebný na prepravu sa stane časom, kedy bude oznámená pripomienka. Čas potrebný na prepravu sa pritom vypočíta podľa dočasného cieľa a súčasnej lokality.

Miesta konania udalosti objavujeme v dátach získaných monitorovaním používateľa. Zaujímavé lokality označujeme ako tie, ktoré sa často opakujú alebo sa na nich používateľ vyskytuje dlhšiu dobu. Nie sú to lokality, ktorými iba prechádza, či navštevuje skôr náhodne. Podobne rozpoznávame napríklad miesto kde používateľ býva. Veľa sa tu zdržuje i keď prichádza na toto miesto nepravidelne. Domov preto nie je označený za udalosť, ktorú treba pripomienkovať, ale za zaujímavú lokalitu.

Na obrázku 1 prezentujeme lokality, ktoré navštívil používateľ (ktorý experimentálne používal našu aplikáciu počas 2 mesiacov). Časové vzdialenosti medzi prepojenými lokalitami sú dĺžky 5 minút. Interval zachytávania lokalít napríklad spôsobil, že dlhší výlet v ľavej časti má iba niekoľko spoločných lokalít na ceste tam a späť. Interval by sme mohli i skrátiť avšak nie je to potrebné, pretože samotný výlet nie je z pohľadu rituálov zaujímavý a nedostatočné zaznamenávanie lokalít nie je problémom.

Obr. 1. Pohyb používateľa počas doby 2 mesiacov. Každý kruh je lokalita. Veľkosť kruhu reprezentuje počet návštev lokality. Dva zhluky v pravo dole sú domov a práca. Dlhá cesta je výlet do neznámej lokality, kde sa používateľ nejakú dobu zdržal a potom sa vrátil.

3.2 Čas

Každú lokalitu vrátane tých, ktoré používateľ navštívil iba raz na krátky čas, zaznamenávame spolu s časovou známkou. Čas je ďalší významný atribút, ktorý je súčasťou kontextu. Neagregujeme udalosti podľa časti dňa ako v podobných projektoch [4]. S časom pracujeme v súvislosti s lokalitami. Čas uvažujeme ako hlavný atribút vplývajúci na rituály používateľa. Rovnako pracujeme s časom ako jednotkou dĺžky prepravy medzi dvoma lokalitami.

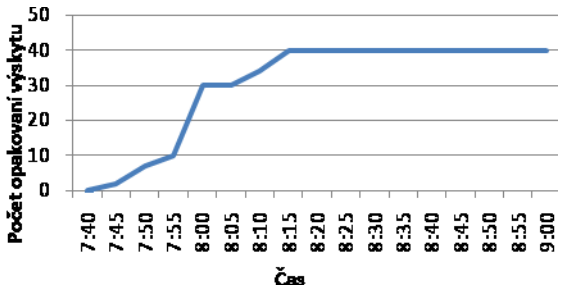
Rituály objavujeme postupným hľadaním najvhodnejších intervalov, ktoré sú medzi jednotlivými udalosťami. Hľadáme opakovania postupným ladením dĺžky intervalu. Algoritmus pre odhaľovanie rituálov pracuje v nasledovných krokoch:

1. Blízke lokality sa spoja (napr. v domácnosti môže byť viac lokalít)
2. Nefrekventované lokality sú vylúčené (menej ako 15 minút)
3. Hľadaný interval je nastavený na dĺžku pozorovania (maximum)
4. Lokality sa zoskupia podľa tohto intervalu
5. Histogram pre každú lokalitu podľa intervalu je vypočítaný
6. Histogram pre každú lokalitu je ohodnotený (vzniknú páry lokalita - skóre)
7. Hľadaný interval sa zmenši alebo zväčší ak sa dajú zvýšiť skóre inak koniec
8. Skok späť na krok 4 s novou dĺžkou intervalu

Kroky od 1 až 3 sa vykonávajú pre filtrovanie nepotrebných záznamov a zjednodušenie ďalšej analýzy. Filtrujeme lokality, ktoré boli zaznamenané pretože ich používateľ navštívil, avšak z hľadiska objavovania rituálov nie sú zaujímavé, pretože sa tam používateľ nezdržuje dostatočne dlho. 5 minút je najmenší pozorovateľný čas (záznam o aktivite robíme každých 5 minút). Hranicu 15 minút sme nastavili ako najnižšiu prípustnú a pritom dostatočne zaujímavú z pohľadu hľadania lokalít, kde sa používateľ zdržal.

Predchádzajúci cyklus je použitý ako sekvenčné hľadanie najvhodnejšieho intervalu medzi opakovaním udalosti. Iterácie postupne aproximujú najvhodnejší výsledok. Dôležitým krokom (krok 6) je ohodnotenie intervalu a histogramu, ktorý jeho použitím vznikne. Pre jednu lokalitu sa skóre vypočíta ako priemerná frekvencia opakovania počas intervalu v pomere ku frekvencii v extréme.

Výsek takéhoto histogramu je zobrazený na obrázku 2. Intuitívne vidíme, že od 8:15 sa používateľ nachádzal na pracovisku 40 krát. Najvhodnejší interval, ktorý odhalíme pre každú lokalitu pomocou algoritmu, označujeme ako interval opakovania rituálu. Väčšina lokalít nemá interval s dostatočným skóre, preto nie sú súčasťou žiadneho rituálu.



Obr 2. Histogram pre krátky výsek záznamov (okolo 8:00 ráno, práca).

Keď už vieme predpovedať kedy bude najbližšia udalosť, na ktorú sa musí používateľ dopraviť, ostáva ešte poznať čas, ktorý potrebuje používateľ na prepravu. Toto odhadujeme zo súčasnej lokality a cieľa. Okrem toho, hlavne v prípade rannej prípravy potrebujeme zvážiť extra čas, ktorý je potrebný pre používateľa. Príprava pritom môže byť rôzna vzhľadom na ďalšie prítomné kontexty. My tento čas odhadujeme iba ako rezervu vypočítanú podľa predchádzajúcich pozorovaní.

3.3 Okolnosti prepravy

Okrem pozície a času môžeme pracovať aj s ďalšími kontextami. V tejto časti diskutujeme najmä o type prepravy a počasí. Obe tieto okolnosti prepravy nie sú priamo získateľné. Treba analyzovať správanie používateľa a čas, ktorý potrebuje na prepravu. Porovnaním viacerých používateľov potom vieme odhaliť typ prepravy ako napr. pešo, hromadnou dopravou alebo autom. V skutku je odhalenie typu prepravy vhodné iba vo výnimočných prípadoch ľudí, ktorí typ prepravy striedajú. Ak chodí taký používateľ do práce striedavo autom a hromadnou dopravou, potom má význam tieto typy odlišovať. Časy potrebné na túto prepravu sa menia. Typ prepravy, v prípade, že sa strieda náhodne, je takmer nemožné určiť a teda i čas potrebný na transport odvodzujeme od priemerných hodnôt.

Inou, viac zaujímavou podmienkou prepravy, je počasie. Počasie ovplyvňuje presun z miesta na miesto najmä v oblastiach, kde vďaka počasiu môže dochádzať k nehodám, zápcham či inému zdržaniu. Sneženie, dažde a podobné podmienky ovplyvňujú čas, ktorý treba na prepravu. Ak používateľ má túto informáciu, môže sa pripraviť na dlhšiu cestu. Avšak v prípade, že používateľ napr. spí, nevie o zmene podmienok a nemôže svoj plán upraviť. V týchto prípadoch nastupuje naša aplikácia, ktorá má za úlohu predpovedať časy potrebné na prepravu za rôznych podmienok. Nutnou podmienkou je pripojenie k internetu prostredníctvom mobilného zariadenia, aby sa aktuálne počasie mohlo zvážiť pri tvorbe denného plánu prepráv. Lokalita a čas sa použije na získanie počasia a teda ďalšej informácie, ktorá pomôžu spresniť čas potrebný na prepravu.

Každý nový kontext prináša do systému, kde má byť zapracovaný, nový rozmer. Nie iba typ prepravy a počasie pritom môžu byť zahrnuté. Rovnako vplýva na čas prepravy aj čas dňa. V ranných hodinách sú cesty plnšie a cesta časovo dlhšia. V skutočnosti časť dňa považujeme za najpodstatnejší a najintenzívnejší kontext, ktorý vplýva na používateľa [8]. V našom riešení využívame túto sústavu kontextov:

(*lokalita*) *x* (*počasie*) *x* (*typ prepravy*) *x* (*časť dňa*)

Znamená to, že ku každej kombinácii kontextov prislúcha záznam o časovej dĺžke prepravy. S množstvom rozmerov sa zvyšuje zložitosť výpočtu. Zložitosť čiastočne znižujeme tým, že vieme pracovať s jednotlivcami a projekciou týchto dimenzií na jeho konkrétny prípad. Mapa časových vzdialeností, ktorú sme diskutovali vyššie, sa vytvára kolaboratívne. Tu treba uvažovať úplný súbor dimenzií, avšak výpočet môžeme vykonať offline na výpočtovo výkonných prostriedkoch.

4 Experimenty

Experimentovali sme s dvoma hypotézami. Prvou je prítomnosť rituálov správania a ich prirodzená dĺžka opakovania. Druhou hypotézou je vplyv kontextov na tieto rituály. Oba experimenty, ktoré tu prezentujeme sme uskutočnili na vzorke aktivity jedného používateľa po dobu 2 mesiacov. Náš monitorovací softvér bol inštalovaný na mobilné zariadenie (Windows Mobile 6). Takto sme pripravili dáta pozorovaním pohybu skúmanej osoby,

ktoré bežne nosila tento mobilný telefón. Zariadenie automaticky ukladalo polohu v podobe identifikátorov GSM veží a časovej známky v intervaloch 5 minút. Lokalita sa zaznamenáva iba v prípade zmeny od posledného pozorovania.

Našim cieľom bolo použiť tieto dáta na objavovanie rituálov správania používateľa a následne pozorovať hypotézy, ktoré sú s jednotlivými experimentmi opísané v nasledujúcich podkapitolách.

4.1 Objavovanie vzorov správania a čas medzi opakovaniami

Cieľom tohto experimentu je ukázať prítomnosť rituálov v správaní používateľa. Chceme ukázať, že intervaly medzi opakovaniami aktivít majú prirodzenú dĺžku. Predpokladáme, že používatelia periodicky v časových intervaloch navštevujú konkrétne lokality. Znamená to, že používateľ opakuje činnosti na báze hodín, dní, týždňov, mesiacov alebo iných intervalov. Naša metóda je navrhnutá tak, aby sme vedeli hľadať intervaly opakovania ľubovoľnej dĺžky. To však nemusí byť potrebné v prípade ak by sme vedeli dokázať, že človek sa riadi intervalmi prirodzenej dĺžky (dni, týždne) a nie úplne ľubovoľnými intervalmi (napr. 10 hodín). V prípade ak vieme ukázať, že hypotéza platí, potom môžeme brať tieto prirodzené intervaly ako heuristiku pri hľadaní vzorov správania a tak zjednodušiť proces objavovania.

V našom experimente sme objavili viac rituálov. Niektoré boli intenzívnejšie a teda prehlásené za správne (nadkritický počet opakovaní v okne pozorovania). Okrem toho pozorujeme, že často opakujúce sa rituály v kratších intervaloch objavujeme s menšou chybou. Je to spôsobené dĺžkou pozorovania používateľa. V prípade väčšieho pozorovacieho okna by sa zvýšil počet opakovaní i pre dlhšie intervaly a tak spresnilo aj objavovanie. V tabuľke 1 prezentujeme niektoré objavené intervaly a ich atribúty spolu s ich identifikáciou (pomenovaním) s pomocou používateľa.

Tab. 1. Rituály objavené našou metódou (ID 1 až 5 sú spolu v skupine).

ID	Dĺžka periódy	Počet opakovaní	Identifikácia rituálu (superv.)
1-5	~7 dní	6 or 7 (každý)	Práca (Pon-Pia)
6	~7 dní	6	Príchod domov (Piatok)
7	~30 dní	2	Nájom

Príchody do práce nie sú opakujúce sa na báze dní, ale na báze týždňov. To môže byť spôsobné napríklad víkendom, ktorý by opakovanie tohto rituálu narúšal, ale i faktom, že používateľ chodí do práce v jemne odlišných časoch. Najväčšia odchýlka pritom nastáva v pondelok a piatok. Napr. v piatok opúšťa pracovisko skôr ako v iné dni. Niektoré dni dokonca používateľ nebol v práci vôbec, čo však rituál narúša minimálne, pretože na báze týždňa vieme rituál pozorovať ako väčšie množstvo opakovaní v 2 mesačnom pozorovacom okne.

Taktiež sme objavili rituál na mesačnej báze. Tento rituál uvádzame v tabuľke napriek jeho vysokej chybe. Táto chyba je spôsobená tým, že v pozorovacom okne máme iba jedno 2 tieto opakovania. Používateľ však vedel identifikovať tento rituál. Vo väčšom okne potom vieme dosiahnuť presnejší rituál. Napriek tomu, že mesiace má rôzne dĺžky by sme mohli pozorovať tento rituál, jeho presnosť by však bola obmedzená.

Záverom nášho pozorovania je, že nie je nutné hľadať intervaly ľubovoľnej dĺžky, ale držať sa predpokladu, že človek sa riadi cyklami prirodzenej dĺžky (dni, týždne, mesiace). Tým okrem iného vieme zjednodušiť metódu odhaľovania rituálov. Analýza potom pokojne môže byť vykonávaná i na mobilnom zariadení, ktorého výpočtové kapacity sú obmedzené.

4.2 Objavovanie rituálov a vplyv kontextov na rituály

V tomto experimente chceme ukázať, že kontexty ovplyvňujú dĺžku prepravy z miesta na miesto. Pracujeme s kontextami čas a počasie, pričom pozorujeme trasu v kontextoch dvoch lokalít. Jedná sa o cestu do práce a z práce, ktorá má rovnakú dĺžku. Tu pozorujeme vplyv kontextov, keďže tento rituál bol objavený ako najpresnejší s najmenšou odchýlkou. Hypotéza je, že kontexty času a počasia vplývajú na prepravu z miesta na miesto. Čas berieme v úvahu ako časť dňa. Počasie uvažujeme hlavne ako namerané zrážky v tú hodinu dňa. Predpokladáme, že čas sa prejaví v počte áut na ceste. V ranných hodinách je na cestách viac áut, keďže sa ľudia potrebujú prepraviť z domova do práce.

Tabuľka 2 demonštruje koľko trvá presun z miesta na miesto s vplyvom spomenutých kontextov. Pracujeme s rovnakými dátami ako v minulom experimente (2 mesiace, 1 používateľ). Používame zaznamenané časy, aby sme vedeli zistiť počasie a podmienky prepravy. Používame škálu 1 až 3 (dobré), aby sme ohodnotili stav počasia (dôležité sú rozdiely pre dážď žiadny dážď, ľahký dážď, silný dážď a hmlu `www.wunderground.com`). Navrhnutou metódou sme odhalili, že čas tak ako aj počasie, vplývajú na dĺžku prepravy.

Tab. 2. Dĺžka prepravy je ovplyvnená rôznymi časom a počasím.

čas / počasie	1	2	3
7:30 – 8:00	21 min.	23 min.	15 min.
8:00 – 8:30	10 min.	9 min.	11 min.
13:00 – 13:30	9 min.	10 min.	8 min.

5 Záver

V článku sme prezentovali metódu pre adaptívne pripomienkovanie udalostí. Hlavná myšlienka je pripomínaní kritického času, v ktorom už musí používateľ opustiť lokalitu aby stihol udalosť predpovedanú v inej lokalite. Používateľ tak nemešká na stretnutia a je upozorňovaní len v prípade keď je to nevyhnutné pre plnenie plánu. Okrem toho zvažujeme zakomponovanie ďalších kontextov, ktoré vplývajú na presun medzi miestami. Zistili sme totiž, že kontexty priamo vplývajú na rituály, a preto má význam adaptovať sa týmto spôsobom. Predpoveď správania pomocou modelu používateľa tak môže byť kvalitnejšia.

Vytvorili sme aplikáciu, ktorá monitoruje používateľov prostredníctvom mobilných zariadení, ktoré títo používateľa neustále nosia so sebou. Cieľom monitorovania je získať množinu záznamov o pohybe používateľa v čase, ktoré potom využívame na ďalšie analýzy. Našim zámerom bolo navrhnúť metódu pre objavovanie rituálov používateľa a experimentovať v doméne pripomienkovania udalostí. Experimentovali sme s hypotézami, ktoré sa opierajú o bežnú logiku ako napr. vyšší nárok na čas pre prepravu v čase rannej špičky alebo zlého počasia. Tieto experimenty sme vykonávali na snímke dát o aktivitách jedného používateľa po dobu dvoch mesiacov.

V ďalšej práci plánujeme zohľadniť viac kontextov, ktoré vplývajú na aktivity používateľa (kultúra, demografické atribúty, nálada a podobne). Na lokalite, čase a počasí predvádzame vplyv kontextov na bežnú úlohu prepravy z miesta na miesto. Úloha prepravy medzi dvoma miestami pritom prináša záznamy o časových vzdialenostiach medzi dvoma lokalitami v určitej oblasti. Takto nám postupne v súlade s vplyvom kontextov vzniká mapa s časmi potrebnými na prepravu za určitých okolností. Túto mapu považujeme za významný prínos v hľadaní vplyvov kontextov na vzory správania, môže vynikať i kolaboratívne. Výstup plánujeme integrovať s AdaptiveProxy [3], čo nám umožní rozšíriť možnosti získavania kontextov používateľa a prinesie miesto pre ich využitie.

Poďakovanie

Táto publikácia vznikla vďaka čiastočnej podpore projektov VG1/0675/11/2011-2014, KEGA 028-025STU-4/2010, APVV-0208-10 a projektu v rámci OP Výskum a vývoj pre projekt: Výskum metód získavania, analýzy a personalizovaného poskytovania informácií a znalostí, ITMS: 26240220039, spolufinancovaný zo zdrojov Európskeho fondu regionálneho rozvoja.

Literatúra

1. Baltrunas, L., Kaminskas, M., Ricci, F., Rokach, L., Shapira, B., and Luke, K.H. Best Usage Context Prediction for Music Tracks. ids.csom.umn.edu.
2. Bamis, A., Fang, J., and Savvides, A. A Method for Discovering Components of Human Rituals from Streams of Sensor Data. Work, (2010), 779-788.
3. Barla, M., Bieliková M. Ordinary Web Pages as a Source for Metadata Acquisition for Open Corpus User Modeling. In Proc. of IADIS WWW/Internet 2010. IADIS Press, 2010. 227-233.
4. Cebrián, T., Planagumà, M., Villegas, P., and Amatriain, X. Music recommendations with temporal context awareness. Proc. of the 4th conf. on RecSys. (2010), 349–352.
5. Ciglan, M. and Nørvåg, K. WikiPop: personalized event detection system based on Wikipedia page view statistics. Proceedings of the 19th ACM international conference on Information and knowledge management, ACM (2010), 1931–1932.
6. Clarke, S.,Driver, C. Context-aware trails mobile computing. Computer, (2004), 97-99.
7. Coppola, P., Della Mea, V., Di Gaspero, L., et al. Context-Aware Browser. IEEE Intelligent Systems, (2009).
8. Halvey, M., Keane, M., and Smyth, B. Predicting navigation patterns on the mobile-internet using time of the week. Special interest tracks and posters of the 14th international conference on World Wide Web, ACM (2005), 958–959.
9. Hsu, S., Wen, M.H., Lin, H.C., Lee, C.C., and Lee, C.-hoang. AIMED-a personalized TV recommendation system. Interactive TV: a Shared Experience, (2007), 166–174.
10. Mei, Q., Zhou, D., and Church, K. Query suggestion using hitting time. Proceeding of the 17th ACM conf. on Information and knowledge mining - CIKM '08, (2008), 469.
11. Oku, K., Nakajima, S., Miyazaki, J., Uemura, S., Kato, H., and Hattori, F. A Recommendation System Considering Users' Past/Current/Future Contexts. ids.csom.umn.edu, (2010), 3-7.
12. Vo, C., Torabi, T., and Loke, S.W. Towards context-aware task recommendation. Pervasive Computing (JCPC), 2009 Joint Conferences on, IEEE (2010), 289–292.

Annotation:

Adaptive Event Reminder: Context Impact on Behavioural Patterns

Our method is designed to schedule events adaptively. We use a mobile device to track user activities and discover patterns in her activities by analyzing locations where she spends time, type of transportation she uses and weather conditions. These observed contexts are used to remind when to leave, when to wake up, etc. Our idea is to help organise activities of the user and adapt.

Realizace omezení pro násobnosti vztahů mezi entitami v relačních databázích*

Zdeněk RYBOLA[1], Karel RICHTA[2,3]

[1]*Katedra softwarového inženýrství, FIT ČVUT v Praze*
Thákurova 9, 160 00 Praha
rybolzde@fit.cvut.cz
[2]*Katedra počítačů, FEL ČVUT v Praze*
Technická 2, 160 00 Praha
richta@fel.cvut.cz
[3]*Katedra softwarového inženýrství, MFF UK v Praze*
Malostranské nám. 25, 118 00 Praha
richta@ksi.mff.cuni.cz

Abstrakt. Modelem řízený vývoj (Model Driven Development, MDD) prosazuje myšlenku tvorby řady modelů na různé úrovni abstrakce – výpočetně nezávislý model (CIM), platformově nezávislý model (PIM), platformově specifický model (PSM) a implementačně specifický model (ISM). Důležitou součástí MDD jsou transformace mezi jednotlivými modely – v dopředném směru od obecnějšího modelu ke konkrétnějšímu, i ve zpětném směru. To umožňuje automatizování vývoje ve formě generování kódu nebo validaci analytických a návrhových modelů.

Při transformaci binárních vztahů z PIM do PSM je třeba uvažovat omezení daná násobností těchto vztahů. Tento příspěvek se zabývá definicí omezení pro násobnosti definované v PIM a způsoby jejich realizace na úrovni PSM v relačních databázích.

Klíčová slova: modelem řízený vývoj, transformace vztahů, násobnosti vazeb, MDD, OCL.

1 Úvod

Modelem řízený vývoj (Model Driven Development, MDD) [5][8] je jedním z cílů Object Management Group (OMG). Principem MDD je maximální využití modelů na různé úrovni abstrakce – od výpočetně nezávislých modelů (Computational Independent Model, CIM), přes platformově nezávislé modely (Platform Independent Model, PIM) a platformově závislé modely (Platform Specific Model, PSM) po implementačně závislý model (Implementation Specific Model, PSM). Nedílnou součástí MDD jsou transformace mezi jednotlivými modely.

Samotný proces vývoje software se skládá z dopředného inženýrství (forward engineering), kdy se z obecnějších modelů transformují modely konkrétnější až po samotný zdrojový kód, a zpětného inženýrství (reverse engineering), kde se provádí transformace od více specifických modelů a zdrojového kódu k obecnějším modelům.

* Tento příspěvek byl podpořen grantovým projektem Studentské grantové soutěže číslo SGS11/087/OHK3/1T/18 a grantem Grantové agentury České republiky (GAČR) číslo GA201/09/0990.

Snem OMG je tzv. vývoj v kruhu (round-trip engineering) – kombinace dopředného a zpětného inženýrství.

Pro definici a popis modelů se používá nejčastěji skupinou OMG vyvíjený jazyk UML [1][7][8]. Tento jazyk umožňuje vytvořit grafickou podobu různých typů modelů – od statických datových modelů přes dynamické modely aktivit a stavové modely po komunikační a komponentové modely. V tomto příspěvku se dále zabýváme pouze statickými datovými modely, které jsou tvořeny diagramy tříd a jejich popisem.

Nezbytnou součástí nástroje pro popis modelů musí být také nástroj pro popis omezení modelu. V případě jazyka UML je tímto nástrojem jazyk OCL [6][8][9]. V jazyce OCL je možné definovat omezení (constraints) v kontextu jednotlivých elementů modelu vytvořeného v jazyce UML formou invariantů, které specifikují podmínky, které musí každá instance daného elementu splňovat. Pomocí jazyka OCL lze také definovat vstupní a výstupní podmínky metod nebo aktivit.

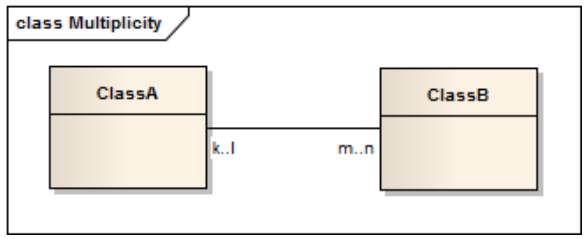
Nejčastěji používaným procesem v rámci MDD, ale i mimo něj, je tvorba PIM pro popis datového modelu aplikace a jeho následná transformace do PSM za účelem generování zdrojového kódu – nejčastěji pro relační databáze. Tento příspěvek se zabývá transformací binárních vztahů z PIM do PSM pro relační databáze s ohledem na omezení daná specifikovanými násobnostmi vztahů. Omezení pro násobnosti jsou definována na úrovni PIM formou OCL omezení (contraints), čímž jsou platformově nezávislá a mohou být transformována různými nástroji podporujícími OCL i jiným způsobem, než jsou námi navržené způsoby, nebo pro jiné platformy než jsou relační databáze. V příspěvku jsou dále definovány možnosti pro realizaci stanovených omezení po transformaci na úrovni PSM v jazyce SQL, který v případě relačních databází slouží jako platformově specifický jazyk.

Důvod a princip omezení daných minimálními a maximálními násobnostmi vazeb v PIM modelu tříd je podrobně popsán v [12]. Tento článek shrnuje základní princip a zaměřuje se na různé možnosti realizace těchto omezení v relačních databázích.

Příspěvek je dále členěn následovně: v sekci 2 definujeme binární vztah a násobnosti jeho vazeb; v sekci 3 popisujeme transformaci binárního vztahu z PIM do PSM společně s popisem realizace základních omezení daných násobnostmi vazeb. V sekci 4 shrnujeme běžné postupy a poznatky z existujících nástrojů pro modelování a transformace. V sekci 5 definujeme možnosti realizace dodatečných omezení nutných pro zajištění některých případů násobností vazeb. Sekce 6 obsahuje shrnutí příspěvku.

2 Binární vztah a jeho násobnosti

Konceptuální model (PIM) [1][5] využívá entity nebo třídy pro popis typů objektů a asociace mezi těmito entitami nebo třídami pro vyjádření vztahů mezi objekty. Tyto vztahy mohou být unární (atributy objektů), binární (vztah mezi dvěma objekty) nebo n-ární (vztah mezi n objekty). Bylo však již dokázáno, že jakýkoli n-ární vztah může být nahrazen (n-1) binárními vztahy, proto se dále budeme zabývat pouze binárními vztahy [11].



Obr. 1: Značení násobností vazeb binárního vztahu v grafické podobě pomocí UML diagramu tříd

Binární vztah je tedy vazba mezi právě dvěma entitami znamenající, že instance těchto entit mají mezi sebou jistý vztah. Vlastnosti tohoto vztahu jsou zčásti dané násobnostmi vztahu. Na Obr. 1 je znázorněno značení násobností vazby mezi dvěma entitami v grafické podobě za pomoci diagramu tříd v jazyce UML. Násobnosti jsou na něm označeny písmeny *k*, *l*, *m* a *n* s následujícím významem:

- *k* představuje minimální počet instancí ClassA ve vztahu k jedné instanci ClassB,
- *l* představuje maximální počet instancí ClassA ve vztahu k jedné instanci ClassB,
- *m* představuje minimální počet instancí ClassB ve vztahu k jedné instanci ClassA,
- *n* představuje maximální počet instancí ClassB ve vztahu k jedné instanci ClassA.

Minimální násobnost určuje povinnost instance dané entity k účasti ve vztahu. Říká tedy, zda každá instance musí mít vztah k některé instanci vztažené entity (minimální násobnost rovna 1) nebo nemusí (minimální násobnost rovna 0). Povinnost účasti ve vztahu je omezení a může být vyjádřeno pomocí jazyka OCL následovně:

```
context a:ClassA inv minBdirect: not a.b.asSet()->isEmpty()        (1)
```

Jelikož při realizaci vztahu se často jedná o jednosměrnou vazbu, jako v případě relační databáze s použitím cizích klíčů, je vhodné toto omezení definovat v opačném směru takto:

```
context a:ClassA inv minBreverse: ClassB.allInstances()->exists(b | b.a = a).
                                                                    (2)
```

Maximální násobnost určuje nejvyšší počet instancí vztažené entity, se kterými může být jedna instance spojená. Maximální násobnost rovna 1 znamená, že se jedná o vztah maximálně k jedné instanci. Naopak maximální násobnost označená znakem * znamená, že instance může mít vazbu na množinu instancí druhé entity, přičemž velikost této množiny není specifikovaná. Maximální násobnost rovna 1 je tedy omezením a může být opět vyjádřeno pomocí OCL následovně:

```
context a:ClassA inv maxBdirect: a.b.asSet()->size() <= 1          (3)
```

nebo v opačném směru takto:

```
context a:ClassA inv maxBreverse: ClassB.allInstances()->count(b|b.a=a) <= 1.
                                                                    (4)
```

3 Transformace binárních vztahů z PIM do PSM pro relační databáze

V relačních databázích jsou entity a třídy reprezentovány tabulkami, jejichž řádky představují jednotlivé objekty daného typu [9][11]. Každý takový objekt je jednoznačně identifikován mechanismem zvaným primární klíč (primary key). Vazby mezi objekty jsou realizovány pomocí mechanismu tzv. cizích klíčů (foreign keys) – jedná se o hodnoty, které odkazují na řádky vztažené tabulky pomocí jejich primárních klíčů. Omezení definované nad cizím klíčem (foreign key contraints) zajišťují automatickou kontrolu existence odkazovaných řádků a události vyvolané změnou či smazáním odkazovaného řádku ze vztažené tabulky.

Tento běžně používaný mechanismus cizích a primárních klíčů však na úrovni PSM přináší dva zásadní problémy vzhledem ke vztahům definovaným v PIM. Cizí klíč reprezentující vazbu objektu na jiný objekt je v databázi uložen jako další sloupeček tabulky. Takový klíč však může vždy odkazovat pouze na jeden jediný řádek v cílové tabulce, a proto může realizovat pouze vztahy s maximálními násobnostmi 1-1 a M-1 (znakem *M* nebo *N* bývá při popisu násobností vztahů reprezentováno nespecifikované množství, které může být vyšší než 1) – na rozdíl od vztahů definovaných v PIM, kde mohou být modelovány i vztahy M-N. Takové vztahy proto při transformaci do PSM pro relační databáze musí být nejprve dekomponovány na dva vztahy M-1 a pomocnou tabulku reprezentující samotný vztah a teprve poté realizovány pomocí cizích klíčů.

Druhým problémem je to, že vazba pomocí cizího klíče je pouze jednosměrná. To znamená, že objekt uložený v tabulce s cizím klíčem má informaci o objektu, se kterým je ve vztahu (zná jeho primární klíč), zatímco tento vztažený objekt žádnou informaci o svých vazbách na druhé objekty nemá. Aby je mohl získat, musí se provést dotaz do vztažené tabulky a hledat záznamy, které odkazují na daný objekt. Toto je opět v rozporu s PIM, kde se vztahy mohou modelovat obousměrné – a také se tak na této úrovni při modelování konceptuálního modelu děje.

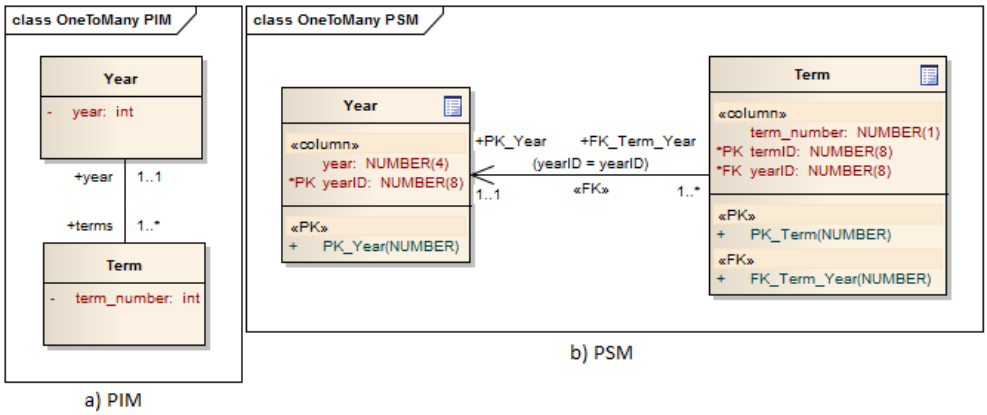
3.1 Transformace omezení pro násobnosti vazeb binárních vztahů

Při transformaci vztahů je nejdříve třeba rozhodnout o směru realizace vazby – tedy o tom, která tabulka (zdrojová) bude obsahovat cizí klíč odkazující do druhé tabulky (cílové). To je však efektivně možné až při znalosti omezení pro násobnosti vztahů a možností jejich realizace.

Jak již bylo zmíněno, pomocí cizího klíče lze vždy odkazovat pouze na jeden řádek z cílové tabulky. Proto v případě realizace vztahu M-1 je nutné použít cizí klíč v tabulce reprezentující entitu s nespecifikovanou násobností M (příp. *) odkazující do tabulky s násobností 1. Příklad je uveden na Obr. 2.

Pro zaručení minimální násobnosti pomocí cizího klíče máme následující možnosti [4][11]: V případě povinné účasti objektu v cílové tabulce ve vztahu (násobnost $k = 1$ na Obr. 2), je možné definovat integritní omezení *not null* pro cizí klíč. To zaručí, že vždy musí být vyplněna hodnota cizího klíče a objekt tedy bude vždy odkazovat na nějaký cílový objekt. Naopak v případě, kdy tato vazba není povinná, integritní omezení *not null* nedefinujeme, čímž umožníme vložení hodnoty *null*, která reprezentuje žádnou vazbu.

Při znalosti těchto možností můžeme rozhodnout o směru vazby pro vztahy 1-1. V případě povinné účasti jedné strany budeme směrovat cizí klíč z tabulky nepovinné entity odkazující na řádek v tabulce povinné entity, abychom mohli definovat *not null* integritní omezení pro tento cizí klíč. V případech, kdy obě entity jsou ve vztahu nepovinně nebo obě povinně, směr realizace cizího klíče může být libovolný, a může být například použita orientace použitá v PIM nebo vybrána náhodně.



Obr. 2: Příklad vztahu M-1 s povinnou účastí obou stran

Při použití cizího klíče není maximální násobnost na zdrojové straně nijak omezena – libovolné množství objektů ze zdrojové tabulky může odkazovat na stejný objekt v cílové tabulce. Pokud je maximální zdrojová násobnost omezena na 1, cizí klíč může být definován s integritním omezením *unique* [4][11]. V takovém případě omezení zajistí, že hodnota cizího klíče je v rámci tabulky unikátní, a tudíž maximálně jeden řádek může odkazovat na stejný řádek cílové tabulky.

Komplikace nastávají v situaci, kdy minimální násobnost na straně zdrojové tabulky, kde je umístěn cizí klíč, je rovna jedné. Jedná se například o vztah M-1, kde každý objekt entity s maximální násobností 1 musí mít vazbu alespoň na jeden objekt entity s nespecifikovanou násobností (situace na Obr. 2) nebo v případě vztahu 1-1, kde obě strany mají minimální násobnost rovnu 1 (situace na Obr. 3). Mechanismus cizího klíče neumožňuje zajistit toto omezení, je proto nutné realizovat omezení pro maximální násobnost definované v sekci 2.

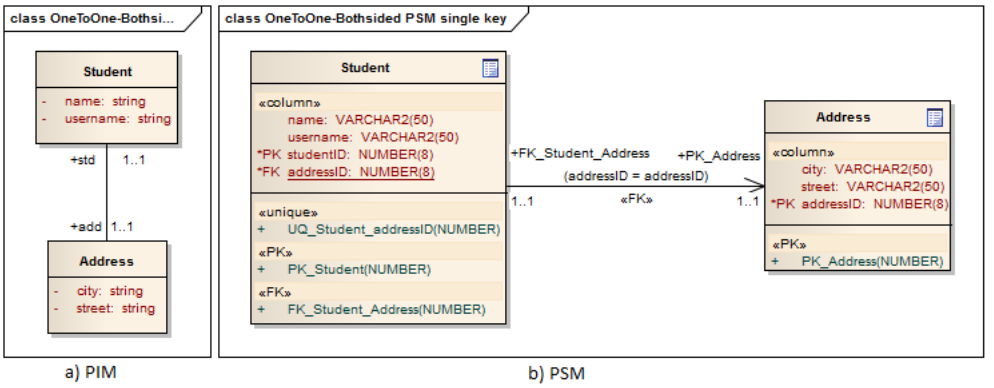
Způsoby realizace tohoto dodatečného omezení se dále zabýváme v sekci 5.

4 Existující řešení a nástroje

Problém transformace modelu tříd z PIM do PSM pro relační databáze je znám již dlouhou dobu a existují zažité postupy, jakým způsobem vztahy transformovat – např v [11].

V běžných situacích vztahů 1-1, 1-N a M-N řešení odpovídají výše zmíněným způsobům transformace formou cizích klíčů a integritních omezení pro ně definovaných. Situace vztahu 1-1, kde je účast obou stran vztahu povinná, je však řešena odlišně. V tomto případě se většinou vztah realizuje sloučením vztažených entit *A* a *B* do jedné entity *C* a uložením do jedné databázové tabulky (tento způsob se někdy používá i v případě, kdy nejsou obě strany povinné). Toto řešení je ale z našeho pohledu nevyhovující hned v několika bodech:

- vztah jedné ze sloučených entit *A* nebo *B* s další entitou *E* se stává vztahem obou entit k dané entitě *E,*
- realizace neodpovídá záměru autora pro dvě samostatné entity,
- dlouhé řádky s *null* hodnotami v případě nepovinnosti jedné nebo druhé strany,
- složité podmínky pro nenulovost hodnot pouze jedné z původních entit *A* či *B*.



Obr. 3: Příklad vztahu 1-1 s povinnou účastí obou vztažených entit

Zajištění minimální násobnosti v případě povinnosti na straně entity N ve vztahu N-1 je pak většinou zanedbáno a ponecháno čistě na aplikaci.

Existuje celá řada nástrojů pro modelování v UML s podporou OCL, které se používají pro modelování a generování zdrojového kódu. Jako příklad uvádíme freeware nástroj Dresden OCL Toolkit a komerční nástroj Enterprise Architect.

Dresden OCL Toolkit [2][3] je nástroj umožňující načíst model v UML společně s omezeními v jazyce OCL a jejich validaci a interpretaci, stejně jako generování zdrojového kódu v SQL nebo Javě. OCL omezení jsou převedena databázové pohledy záznamy porušující dané omezení. Dresden OCL Toolkit je mimo jiné distribuován jako samostatná knihovna, plugin do Eclipse IDE a je také integrován do UML nástroje ArgoUML. Při generování zdrojového kódu v SQL z PIM se však nebere v potaz minimální násobnost vazeb mezi třídami, vygenerované cizí klíče nejsou omezené a nezaručují tak dodržení stanovených násobností.

Další nástroj používaný pro modelování v UML je Enterprise Architect [13]. Tento nástroj umožňuje vygenerovat PSM pro relační databázi z PIM modelu tříd a následně také generovat zdrojový kód v SQL. Ani Enterprise Architect však nebere při transformaci v potaz minimální násobnosti vztahů a nevytváří pro cizí klíč žádná integritní omezení. Dodržení stanovených násobností tak není zaručené.

Z těchto důvodů jsme se rozhodli zdůraznit tato omezení daná minimální násobností vztahů a ukázat způsob jejich reprezentace na úrovni PIM a realizace na úrovni PSM.

5 Realizace dodatečných omezení pro minimální násobnost

V případě povinné účasti zdrojové entity (entita obsahující cizí klíč v tabulce) ve vztahu k cílově entitě na úrovni PIM, je nutné kromě možností samotného cizího klíče na úrovni PSM také realizovat speciální dodatečné omezení dle (2). Pro případ na Obr. 3, kde je použit směr cizího klíče z tabulky Student odkazující do tabulky Address, toto omezení vypadá následovně:

```
context a:Address inv studentExist: Student.allInstances()->exist(s|s.add=s)
```

Toto omezení může být realizováno několika různými způsoby, které jsou dále rozvedeny v následujících podsekcích, kde se omezujeme na popis případu vztahu 1-1 na Obr. 3. Situace pro vztah M-1 na Obr. 2 by byl však řešen analogicky.

5.1 Databázové pohledy

Základní možností realizace zmíněného omezení je definice databázového pohledu [4][9]. Tento pohled vyhledává záznamy uložené v řádcích tabulky Address, které porušují definované omezení – tedy takové objekty, na které neodkazuje žádný objekt v tabulce Student. Tento pohled je možné definovat následujícím způsobem:

```
CREATE VIEW violating_address AS
SELECT a.addressID FROM Address a WHERE NOT EXISTS
(SELECT 1 FROM Student s WHERE s.addressID = a.addressID);
```

Pokud však omezení realizujeme pouze takovým pohledem, jedná se vlastně pouze o způsob detekce porušení stanovené násobnosti. Aplikace sama musí ve zvolené okamžiky kontrolovat jejich porušení – pokud pohled nevrací žádné záznamy, je databáze konzistentní; pokud pohled obsahuje nějaké záznamy, jedná se o záznamy porušující stanovené omezení minimální násobnosti. Aplikace pak musí sama provést potřebné kroky k nápravě – ať už dodatečné vložení vztaženého záznamu do druhé tabulky, smazání záznamu porušujícího omezení nebo jeho ignorováním při práci s daty.

5.2 Databázové omezení CHECK

Další možností kontroly dodatečných omezení pro minimální násobnost vztahů je databázové integritní omezení *CHECK* (check constraint). Toto omezení se definuje při definici tabulky a používá se k omezení hodnot vkládaných do sloupečků tabulky – např. pro omezení rozsahu nebo výčtové hodnoty.

Podle specifikace SQL:1999 [4] je možné omezení *CHECK* definovat pomocí vnořeného *SELECTu*, který vybere možné nebo naopak nepovolené hodnoty pro daný sloupeček. Omezení pro náš případ na Obr. 3 bychom tedy mohli definovat nad primárním klíčem tabulky Address, na který musí odkazovat alespoň jeden cizí klíč z tabulky Student:

```
ALTER TABLE Address ADD CONSTRAINT address_check
CHECK (addressID = ANY (SELECT addressID FROM Student))
```

Toto omezení je kontrolováno vždy v okamžiku, kdy dojde ke změně dat, nad nimiž je omezení definováno – jde tedy o operace *insert* a *update* nad tabulkou Address a *insert*, *update* a *delete* nad tabulkou Student.

Samotné omezení definované tímto způsobem spolu s definovaným cizím klíčem nám však neumožní vkládání nových vztažených objektů. Do tabulky Student nemůžeme vkládat záznamy jako první kvůli definovanému cizím klíči, protože potřebujeme, aby odkazovaný záznam v tabulce Address již existoval, a do tabulky Address také nemůžeme vkládat jako první, protože při kontrole omezení *CHECK* potřebujeme, aby existoval nějaký záznam v tabulce Student, který na vkládaný záznam odkazuje.

Naštěstí nám standard SQL:1999 poskytuje možné řešení: při definici integritního omezení včetně cizích klíčů (foreign key contraints) a omezení *CHECK* je možné omezení definovat tzv. *deferrable* (odložitelné) [4]. Pokud je omezení nastaveno jako *deferred* (odložené), jeho kontrola se provádí až na konci celé transakce, nikoli po provedení každé DML operace, která nějak ovlivní data, nad nimiž je omezení definováno. Omezení lze

nastavit *deferred* automaticky od začátku každé transakce jako v následujícím příkladu, nebo pomocí příkazu *SET CONSTRAINTS* v kdekoli v průběhu transakce.

K dispozici teď máme dvě možnosti – odkládat můžeme kontrolu omezení *CHECK* nad tabulkou Address nebo kontrolu integrity cizího klíče nad tabulkou Student. V prvním případě by bylo nutné omezení pro případ na Obr. 3 definovat takto:

```
ALTER TABLE Address ADD CONSTRAINT address_check
CHECK (addressID = ANY (SELECT addressID FROM Student))
DEFERRABLE INITIALLY DEFERRED;
```

S takovým omezením můžeme nejprve vložit záznam do tabulky Address, aniž by existoval student na dané adrese žijící, a teprve potom studenta. V druhém případě by definice cizího klíče vypadala následovně:

```
ALTER TABLE Student ADD CONSTRAINT student_address
FOREIGN KEY (addressID) REFERENCES Address
DEFERRABLE INITIALLY DEFERRED;
```

V tomto případě můžeme v rámci transakce nejprve vložit záznamy do tabulky Student obsahující cizí klíč a až následně vložit odkazované záznamy do cílové tabulky Address, přičemž dojde ke kontrole omezení *CHECK*.

Dojde-li však k porušení jakéhokoli odloženého omezení při kontrole na konci transakce, databázový systém nedokáže vrátit efekt jednoho příkazu, který toto porušení způsobil, a musí vrátit celou transakci [4].

Dalším velkým nedostatkem tohoto způsobu je fakt, že většina dodavatelů databázových systémů, včetně Oracle, nepodporuje vnořené dotazy uvnitř definice omezení typu *CHECK*, takže ve většině případů nelze tento způsob použít.

5.3 Triggery

Kontrolu splnění podmínek dodatečných omezení minimálních násobností vztahů lze také provádět pomocí *triggerů* (např. v Oracle). Triggery [1][4] jsou speciální procedury, které jsou připojené k některým typům událostí nad tabulkou – trigger je možné navázat na operace *insert*, *update* nebo *delete* z dané tabulky. Navíc je možné specifikovat, zda se bude *trigger* spouštět před samotnou operací, která vyvolání způsobila, nebo až po jejím provedení.

V případě realizace omezení pro minimální násobnost je třeba definovat *trigger* pro operace *insert* a *update* nad tabulkou Address, protože je třeba kontrolovat existenci odkazujícího záznamu v tabulce Student při vkládání nové adresy nebo při změně primárního klíče adresy. Dále je třeba definovat podobný *trigger* nad tabulku Student pro operace *update* a *delete*, kdy může dojít k porušení integrity při odstranění posledního záznamu v tabulce Student, který odkazoval na jeden záznam v tabulce Address.

Samotný *trigger* pak provádí kontrolu existence záznamu v tabulce Student, který odkazuje na právě vkládaný záznam do tabulky Address. Pokud *trigger* žádný takový záznam nenajde, detekoval porušení daného omezení a vyvolá výjimku, kterou může aplikace odchytit a zachovat se podle toho.

Jelikož je *trigger* spouštěn vždy po nebo před provedením příkazu nad danou tabulkou, není možné zajistit vložení do obou vztažených tabulek najednou a pak provést kontrolu omezení. Stejně jako v případě omezení formou *CHECK* nelze vkládat napřed do tabulky

Student kvůli kontrole cizího klíče, ani do tabulky Address kvůli kontrole v *triggeru*, který zatím nenajde odkazující záznamy.

Z tohoto důvodu je nutné opět tento model částečně upravit. Opět se nám nabízejí dvě možnosti. První z nich je odložení kontroly cizích klíčů jako v případě omezení *CHECK*. To nám opět umožní prvně vložení záznamů do tabulky Student s neplatným odkazem na adresy a až následně vložení do tabulky Address, kdy dojde ke kontrole omezení pomocí *triggeru*, které bude splněné.

Druhou možností je úprava chování *triggeru*. Takový *trigger* by byl spouštěn před operací vyvolávající jeho spuštění a kontroloval by stav před provedením operace. Jeho náplní by nebylo vyhledávání existence záznamů, které odkazují na právě manipulovaný záznam v tabulce Address, ale kontroloval by konzistenci databáze. V případě, že by databáze byla konzistentní – tj. všechny objekty uložené v řádcích tabulek splňují kontrolované omezení – *trigger* by umožnil vložení záznamu, který tuto konzistenci poruší. Při následném vyhodnocování další podobné operace by však při kontrole konzistence narazil na její porušení a operaci by zabránil. Tím by bylo zajištěno, že sice mohu vložit záznam do tabulky Address, na který neodkazuje žádný záznam z tabulky Student, ale než tam pak budu moci vložit další, musím nejprve napravit porušení omezení pro minimální násobnost a vložit odkazující záznam do tabulky Student. Takový *trigger* by musel být definován *FOR EACH ROW* (pro každý řádek). Pro kontrolu konzistence by mohl být využit pohled definovaný v sekci 5.1 – pokud pohled neobsahuje žádné záznamy, je databáze z pohledu daného omezení konzistentní, pokud pohled nějaký záznam obsahuje, jedná se o záznam porušující dané omezení.

6 Závěr

V tomto příspěvku jsme popsali omezení související s minimálními a maximálními násobnostmi binárních vztahů modelovaných v PIM. Dále jsme popsali způsob transformace binárních vztahů do PSM pro relační databáze a ukázali jsme si možnosti realizace běžných omezení pro násobnosti pomocí omezení pro cizí klíče realizující vazbu.

Ukázali jsme situace, při kterých si nevystačíme pouze s možnostmi cizích klíčů, a definovali jsme dodatečné omezení, které je třeba v PSM realizovat, aby byly zajištěny minimální násobnosti definované v PIM. V sekci 5 jsme si ukázali možnosti, jak lze tato dodatečná omezení realizovat v relační databázi – pomocí databázových pohledů, omezení *CHECK* a za pomoci *triggerů*. Pro všechny tři možnosti jsme popsali princip včetně úskalí, která jsou s danou realizací spojená.

Literatura

[1] Arlow, J., Neustadt, I.: UML 2.0 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition). Addison-Wesley Professional, 2005

[2] Demuth, B.: The Dresden OCL Toolkit and its Role in Information Systems Development. In: *13th International Conference on Information Systems Development* (2004)

[3] Demuth, B.: DresdenOCL. http://www.reuseware.org/index.php/DresdenOCL (Jul 2011)

[4] Melton, J., Simon, A. R.: *SQL:1999, Understanding Relational Language Components*, Morgan Kaufmann Publishers, 2002

[5] OMG, Miller, J., Mukerji, J.: *MDA guide version 1.0.1*. http://www.omg.org/cgibin/doc?omg/03-06-01.pdf (Jun 2003)

[6] OMG: *Object constraint language, version 1.3*. http://www.omg.org/spec/OCL/2.2/PDF (Feb 2010)

[7] OMG: *UML 2.3*. http://www.omg.org/spec/UML/2.3/ (Feb 2011)

[8] Richta, K.: Jazyk OCL a modelem řízený vývoj. In: *Moderní databáze – Architektura moderních IS 2010* (2010)

[9] Richta, K.: Rekonstrukce OCL z SQL. In: *Datakon 2010* (2010)

[10] Richters, M., Gogolla, M.: OCL: syntax, semantics, and tools. In: *Object Modeling with the OCL, The Rationale behind the Object Constraint Language*. pp. 42-68. Springer-Verlag (2002)

[11] Rob, P., Coronel, C.: *Database Systems: Design, Implementation, and management*. Boyd & Fraser, 2nd edn. 1995

[12] Rybola, Z., Richta, K.: Transformation of Binary Relationships with Particular Multiplicity. In: *DATESO 2011* (2011)

[13] Sparx Systems: Enterprise architect - UML design tools and UML CASE tools for software development. http://www.sparxsystems.com.au/products/ea/index.html (Mar 2011)

Annotation:

Realization of Constraints for Relationship Multiplicities in Relational Databases

Model Driven Development (MDD) is a dream of Object Management Group (OMG). It defines several models of different abstraction level ranging from Computational Independent Model (CIM), to Platform Independent Model (PIM), to Platform Specific Model (PSM), to Implementation Specific Model (ISM). MDD also defines forward and reverse engineering process for developing software. Necessary part of such process are transformations between models.

This paper deals with binary relationships modelled in PIM and their transformation to PSM for relational databases. We define constraints for minimal and maximal multiplicities of such a relationship and present options for their realization in SQL for PSM of relational database using foreign keys and additional constraint for required minimal multiplicity of source entity using views, check constraints and triggers.

Zoskupovanie novinových článkov podľa udalostí

Michal HOLUB, Mária BIELIKOVÁ

Ústav informatiky a softvérového inžinierstva, Fakulta informatiky a informačných technológií, Slovenská technická univerzita v Bratislave
Ilkovičova 3, 842 16 Bratislava
{holub,bielik}@fiit.stuba.sk

Abstrakt. V informačných portáloch na webe môžeme nájsť množstvo informácií. Problémom je, že sú roztrúsené. Vyhľadávače nám pomáhajú nájsť také dokumenty, ktoré obsahujú zadané kľúčové slová, avšak nedokážu získať a prezentovať vzťahy medzi jednotlivými zdrojmi informácií a agregovať ich. Riešením je prepojenie informácií z rôznych zdrojov a ich jednotné prezentovanie používateľovi. V tomto príspevku opisujeme metódu integrácie novinových článkov do skupín podľa spoločnej udalosti, o ktorej informujú. Metóda je založená na extrakcii kľúčových z článkov. Novinové články získavame z rôznych slovenských portálov. Článok hneď po jeho publikovaní zaradíme do príslušnej skupiny k ostatným článkom z iných zdrojov, ktoré sa venujú tej istej udalosti v daný deň. Navrhnutú metódu používame na personalizovanom novinovom portáli, ktorý používateľovi prezentuje aktuálne dianie v podobe udalostí a odkazov na články o nich.

Kľúčové slová: integrácia informácií, novinové články, udalosti, spracovanie textu, podobnosť, zoskupovanie.

1 Úvod

Web dnes obsahuje veľkú časť poznatkov ľudstva publikovaných vo forme webových stránok, ktoré sú časťami webových portálov. Používatelia v tejto informačnej báze stále častejšie hľadajú odpovede na svoje informačné potreby. Problémom pri hľadaní vhodných informácií je skutočnosť, že žiadny webový informačný portál neobsahuje všetky dostupné informácie, ani len zo svojej tematickej kategórie. Navyše, nájsť informačný portál pre aktuálnu potrebu nie je tiež spravidla možné vzhľadom na množstvo informácií, priamo. Pre zodpovedanie informačnej potreby preto používame vyhľadávače na nájdenie dokumentov, v ktorých sa odpoveď pravdepodobne nachádza. Odpoveď však môže pozostávať z viacerých menších častí, ktoré sú roztrúsené po mnohých portáloch, a ktoré tak musíme agregovať.

Vyhľadávače dokážu efektívne nájsť dokumenty obsahujúce kľúčové slová z dopytu zadaného používateľom. Avšak nedokážu prezentovať súvislosti, usporadúvajú dokumenty na základe miery súhlasu s dopytom. Používateľ musí ručne prejsť množinu vrátených dokumentov, nájsť v nich požadované informácie a spojiť ich dokopy, aby získal kompletnú odpoveď alebo získal širší prehľad určitej oblasti. Tento problém môžeme vyriešiť automatickou integráciou informácií z viacerých zdrojov [11].

Ďalším problémom klasických vyhľadávačov je, že používateľ musí vedieť sformulovať dopyt. Sú však situácie, kedy presne nevieme, čo hľadáme, prípadne to nevieme presne opísať. Je preto ťažké zadať presný dopyt. To nastáva, keď chceme získať prehľad nejakej oblasti, avšak nehľadáme konkrétnu informáciu. Príkladom môžu byť aktuálne svetové

udalosti. Používateľ nevie, čo sa práve vo svete deje, a tak nemôže do vyhľadávača zadať dopyt na získanie informácií o konkrétnej udalosti. Chce však získať prehľad.

Navyše, hneď po publikovaní nového článku na spravodajskom portáli naň ešte neexistuje dostatočný počet odkazov z iných portálov. Algoritmy využívajúce štruktúru prepojení webu na hodnotenie a usporadúvanie výsledkov vyhľadávania (akým je napr. PageRank) tento článok bez uvažovania ďalšieho kontextu nezaradia na popredné miesta, hoci je aktuálny a relevantný voči zadanému dopytu [7].

Existuje veľa spravodajských portálov, ktoré publikujú stovky článkov denne. Je nemožné sledovať ich všetky. Na druhej strane, nie všetky novinové portály informujú o každej udalosti. Ak teda sledujeme len jeden vybraný portál, môžu nám uniknúť potenciálne zaujímavé články. Riešením je automatické zoskupovanie článkov z viacerých zdrojov a ich jednotné prezentovanie používateľovi podľa udalosti, o ktorej informujú [12]. Používateľ získa prehľad aktuálneho diania a v prípade záujmu o ďalšie informácie vie, čo hľadať.

V tomto príspevku prezentujeme našu prácu z oblasti hľadania vzťahov medzi informačnými entitami z viacerých zdrojov na webe a ich prepájania. Zameriavame sa na novinové články, ktoré zoskupujeme podľa spoločnej udalosti, o ktorej informujú. Zoskupovanie realizujeme v reálnom čase, nejedná sa teda o tradičné zhlukovanie. Články nezaraďujeme do rovnakej skupiny na základe podobnosti témy alebo kategórie, ale na základe konkrétnej udalosti z jedného dňa, ku ktorej sa viažu. Táto činnosť sa často vykonáva manuálne a je známa ako monitorovanie (prehľad) tlače.

2 Príbuzné práce

Veľmi rozšírený spôsob prepájania informácií z rozličných zdrojov na webe je vytváranie zmiešanín (angl. mashup[1]). Mashup je spojenie vecí, ktoré pôvodne neboli určené na spoluprácu, za účelom dosiahnutia nového cieľa [1]. Typickým príkladom mashup je zobrazenie dodatočných informácií na mapovom podklade (napr. poloha objektov akými sú autobusové zastávky, pamiatky, obchody, divadlá).

Mnoho webových portálov a služieb poskytuje verejne dostupné API, pomocou ktorého môžu používatelia zahrnúť ich funkcionalitu do svojich aplikácií. Mashup môžeme vytvoriť aj pospájaním výstupov viacerých API. Hlavným účelom takýchto mashup je najmä agregácia údajov z rôznych zdrojov, vytváranie alternatívnych používateľských rozhraní existujúcich portálov alebo služieb, personalizácia webových stránok alebo monitorovanie vybraného zdroja údajov a detekcia zmien [14].

Typický prípad použitia integrácie informácií na webe, opísaný v [2], vychádza z blogu obsahujúceho recenzie filmov. Každý príspevok na blogu je automaticky doplnený o ďalšie informácie o filme, ktorého sa recenzia týka, akými sú obrázky z oficiálnej stránky filmu, životopisy hercov získané z encyklopédie alebo časy premietania recenzovaného filmu v miestnych kinách. Na realizáciu opísaného scenára potrebujeme zdroje vhodne anotovaných údajov, ktoré sú navyše strojovo spracovateľné, t.j. v príslušnom otvorenom formáte. Vhodnými formátmi sú napr. RSS a RDF. V prípade dostupnosti takto anotovaných zdrojov je výzvou automatické objavovanie sémantických asociácií medzi jednotlivými informačnými entitami [9].

Viacero prác sa zaoberá integráciou informácií v konkrétnej doméne, akou sú webové spravodajské portály. Použitie rozličných zhlukovacích algoritmov a metrík podobnosti na

[1] Ďalej v texte používame anglický pojem vzhľadom na to, že nie je v tejto oblasti vhodný slovenský pojem, ktorému by komunita rozumela.

zhlukovanie rádovo tisícok článkov vyhodnotili v [5]. Z výsledkov vyplýva, že najvhodnejším algoritmom na zhlukovanie novinových článkov je K-means s mierou podobnosti vyjadrenou kosínusom vektorov reprezentujúcich porovnávané články. Zo záverov však nie je jasné, či bolo cieľom zhlukovať články podľa širšieho tematického zamerania alebo podľa spoločnej udalosti, o ktorej informujú. V prvom prípade možno výsledok zhlukovania použiť na odporúčanie doplňujúcich článkov, z ktorých sa čitateľ dozvie viac o danej téme. V druhom prípade možno výsledky použiť ako zdroje informácií k danej udalosti, ktoré sú veľmi podobné a čitateľovi spravidla stačí prečítať si článok z jedného zdroja.

V [10] riešili odporúčanie novinových článkov na základe podobnosti ich obsahu. Výsledkom práce je vektorová reprezentácia článku, ktorá slúži na uchovanie významných prvkov článku. Vektor každého článku obsahuje kľúčové slová extrahované z nadpisu článku a z jeho hlavného textu, index čitateľnosti článku, ako aj pomenované entity (mená a miesta spomenuté v článku). Takto reprezentované články z vybraného slovenského novinového portálu autori zhlukovali (využijúc pri tom metriku kosínusovej podobnosti) za účelom odporúčania tematicky príbuzných článkov na ďalšie čítanie.

Iný prístup k zoskupovaniu podobných článkov je použitie stromovej štruktúry [15]. Jednotlivé články sú reprezentované ako listy stromu. Tematicky podobné články majú spoločný rodičovský uzol, ktorý zoskupuje ich kľúčové slová, čím vytvára akýsi metačlánok. V takomto strome je možné rýchlo nájsť článok podobný k zadanému, ktorý využijeme na odporúčanie. Výhodou je, že prepočítanie podobnosti je možné realizovať v reálnom čase, nakoľko novovzniknutý článok je možné rýchlo zaradiť do vhodného podstromu. Nie je pritom nutné prepočítať podobnosti ostatných článkov v strome (iba okolie pridaného článku). Stromová reprezentácia množiny článkov je tiež výhodná na odporúčanie článkov z rovnakej tematickej kategórie.

V [6] je prezentovaná metóda na analyzovanie témy a na detekciu kategórie článku. Pracuje v reálnom čase hneď po publikovaní nového článku. Metóda je založená na extrakcii fráz tvorených podstatnými menami a pracuje bez použitia korpusu dokumentov. Pri extrakcii fráz sa vykonáva morfologická analýza identifikujúca korene slov a určovanie slovných druhov slov vo vetách. Jeden článok môže byť zaradený do viacerých kategórií, pričom jeho relevantnosť voči každej z kategórií môže byť iná. Autori experimentovali s 800 článkami v anglickom a japonskom jazyku. Dosiahnuté hodnoty presnosti (angl. *precision*) a úplnosti (angl. *recall*) boli nad 90 %. Aj táto metóda môže byť využitá na odporúčanie článkov z tematicky príbuzných kategórií, ktoré slúžia ako doplňujúce čítanie.

Zoskupovaniu novinových článkov podľa udalostí, o ktorých informujú, sa venuje projekt Europe Media Monitor [3, 12]. Je to projekt Európskej únie, v ktorom sa denne spracováva približne 100 000 článkov z 2 000 zdrojov. Články sú napísané v približne 50 rôznych jazykoch. Ich získavanie sa deje prostredníctvom RSS, ako aj prostredníctvom automatickej extrakcie hlavných častí textov z HTML dokumentov. Získané dáta sa používajú vo viacerých službách monitorovania udalostí a trendov, akou je napr. NewsBrief[2] [3, 12].

NewsBrief je služba zameraná na detekciu aktuálnych udalostí a krátkodobých trendov. Na zoskupovanie článkov pojednávajúcich o tej istej udalosti dňa používa metódu zhlukovania. Jeden zhluk má priradený nadpis a obsahuje odkazy na články z rôznych zdrojov, v ktorých sa píše o udalosti vyjadrenej v nadpise. Ako nadpis zhluku je použitý nadpis najreprezentatívnejšieho článku, ktorý je považovaný za medoid zhluku. Každých 10 minút sa načítajú články publikované za posledné 4 hodiny a zoskupia sa použitím K-means zhlukovacieho algoritmu. Vzdialenosť dvoch článkov je vyjadrená kosínusovou

[2] http://www.newsbrief.eu

podobnosťou. Na reprezentáciu článku sa používa vektor obsahujúci počet výskytov slov v článku. Použitím tejto metódy sa niekedy zoskupia aj články, ktoré spolu nesúvisia. Tu vidíme priestor na zlepšenie.

Vo všeobecnosti sa uvedené práce zaoberajú zhlukovaním celej množiny novinových článkov s rozličnými cieľmi (zoskupiť články podľa príbuznosti tém, kategorizovať ich, nájsť vhodné články na odporúčanie ďalšieho čítania). V žiadnej práci sa však nerieši zoskupovanie článkov v reálnom čase. Najbližšie je tomu služba NewsBrief, ktorá pracuje so 4 hodinovým oknom článkov. Jej výsledky však obsahujú pomerne veľa chybne zaradených článkov, v čom vidíme priestor na zlepšenie.

Opísané metódy väčšinou pracujú s vektorovou reprezentáciou článku, ktorej dominujú kľúčové slová. Výsledky tak najviac ovplyvňuje spôsob ich extrakcie. Tu podľa nás neexistuje jediný všeobecne použiteľný prístup, ten závisí od cieľov. Pri zaraďovaní článkov do tematických kategórií je možné využiť mapovanie príbuzných frekventovaných slov na spoločné koncepty, napr. použitím databázy WordNet [13]. Naopak, pri určovaní udalosti, o ktorej článok informuje, je vhodné použiť metódy na extrakciu pomenovaných entít, keďže tieto udalosť charakterizujú lepšie ako počet výskytov jednotlivých všeobecných slov.

3 Zoskupovanie novinových článkov

Pri zoskupovaní entít sa vo všeobecnosti často používa zhlukovanie. Je tomu tak aj v prípade zoskupovania novinových článkov. Zhlukovacie algoritmy využívajú rôzne, najčastejšie vektorové, reprezentácie entít (článkov). V prvom kroku sa náhodne vybrané entity určia za centrá budúcich zhlukov. Následne je každá entita zaradená do zhluku podľa zvolenej metriky na určovanie podobnosti. Najpoužívanejšími metrikami sú kosínusová podobnosť, Jaccardov index, Pearsonov korelačný koeficient, a iné. Ďalším krokom pri zhlukovaní je výpočet nových centier vytvorených zhlukov a preskupenie entít. Toto sa opakuje, až kým nedosiahneme ustálený stav.

Použitie zhlukovacieho algoritmu je vhodné v prípade, že máme naraz k dispozícii všetky spracovávané články a chceme ich rozdeliť napr. podľa témy. Tiež je nutné poznať počet zhlukov, do ktorých plánujeme články zaradiť. Od tohto počtu tiež závisí miera granularity rozdelenia článkov. Tento prístup však nie je vhodný v našom prípade, kedy chceme priradiť článok k príslušnej udalosti hneď, ako ho získame z novinového portálu. Nemáme tak k dispozícii všetky články naraz. Taktiež nevieme odhadnúť výsledný počet zhlukov (t.j. udalostí, ktoré sa za deň stanú, a o ktorých sa bude písať). Posledným špecifikom je veľkosť priestoru, s ktorým pracujeme. Do úvahy berieme články publikované len v rámci jedného dňa. Ak je aj nejaká téma aktívna počas viacerých dní, skladá sa z viacerých udalostí, o ktorých sa každý deň píšu samostatné články.

3.1 Reprezentácia novinových článkov

Na zoskupenie článkov počítame mieru podobnosti vybraných metadát reprezentujúcich článok. Čím viac údajov rovnakého typu majú dva články spoločné, tým pravdepodobnejšie je, že informujú o tom istom. Každý typ údaju opisujúceho článok má vlastný prah podobnosti a vlastnú váhu v reprezentácii článku.

V našej metóde reprezentujeme články množinou kľúčových slov, ktoré extrahujeme z ich textovej reprezentácie. Nerozlišujeme pri tom medzi nadpisom článku a jeho hlavným obsahom, prípadne krátkou anotáciou, ktorú novinové články zvyknú mať. Na extrakciu kľúčových slov používame viaceré webové služby, špecializované na tento účel, ktoré

pracujú s textom v anglickom jazyku. My pracujeme s textami v slovenskom jazyku. Preto text najskôr preložíme do anglického jazyka.

Je dôležité pracovať len s textovou reprezentáciou článku oddelenou od zvyšku webovej stránky, na ktorej bol publikovaný. Na získanie hlavného textu článku používame webovú službu *Readability*, ktorá je súčasťou projektu *Metall*[3] vyvinutého na našej fakulte. Jej vstupom je URL adresa článku (alebo ľubovoľnej webovej stránky). Výstupom je hlavný text očistený od ostatných častí webovej stránky (menu, hlavička, logo portálu, reklama, atď.). Tento text, s ktorým ďalej pracujeme, tvorí samotný článok s nadpisom, úvodom a hlavným telom.

Na preklad do anglického jazyka používame webovú službu Google Translate. Tu síce môžeme stratiť význam niektorých slov a vety nemusia byť preložené gramaticky správne, no význam dôležitých slov (ktoré sú potenciálnymi kandidátmi na kľúčové slová) ostane zachovaný.

Extrakciu kľúčových slov realizujeme viacerými spôsobmi. Kombinujeme rôzne webové služby, z ktorých každá vracia mierne odlišný typ kľúčových slov. Používame OpenCalais, delicious, tagthe.net. Vrátené kľúčové slová sú prevedené na slovný základ, nemusíme teda využívať lematizáciu. Spomenuté webové služby sa tiež postarajú o odstránenie stop slov z textu. Na samotnú extrakciu používame webovú službu *Meta* z projektu *Metall*, ktorá agreguje výstupy spomenutých služieb [4].

Tieto kľúčové slová zahŕňajú ako všeobecné slová (napr. aktivity opísané slovesami, ktoré sú dominantné pre daný článok), tak aj pomenované entity (napr. osoby, firmy a miesta spomínané v článku). Týmto pokrývame všetky časti reprezentácie navrhnutej v [10] okrem indexu citateľnosti, ktorý však nemá v našom prípade význam (dva články o rovnakej udalosti môžu byť napísané jednoducho aj zložito, v oboch prípadoch ich chceme zoskupiť do jednej skupiny).

Takúto reprezentáciu článkov ďalej využívame na ich zoskupenie. Problém môže nastať, ak rovnaké kľúčové slová opisujú dve rôzne udalosti v rámci jedného dňa, typicky keď je s jednou pomenovanou entitou spojených viacero aktivít (napr. prezident sa počas dňa zúčastnil dvoch významných udalostí). V takom prípade môžeme využiť čas publikovania článku (keďže tieto dve udalosti sa nemohli stať naraz).

3.2 Algoritmus na pridanie článku do skupiny

Zoskupujeme novinové články napísané v slovenskom jazyku, publikované na rôznych slovenských spravodajských portáloch. Zoskupovanie prebieha tak, aby sa články informujúce o tej istej udalosti, ktorá sa stala v priebehu jedného dňa, dostali do rovnakej skupiny. Článok môže informovať aj o viacerých udalostiach, potom bude zoskupený s ďalšími článkami, ktoré informujú o viacerých udalostiach, príp. bude v skupine sám. Každý článok je zaradený do práve jednej skupiny.

Každý spravodajský portál, z ktorého články spracovávame, poskytuje RSS kanál na prístup k najnovším príspevkom. Z RSS kanálov všetkých portálov pravidelne sťahujeme nové články a v reálnom čase (hneď po ich získaní z RSS kanála) ich zaraďujeme do už vytvorených skupín. Ak žiadna existujúca skupina nevyhovuje, vytvoríme pre článok novú skupinu. To nastane väčšinou vtedy, keď článok informuje o novej udalosti ako prvý. Jednotlivé príspevky rozlišujeme podľa URL adresy, ktorá je jedinečná pre každý z nich.

Zaradenie nového článku do skupiny realizujeme podľa algoritmu 1. Množinu kľúčových slov novo vytvoreného článku porovnáme s množinami kľúčových slov už zoskupených článkov z príslušného dňa. Z tohto porovnania zistíme prienik každej dvojice

[3] http://peweproxy.fiit.stuba.sk/metall

množín. Následne vypočítame, do akej miery sa nový článok *x* zhoduje s už zaradeným článkom *y* podľa vzťahu

$$S = \frac{|kw(x) \cap kw(y)|}{|kw(x)|}$$

kde *S* je miera zhody článku *x* s článkom *y* a *kw(x)* je množina kľúčových slov extrahovaných z článku *x*.

V menovateli vystupuje iba počet kľúčových slov článku *x*, pretože počítame iba do akej miery sa *x* podobá na *y* (nie ich vzájomnú podobnosť). Toto je dôležité, pretože môžeme mať dva články, ktoré síce informujú o tej istej udalosti, ale každý zachádza do inej miery podrobnosti. V takom prípade nebude ich vzájomná podobnosť rovnaká.

Algoritmus 1 Zaradenie nového článku *n* do skupiny

```
1:  max zhoda = 0,40; zhodný článok = NULL
2:  for each už zaradený článok x do
3:      vypočítaj počet k identických kľúčových slov n a x
4:      zhoda = k / počet všetkých kľúčových slov n
5:      if zhoda > max zhoda then
6:          max zhoda = zhoda
7:          zhodný článok = x
8:  if zhodný článok == NULL then
9:      zaraď n do novej skupiny
10: else
11:     zaraď n do skupiny k článku zhodný článok
```

Na začiatku výpočtu nastavíme požadovanú mieru podobnosti na určitú hraničnú hodnotu (ktorú sme určili experimentálne na 40 %). Prah predstavuje najmenší možný počet zhodných kľúčových slov, aby sme dali dva články do rovnakej skupiny. Ak pre článok nenájdeme žiadnu vhodnú skupinu (t.j. žiaden už zaradený článok s ním nemá dostatočný počet zhodných kľúčových slov), považujeme tento článok za prvý, ktorý informuje o novej udalosti a vytvoríme preň novú skupinu. Algoritmus opakujeme vždy po získaní nového článku z niektorého zo sledovaných spravodajských portálov.

S pribúdajúcimi článkami narastá teoretický počet možných porovnaní kľúčových slov, čo zvyšuje zložitosť a zväčšuje čas potrebný na zaradenie článku. Keďže však uvažujeme články iba z jedného dňa, ktorých sa na všetkých sledovaných portáloch spolu vyprodukuje približne 1 000, sme schopní porovnanie vykonať v reálnom čase (v jednotkách sekúnd). Pri prudkom náraste počtu článkov tak, že by nebolo možné v reálnom čase vykonať porovnanie kľúčových slov, je možné toto porovnanie obmedziť iba na top *N* najnovších článkov. Je málo pravdepodobné, že najnovší článok bude informovať o tej istej udalosti ako najstaršie články.

Porovnávanie množín kľúčových slov je možné v prípade dostupnosti vhodnej hardvérovej platformy aj paralelizovať. Porovnania novo pridaného článku *x* s už zaradenými článkami *y* a *z* sú nezávislé, môžu byť vykonané súčasne, čím v prípade potreby urýchlime výpočet.

Nie všetky novinové články publikované na spravodajských portáloch sa týkajú aktuálnych udalostí. Existujú články s úvahami a subjektívnymi názormi autora (napr. komentáre, blogy, rozhovory). Ak sa jedná o komentár k aktuálnej udalosti, v našej reprezentácii bude takýto článok priradený do skupiny k ostatným, ktoré o udalosti informujú objektívne. Iné články, napr. rozhovory, budú tvoriť samostatné jednoprvkové

skupiny (je veľmi malá pravdepodobnosť, že podobný rozhovor na podobnú tému bude naraz publikovaný na viacerých portáloch).

3.3 Určenie nadpisu pre skupinu článkov

Po zaradení článkov do skupín ich prezentujeme používateľovi prostredníctvom webového portálu. Každá skupina je zobrazená ako samostatná jednotka informujúca o udalosti, ktorá má priradené odkazy na zdroje (konkrétne články z pôvodných spravodajských portálov). Na zobrazenie skupiny musíme túto najskôr vhodne pomenovať. Nadpis musí čo najlepšie vystihovať danú udalosť.

Každý článok zaradený v danej skupine má svoj pôvodný nadpis. Máme vypočítanú zhodu pre každú dvojicu článkov. Na určenie nadpisu celej skupiny použijeme nadpis toho článku, ktorý skupinu najviac reprezentuje. Tým je článok, ktorý má najväčšiu zhodu so všetkými vo svojej skupine. Ak skupinu chápeme ako zhluk, potom článok s najväčšou zhodou s ostatnými článkami je jej centroid.

Tento prístup má výhodu v tom, že sa nemusíme zaoberať automatickým zostavením nadpisu skupiny z extrahovaných kľúčových slov. Tu by sme museli vyriešiť správne poradie slov a ich skloňovanie, čo nie je naším primárnym cieľom. Nevýhodou je, že nadpis reprezentatívneho článku nemusí úplne vystihovať všetky ostatné články, čo sa však ťažko posudzuje aj ľuďom.

4 Overenie a experimenty

Na zistenie miery zhody jedného článku voči druhému používame prah určujúci podiel kľúčových slov prvého článku, ktoré sa musia zhodovať s kľúčovými slovami druhého článku. Tento prah sme experimentálne určili na 40 %, pričom sme experimentovali s tromi rôznymi hodnotami. Najskôr sme manuálne stiahli niekoľko článkov zo slovenských spravodajských portálov a ručne ich zaradili do 5 skupín. Potom sme z nich pomocou uvedených webových služieb extrahovali kľúčové slová. Zistili sme, že články patriace do rovnakej skupiny mali aspoň 60 % zhodných kľúčových slov. Na druhej strane, počet zhodných kľúčových slov medzi článkami z dvoch rôznych skupín neprekročil 20 %. Na základe týchto zistení sme očakávali ideálny prah v rozmedzí týchto hodnôt a rozhodli sme sa experimentovať s tromi prahmi: 30 %, 40 % a 50 %.

Ako referenčnú vzorku na porovnanie našej metódy sme zvolili rozdelenie novinových článkov z portálu NewsBrief. Z tohto portálu sme použili články napísané v slovenskom jazyku.

Z portálu NewsBrief sme stiahli všetky zhluky slovenských článkov, z ktorých sme následne získali URL adresy jednotlivých článkov. Články sme náhodne usporiadali a túto vzorku sme použili ako vstup pre našu metódu. Jednotlivé články sme následne v cykle spracovali a zaradili do skupín. Hoci sme mali k dispozícii všetky naraz, pristupovali sme k nim v zmysle, ako keby sme ich získavali postupne (neuvažovali sme reálny čas napísania, ale náhodné usporiadanie).

Pracovali sme so vzorkou 106 článkov, ktoré boli pôvodne rozdelené do 24 skupín. Najmenšia skupina obsahovala 2 články, najväčšia 17. Tu sa ukázalo, že NewsBrief nezaradí všetky články správne. Preto sme vytvorené skupiny museli manuálne upraviť. V spomínanej najväčšej skupine boli články o 8 nesúvisiacich udalostiach.

Výsledky pre tri rôzne hodnoty prahu minimálnej zhody kľúčových slov uvádzame v tab. 1. V prvom riadku je uvedený celkový počet skupín, ktoré sme v jednotlivých experimentoch vytvorili. Ďalej uvádzame počet skupín, ktoré boli vytvorené presne podľa

referenčnej vzorky (bez manuálnej úpravy jej chybne zaradených článkov). Ďalšie riadky lepšie odzrkadľujú vlastnosti navrhnutej metódy, keďže tu sme vytvorené skupiny vyhodnotili manuálne.

Niektoré články sme zaradili do nesprávnej skupiny, t.j. k článkom o inej udalosti. Ich počet však bol pri vyšších hodnotách prahu pomerne malý. Vytvorili sme tiež skupiny, ktoré obsahovali články informujúce o dvoch udalostiach (každá udalosť bola zastúpená viacerými článkami, čo je rozdiel oproti predošlému riadku). Tieto skupiny by sa teda dali ďalej rozdeliť. Pri použití prahovej hodnoty 50 % sme takúto skupinu nedostali.

Na druhej strane, boli udalosti, o ktorých informovali články vo viacerých skupinách (pričom skupina bola vytvorená korektne, všetky články v nej spolu súviseli). To znamená, že tieto skupiny by sa dali spojiť do väčšej. Dialo sa tak najmä pri 50 % hodnote prahu, ktorá sa ukázala byť príliš striktná. Hodnota 40 % bola dobrým kompromisom medzi príliš veľa a príliš málo skupinami.

V referenčnej vzorke existovali skupiny, v ktorých bol jeden článok o inej udalosti ako zvyšné. Pomocou navrhnutej metódy sa nám podarilo opraviť zaradenie týchto článkov, takže sa dostali do správnej skupiny. Najviac článkov sa nám takto podarilo presunúť pri hodnote prahu 40 %.

V referenčnej vzorke boli tiež skupiny, ktoré obsahovali viacero článkov o rôznych udalostiach. Z nich sa nám úspešne podarilo spraviť až 12 nových skupín (40 % prah). Toto pokladáme za najdôležitejší prínos nášho prístupu oproti metóde používanej portálom NewsBrief. Z výsledkov vidíme, že optimálna hranica pomeru zhodných kľúčových slov voči všetkým opisujúcim článok je 40 %.

Tab. 1. Výsledky zoskupovania článkov.

	30 % prah	40 % prah	50 % prah
Celkový počet skupín	34	46	60
Počet skupín podľa NewsBrief	19	10	4
Články v nesprávnej skupine	11	4	4
Rozdeliteľné skupiny	4	1	0
Spojiteľné skupiny	6	18	43
Opravené zaradenie článku	3	4	2
Opravené skupiny	4	12	8

5 Záver a budúca práca

V tomto článku sme prezentovali prístup zoskupovania článkov podľa udalostí, o ktorých informujú. Tento prístup je odlišný od tradičného zhlukovania v tom, že článok zaraďujeme hneď, ako bol publikovaný. Navyše nechceme meniť zaradenie už spracovaných článkov.

Ukázali sme, ako prospešné môžu byť kľúčové slová extrahované z článku. Ďalšie metadáta nám môžu pomôcť zlepšiť zaradenie a spresniť ho v hraničných prípadoch. Medzi príklady takýchto metadát zaraďujeme napr. čas publikovania článku či jeho kategóriu.

Prezentovaná metóda môže byť použitá aj na zoskupovanie iných typov webových objektov. Základom je textová reprezentácia objektu, z ktorej vieme extrahovať kľúčové slová. Slabým miestom prezentovanej metódy sú webové služby, ktoré používame na extrakciu kľúčových slov. Z tohto dôvodu používame viac webových služieb a ich výstupy agregujeme. Všetky použité webové služby pracujú s anglickými textami, preto musíme články prekladať. Pri tom sa niekedy môže stratiť alebo zmeniť pôvodný význam slova.

Webové služby nám však vrátia viacero kľúčových slov a je nepravdepodobné, že by sa pri preklade stratil význam všetkých slov, ktoré budú neskôr označené za kľúčové. Našu metódu by mohlo vylepšiť použitie doplňujúcej služby, ktorá by bola schopná extrahovať kľúčové slová zo slovenského textu.

Zoskupené články plánujeme v budúcnosti prezentovať na automaticky zostavenom portáli s prehľadom aktuálnych udalostí. Ďalšou prácou, ktorou sa plánujeme v budúcnosti zaoberať, je automatické určenie tematickej kategórie pre skupinu článkov, ako aj jej vhodného nadpisu. Portál bude personalizovaný a používateľom ponúkne informácie podľa ich záujmov. Tie budeme zisťovať najmä vďaka využitiu implicitnej spätnej väzby [8].

Poďakovanie
Táto publikácia vznikla vďaka čiastočnej podpore projektov VG1/0675/11/2011-2014, APVV-0208-10 a projektu v rámci OP Výskum a vývoj pre projekt: Výskum metód získavania, analýzy a personalizovaného poskytovania informácií a znalostí, ITMS: 26240220039, spolufinancovaný zo zdrojov Európskeho fondu regionálneho rozvoja.

Literatúra

1. Alba, A., Bhagwan, V., Grace, J., Gruhl, D., Haas, K., Nagarajan, M., Pieper, J., Robson, C., Sahoo, N.: Applications of Voting Theory to Information Mashups. In: *Proc. of 2008 IEEE Int. Conf. on Semantic Computing*, IEEE Computer Society, Washington (2008), 10–17.
2. Ankolekar, A., Krötzsch, M., Tran, T., Vrandecic, D.: The two cultures: mashing up web 2.0 and the semantic web. In: *Proc. of the 16th Int. Conf. on World Wide Web*, ACM Press, New York (2007), 825–834.
3. Atkinson, M., Van der Goot, E.: Near real time information mining in multilingual news. In: *Proc. of 18th Int. Conf. on World wide web*, ACM Press, New York (2009), 1153–1154.
4. Barla, M., Bieliková, M.: Ordinary Web Pages as a Source for Metadata Acquisition for Open Corpus User Modeling. In: *WWW/Internet*, IADIS Press (2010), 227-233.
5. Bouras, C., Tsogkas, V.: Assigning Web News to Clusters. In: *Proc. of 2010 Fifth Int. Conf. on Internet and Web Applications and Services*, IEEE Computer Society, Washington (2010), 1–6.
6. Bracewell, D.B., Yan, J., Ren, F., Kuroiwa, S.: Category Classification and Topic Discovery of Japanese and English News Articles. *Electronic Notes in Theoretical Computer Science* 225 (2009), 51–65.
7. Del Corso, G.M., Gullí, A., Romani, F.: Ranking a Stream of News. In: *Proc. of 14th Int. Conf. on World Wide Web*, ACM Press, New York (2005), 97-106.
8. Holub, M., Bieliková, M.: Estimation of User Interest in Visited Web Page. In: *Proc. of 19th Int. Conf. on World Wide Web*, ACM Press, New York (2010), 1111–1112.
9. Kochut, K.J., Janik, M.: SPARQLeR: Extended Sparql for Semantic Association Discovery. In: *Proc. of 4th European Conf. on The Semantic Web: Research and Applications*, Springer-Verlag Berlin, Heidelberg (2007), 145-159.
10. Kompan, M., Bieliková, M.: Content-Based News Recommendation. In: *E-Commerce and Web Tech.*, Aalst, W. et al. (Eds.), Springer Berlin Heidelberg (2010), 61-72.

11. Raza, M., Hussain, F.k., Chang, E.: A methodology for quality-based mashup of data sources. In: *Proc. of 10th Int. Conf. on Information Integration and Web-based Applications & Services*, ACM Press, New York (2008), 528–533.
12. Steinberger, R., Pouliquen, B., van der Goot, E.: An introduction to the Europe Media Monitor Family of Applications. In: *Proc. of SIGIR 2009 Workshop on Information Access in a Multilingual World*, Gey, F., Kando, N., Karlgren, J. (Eds.), 1–8.
13. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: *Proc. of 16th Int. Conf. on World Wide Web*, ACM Press, New York (2007), 967-706.
14. Wong, J., Hong, J.: What do we ”mashup” when we make mashups? In: *Proc. of 4th Int. workshop on End-user Software Eng.*, ACM Press, New York (2008), 35–39.
15. Zeleník, D., Bieliková M.: News Recommending Based On Text Similarity And User Behaviour. In: *7th International Conference on Web Information Systems and Technologies* (2011).

Annotation:

Grouping of News Articles according to Events

Almost everything can be found on the Web. The problem is that information is scattered across many different places and traditional search engines cannot integrate them. In this paper we present a method for automatic integration of news articles written in Slovak language, which inform about the same event. The articles were published on various news portals. We group news articles according to the compliance of their keywords. We present a comparison of our method with a service of similar functionality, results of which we were able to improve. We use the proposed method to create a personalized news portal which keeps the user up-to-date about current affairs. He has got an overview of everything being published on Slovak news portals and therefore does not have to manually watch all of them.

Building a Knowledge Base with Data Crawled from Semantic Web

Ivo LAŠEK[1], Peter VOJTÁŠ[2]

[1]*Katedra softwarového inženýrství, FIT ČVUT v Praze*
Kolejní 2, 160 00 Praha 6
lasekivo@fit.cvut.cz
[2]*Katedra softwarového inženýrství, MFF UK Praha*
Malostranské nám. 25, 118 00 Praha
vojtas@ksi.ms.mff.cuni.cz

Abstract. In this paper, we compare various approaches to semantic web data crawling. We introduce our crawling framework, which enables us to organize and clean the data before they are presented to the end user or used as a knowledge base. We present methods of semantic data cleaning in order to keep the knowledge base consistent. We used the proposed framework to build a knowledge base containing data about persons crawled from semantic web data sources. In this paper we present the results of the crawling process.

Keywords: semantic web, search, crawling

1 Introduction

Nowadays, conventional search engines still present their results rather in a document centric way. Both most popular search engines (Google and Yahoo!) are already able to parse some semantic information, but we can't perform any structured queries over the collected data. By Google, this functionality is called Google Rich Snippets [1] and the data extracted from semantic documents is displayed as a grey information snippet under each search result (in case that the target document contains some structured data). Yahoo! offers very similar functionality under the name SearchMonkey [2]. Both extract the structured data from variety of sources using standard markup and vocabularies (e.g. Microdata, Microformats, RDFa, eRDF). Google as well as Yahoo! states that they use these data to enhance their search results. Yahoo! goes even further and provides developers with the possibility to develop their own applications exploiting such a structured data.

Many good managed public structured data sources have emerged like DBpedia[1] or FreeBase[2]. Popular web pages present part of their data in a structured form. For example LinkedIn[3] uses Microformats to publish contacts of registered users, or basic information about their employment history. There are initiatives demanding the government data to be published as Linked Data. In Czech Republic this interest is represented by the initiative OpenData.cz[4].

[1] http://dbpedia.org
[2] http://www.freebase.com
[3] http://www.linkedin.com
[4] http://opendata.cz

These and many other examples lead us to the idea to form a knowledge base composed of the data collected from such structured data sources. The amount of the data already available in a structured form on the web offers the possibility to form huge knowledge bases with fresh data published every day.

However, conventional search engines don't combine the information from multiple sources to form a completely new data source, to find even some new facts. The data with a semantic meaning is used rather as a better form of keywords describing the content of an unstructured document.

For last couple of years, a new form of a search engine has emerged. In this article, we will call it the semantic search engine. Semantic search engines address the problems of crawling and integration of the data obtained from the semantic web. So far, most of the work focused on crawling as much data as possible and providing some kind of a simple user interface to search them. In most of the cases the interface composes of an ordinary search field.

Little or no stress was put on the quality of the crawled data. The search results of the semantic search engines are full of duplicates, which makes it difficult to locate relevant data sources. The documents presented in search results contain often almost no relevant information apart from its name represented by the label.

In this article we introduce our framework for crawling semantic data from the web. We address the problem of the quality of the data. The results of a crawling task performed by our framework are introduced. We present basic statistics of the crawled data and evaluate its quality. We provide an important view of the nature of the data that can be obtained from semantic data sources (Linked Data). The results of our work can serve as a base for designing specific knowledge bases composed of Linked Data.

2 Related work

Semantic search engines can be classified into two categories as stated in [9]. Systems that operate on a document abstraction level (we call them document centric) and systems that operate on an object oriented model (we call them entity centric).

2.1 Document Centric Approach

The representatives of this group are Swoogle [7] and Sindice [14]. While Swoogle does not seem to display continuous crawling capabilities, Sindice is actively developed and serves as a service for public use.

The information in a document centric semantic search engine is organized by documents - i.e. the data sources, where the information comes from. A document can be a single web page, an RDF document, an RSS feed etc. The document centric search engines return documents as search results. It does not matter, how many entities the document describes. Also, when one entity is described in multiple documents, the search engine returns them separately. Then, the user has to merge the information manually. From this point of view, the entity centric approach seems more promising.

2.2 Entity Centric Approach

An entity centric semantic search engine works on the higher abstraction level. The information from several documents describing the same thing is aggregated as one entity. Such entity is thus better described, because the search engine collects data from multiple semantic documents.

To the best of our knowledge, there are two running examples on the web: SWSE [10] and FalconS [3]. The idea of the entity centric approach is mentioned also in earlier works about SemSearch [13] and TAP [8]. The problem by this approach is that matching same entities from different data sources is a difficult task. SWSE relies on the use of the matching based on unique identifiers and Inverse Functional Properties in combination with reasoning. FalconS adds linguistic matching and structural matching techniques [12].

Current semantic search engines try to store and search as much data as possible. They often store entities with only a limited amount of properties. The matching of such entities is impossible. The information provided about the entity is simply insufficient, to identify it reliably. The final search results are messy and the end user is usually overwhelmed with many similar results from different sources that were not appropriately merged.

In our work, we would like to fill this gap proposing a set of rules for creating entities in the repository. We define qualitative requirements in order to avoid creation of entities with almost no information value. We prefer the quality of the data over the quantity.

The search of crawled data is performed in the case of all mentioned projects as a full-text search of textual attributes of crawled entities. Hence, the ordering of search results is a critical factor for the final user experience. One possible ranking approach (ReConRank) to determine the relevancy of crawled resources was introduced in [11]. The ReConRank is a variation of the classical PageRank [15] algorithm, adapted to RDF data. The searching of crawled data is not the part of our work. However, the ranking function may be used for data consolidation described in Section 4.

3 Semantic Data as a Knowledge Base

When we talk about semantic data as a source of information for a knowledge base we do not necessarily mean the knowledge base consisting of all the information available on the web. Rather we think about specific knowledge bases containing only information of a particular type. For example a geo-location application may exploit information about places, social web applications might crawl structured information about people, a price comparison portal may crawl information about products to enrich their description with its parameters.

Particularly, we focus at the domain of persons, organizations, places and products. According to a study carried out by Microsoft [5] the People / places / things domain was the most common query type.

To gain the most value from the crawled data, the optimal organization of a repository is entity centric. In an entity centric repository, the data describing same entities from different sources are merged and the entities are thus better described.

To increase the usability of a knowledge base, the number of duplicates should be minimized. This is not an easy task. Various approaches to ontology matching are introduced in [6]. But even sophisticated ontology matching techniques do not ensure 100 percent accuracy.

However, Web Ontology Language (OWL) offers ways to identify same entities, even when they use different identifiers:

- The use of the property `owl:sameAs`[5]. This property links an individual to an individual and indicates that two URI references actually refer to the same thing.

[5] In the whole article under `owl:` we understand the XML namespace `http://www.w3.org/2002/07/owl/#`.

- Identification of Inverse Functional Properties and their mapping. If a property is declared to be inverse-functional (`owl:InverseFunctionalProperty`), then the object of a property statement uniquely determines the subject. If two objects share the same value of a common inverse-functional property, they may be considered to be the same.

In the section 6 we evaluate the usage of these properties and their reliability in relation to entity matching.

4 Semantic Web Crawling

Our crawler is able to work with various document types (including RDF/XML, Turtle, Notation 3, RDFa, Microformats). We try to extract semantic information from ordinary HTML web pages too. For example, we use the title of a webpage as a label for a given URI. Thanks to the extendibility of the system, it is possible to add other custom parsers that would be able to work with unstructured documents too.

4.1 The Crawling Framework Architecture

The basic concept of the semantic web crawling framework is showed in Figure 1. We designed all the parts of the system as independent services that can be changed in case of need. The independent parts can be scaled individually too. For example collecting URLs in the Spider service is less demanding than storing the results to the repository.

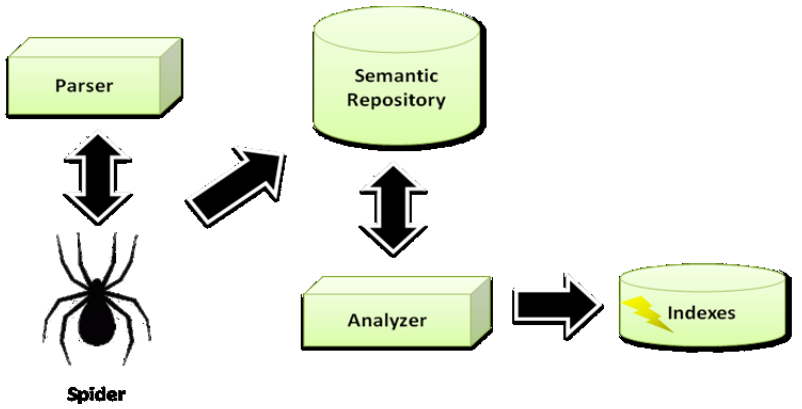


Figure 1. Semantic web crawling framework - system architecture.

The whole workflow starts with the *Spider* that discovers new documents in the web, downloads them, determines their content type and sends the downloaded document to an appropriate *Parser*.

The document is parsed and returned in a common format to the *Spider*. We use N-Quads [4] as the inner format. The N-Triples is extended with the information about the context (the source of the information represented by the triple). Thus quadruples are formed.

The *Spider* then stores the extracted quadruples in the *Semantic Repository* and extracts URIs from the parsed document. The extracted URIs are queued for crawling. We use the breath first search strategy to crawl relevant data sources from the semantic web graph.

The *Analyzer* walks through the data stored in the *Semantic Repository* and groups the information about same facts with different URIs.

In the rest of this paper, with entity we mean a real world object that can be described by a set of properties. In the context of semantic web each such an object has a unique identifier (URI). The description of the grouping process follows.

Only entities fitting certain qualitative requirements are selected (more on the entity selection in Section 4.2). For each suitable entity a representative is created. The representative is a new entity with the same properties as the former one. If there are more entities with different URIs describing the same thing, they are merged under one representative. The process of entity merge is described in following section. The representatives are marked in the *Semantic Repository* and during the queries only entities with this mark are returned. The rest is filtered out from the results.

4.2 Data Consolidation

In order to achieve certain level of quality of the data in the repository, we need to define qualitative requirements on entities. In order to form a representative, an entity has to fit these requirements.

In our case, we defined following general rules:

- The entity has to have at least three properties. This requirement ensures certain quality of the data. There are many resources on the web that contain only a label and nothing more. They are often created for testing purposes and even the label is wrong.
- At least one property has to be inverse-functional (i.e. uniquely identifies the entity). This requirement makes it easier to match the same entities. For example, for an entity of the type Person, we require the presence of email, homepage address or similar type of a unique identifier.

The requirements should ensure that the entity matching will be easier. They are often domain dependent. For some domains the inverse-functional property requirement may be too restrictive, but a combination of other attributes may work well. For example in the product domain, we will probably not find an ean code for each product. However, the combination of product manufacturer and product number properties serves also as a good identifier.

When a representative is created, it takes attributes of the source entity and both the representative and the entity are marked as same (using `owl:sameAs` property). If there are other entities same as the source entity, they are merged under the same representative. Their attributes are copied to the representative and they are marked as same.

The merged entities do not have to fit the requirements. It is enough, when they are mapped to some existing representative. Then their attributes are used to supplement the existing information about the representative.

An example of the data consolidation is shown in Figure 2. At the beginning we crawled two resources describing the same person (Tim Berners-Lee) with different identifiers (`<http://www.w3.org/People/Berners-Lee/card#i>` and `<http://rdf.freebase.com/rdf/en.tim_berners-lee>`). From W3.org we got the homepage and the email, from Freebase we have the information about the name. Dashed lines denote the information added by the *Analyzer*. First the representative (`<http://myrep.cz/tim-berners-lee>`) is created using the information

obtained from W3.org. The entity crawled from this resource has properties that are inverse-functional (homepage and email). Then the other entity, obtained from Freebase, is mapped to this representative thanks to the information that it is same as the W3.org entity. Attributes of the first entity are copied to the representative and both entities are marked as same. The representative is derived from the first discovered entity that fits described criteria. As all attributes of matched entities are added to the representative at the end, the order of mapping does not matter.

With more crawled data, the *Analyzer* is able to identify possible duplicates and merge them. This situation is denoted in Figure 3. In this example, there are two representatives (entity A and B), created from two entities (C and D). During crawling, another entity is discovered – entity E. The entity E holds the information that it is same as C and D. Then the analyzer chooses one of the representatives (A in this case) and merges the other with it. The other representative (B) is marked as an ordinary entity and can't be searched any more. But it still remains in the repository in case that someone already uses its URI.

The selection of the representative that will be preserved can be done based on various metrics including Ontology Rank [7] or ReConRank [11]. Currently, we use a simple metric. We count triples the particular representative is involved in. The representative with the higher count of triples is preserved.

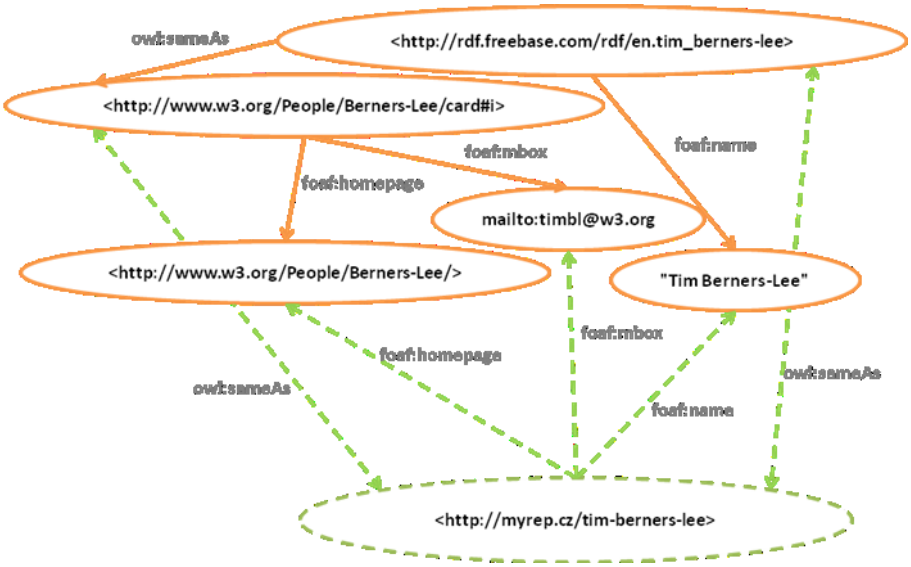


Figure 2. Data consolidation example[6]. Dashed lines denote the information added by the *Analyzer*.

[6] In this article by `foaf:` we abbreviate the XML namespace `http://xmlns.com/foaf/0.1/`.

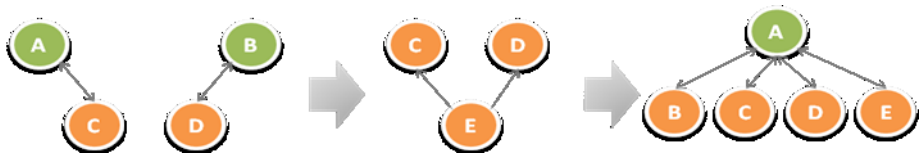


Figure 3. Dealing with duplicates. Arrows denote entities that are marked as same

4.3 Indexes

Currently, we use two types of additional indexes apart from ordinary indexes maintained by the semantic repository. One is the fulltext index in order to enable fulltext search for the end users. Only representatives are indexed for fulltext search. They are indexed by the values of literal attributes.

Another index serves for faster identification of same entities. In this index, every entity (identified by its URI) merged with a representative has a pointer to this representative. Entities pointing to the same representative are considered to be same.

5 Implementation

The whole system is implemented in Java. For crawling, we use the LDSpider project. As the parser of various types of documents, we use Any23. Any23 is publicly available as a service. However, we deployed it as a service on our local server in order to have the parser under control.

As semantic repository, we use Sesame which enables the work with quadruples including the information about the context of a triple. We created an interface that enables LDSpider to store the data directly in Sesame.

The Analyzer is our custom made application, which we modified for the evaluation purposes.

The indexes are implemented using Apache Solr, which is an open source search server based on Lucene.

6 Evaluation

6.1 Used Data

We ran a random crawling task (starting from a random URI) for 51 hours. During this crawl, we downloaded data from 25 129 structured resources. We were crawling data without any filtering of the URIs, without limiting it to crawl a specific domain. During the evaluation, we tried several different crawling tasks. It is interesting, that regardless of the starting URI, the crawl always spread in a wide range of resources, even when we used a starting resource with only three triples, the crawl resulted in millions of crawled triples and we stopped it after a certain period, not because it would run out of available URIs.

A basic summary of the crawled data is shown in the Table 1. Totally we crawled information about 839 029 distinct entities. Out of them 264 309 were of the type Person. This large number of occurrences of the type Person is caused by the big popularity of *Friend of a Friend* (*FoaF*) profiles. Also the vast majority of the entities is described using the Foaf ontology. By majority of companies (426 out of total 448) the type

`foaf:Organization` was used too. By places, the situation is not so clear. The ontologies of Yago, DBPedia and Freebase were used uniformly. DBPedia slightly prevails, but from such a small number of identified places, we can't conclude any interesting fact.

Entity type	Number of occurrences
Any type	839 029
Person	264 309
Organization	448
Places	10

Table 1 Crawled data summary - selected data types.

6.2 Tests

First we were interested, how many triples in average, describe one entity. The number of triples gives us a clue, what is the level of quality of crawled data. We were working with entities of the type `foaf:Person`, as it is the most common type in our data set. In average one Person is described with 25 triples. This is fairly enough to gain useful information about a person. Unfortunately, many of the provided statements are duplicated using for example only slightly different URIs (like `<http://www/my.opera.com/GOISON/xml/foaf#me>` and `<http://www/my.opera.com/GOISON/xml/foaf>`). Some entities are involved in many statements, but the real information gain is low. Thus, more revealing is the number of entities that are involved in at least average count of triples. There are 36 503 entities described by more than 24 triples.

Another characteristic, we were interested in, was the use of Inverse Functional Properties by persons. Table 2 shows the number of occurrences of various Inverse Functional Properties of a person from the Foaf ontology. Most of the persons could be identified by the value of `foaf:homepage` (62 % of all persons in the dataset) or `foaf:mbox_sha1sum` (63 % of all persons in the dataset). The low usage of e-mails (`foaf:mbox`) is understandable. Because of the problems with spam, people rather use the encrypted version of their e-mail address.

Property	Occurrences	Property	Occurrences
foaf:mbox	617	foaf:logo	3
foaf:homepage	164 080	foaf:mbox_sha1sum	166 300
foaf:isPrimaryTopicOf	37	foaf:msnChatID	445
foaf:aimChatID	222	foaf:opened	151
foaf:icqChatID	233	foaf:weblog	12 003
foaf:jabberID	128	foaf:yahooChatID	755

Table 2. The usage of inverse functional properties by persons.

Finally, we tested the accuracy of various properties according to possible entity matching. Figure 4 shows the count of entities matched based on selected properties. We identified 132 entities that were marked as same, but had different URIs. We checked manually all the 132 entities and all of them were mapped correctly according to their other properties.

68 entities out of 132 had exactly the same name and would match correctly based on this attribute. By the others, the name was missing, or the names were filled in different forms (e.g. "Herman, Ivan" and "Ivan Herman"). However, the name is a very poor identifier. When we were testing matching only based on the name, the accuracy was very low. For example there were thousands of entities whose name was simply "David", though none of them was same as some another. Homepage and email serve as very good identifiers according to accuracy. Unfortunately, very often these properties have different values even by same entities. For instance, many people have more than one homepage. There are countless variants, how to write an URL too. Therefore only 23 entities could be matched based on the `foaf:homepage` attribute and 22 based on the `foaf:mbox_sha1sum`.

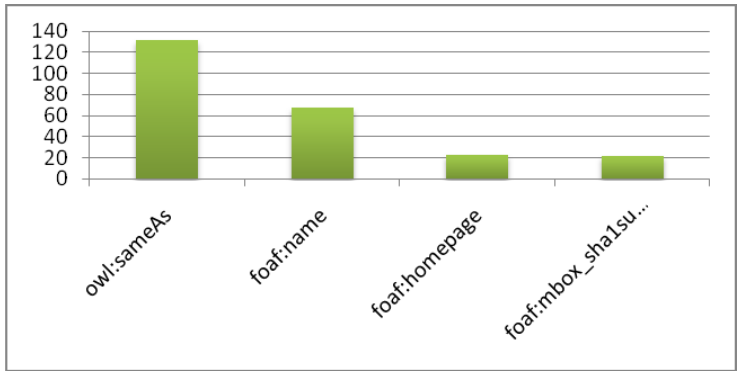


Figure 4. Entity matches based on individual properties.

The evaluation showed the strength of the usage of the `owl:sameAs` property for entity mapping. While using Inverse Functional Properties, we were able to map fewer entities. By persons, the usage of the homepage and the encrypted email address were the most frequently used properties.

To determine the recall in the experiments is a tough task. We would have to walk through all the 264 309 entities and check manually whether they are same. We plan to develop some heuristics that would facilitate this task.

7 Conclusion

In this paper we described various approaches to semantic web data crawling. We introduced and deployed the crawling framework supporting the entity centric organization of data. We proposed the rules for data consolidation to achieve better quality of stored information.

During our experiments, we evaluated the crawled data and possibilities of their consolidation based on their Inverse Functional Properties. The safest attribute for matching entities in terms of accuracy remains the `owl:sameAs`.

In our future work, we would like to perform crawling tasks in bigger extents. We have been limited by computational resources so far. We plan to improve the performance of the semantic data storage, as Sesame itself proved to be quite slow for the data analysis.

Acknowledgements
This work has been partially supported by the grant of The Czech Science Foundation (GAČR) P202/10/0761 and by the grant of Czech Technical University in Prague SGS11/085/OHK3/1T/18.

References

1. Google Rich snippets, http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=99170.
2. Yahoo! SearchMonkey, http://developer.yahoo.com/searchmonkey/.
3. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: *Proceeding of the 17th international conference on World Wide Web*, 1101-1102, New York, NY, USA, 2008, ACM.
4. Cyganiak, R., Harth, A., Hogan, A.: N-Quads: Extending N-Triples with Context. http://sw.deri.org/2008/07/n-quads/
5. Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., Robbins, D. C.: Stuff i've seen: A system for personal information retrieval and re-use. In: *SIGIR'03*, 72-79, ACM Press, 2003.
6. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer-Verlag, Berlin Heidelberg (DE), 2007.
7. Finin, T., Peng, Y., Scott, R., Joel, C., Joshi, S. A., Reddivari, P., Pan, R., Doshi, V., Ding, L.: Swoogle: A search and metadata engine for the semantic web. In: *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*, 652-659, 2004 ACM.
8. Guha, R., Mccool, R., Miller, E.: Semantic Search. In *Proceedings of International World Wide Web Conference*, 700-709, 2003, ACM.
9. Harth, A., Hogan, A., Umbrich, J., Decker, S.: SWSE: Objects before documents! In: *Proceedings of the Billion Triple Semantic Web Challenge, in conjunction with 7th International Semantic Web Conference*, 2008.
10. Harth, A., Hogan, A., Delbru, R., Umbrich, J., Decker, S.: SWSE: answers before links. In: *Proceedings of Semantic Web Challenge*, 2007.
11. Hogan, A., Harth, A., Decker, S.. ReConRank: A Scalable Ranking Method for Semantic Web Data with Context. In: *2nd Workshop on Scalable Semantic Web Knowledge Base Systems*, 2006.
12. Jian, N., Hu, W., Cheng, G., Qu, Y.: Falcon-AO: Aligning Ontologies with Falcon. In: *Proceedings of K-Cap 2005 Workshop on Integrating Ontologies*, 87-93, 2005.
13. Lei, Y., Uren, V., Motta, E. Staab, S., Svátek, V. (Eds.): SemSearch: A Search Engine for the Semantic Web. *Managing Knowledge in a World of Networks*. Springer Berlin / Heidelberg, Volume 4248, 238-245, 2006.
14. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: A Document-oriented Lookup Index for Open Linked Data. *International Journal of Metadata, Semantics and Ontologies*, Volume 3(1), 37-52, 2008.
15. Page, L., Brin, S., Motwani, R., Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

Centralizované a decentralizované hodnocení kvality webových zdrojů

Martin ŘIMNÁČ[1], Roman ŠPÁNEK[1]

[1]*Ústav informatiky AV ČR, v.v.i.*
Pod Vodárenskou věží 271/2
182 07 Praha 8
[rimnac,spanek]@cs.cas.cz

Abstrakt. Příspěvek se zabývá hodnocením kvality webových stránek. Zatímco klasické metody hodnotí stránky na základě nepřímých měr, článek v souvislosti se sémantickým webem navrhuje použít pro hodnocení obsahu měr přímých založených na porovnávání dat prezentovaných zdroji. Příspěvek navrhuje takové centralizované a decentralizované míry a diskutuje jejich vlastnosti a použitelnost laickou veřejností. Použití takových měr je ilustrováno na reálných webových zdrojích prezentujících výsledky tenisových zápasů.

Klíčová slova: sémantický web, extrakce dat, aktuálnost dat, reputace.

1 Úvod

Hodnocení kvality obsahu webových stránek je jeden ze základních problémů řešených již od samého počátku webových technologií. Zatímco samotné uživatelovo pojetí kvality může být značně subjektivní, různé vyhledávací nástroje používají rozličné nepřímé míry pro ohodnocení obsahu, např. PageRank odvozuje kvalitu od počtu dokumentů odkazujících na hodnocený dokument. Nevýhodou těchto nepřímých měr je, že nikterak nehodnotí (obvykle strojově nečitelný) obsah, jen strukturální vlastnosti dokumentu.
Vize sémantického webu [1] přinesla nové možnosti nahlížení na data a jejich možnou strojovou prezentaci a interpretaci. Základním formátem pro prezentaci dat je jednoduchý formalismus *RDF* (Resource Description Framework) trojic (objekt, vlastnost, subjekt), kde objekt, vlastnost i subjekt mohou být tzv. resource (česky též zdroje[1]) jednoznačně definující nějaký význam. Výhodou použitého formalismu je, že vedle odkazování na resource v jiných dokumentech (analogie hypertextového odkazu) umožňuje i díky konstruktu *@rdf:about* rozšiřovat popis resource z externího dokumentu.
Předpokládejme, že existuje nějaký resource s danými vlastnostmi, který je identifikován pomocí *URI*. Nechť existují dva dokumenty s *URL1* a *URL2* rozšiřující popis resource *URLI* o další vlastnosti či prezentující jiné hodnoty vlastností. V této situaci vyvstává otázka, který z dokumentů prezentuje kvalitnější (aktuálnější a přesnější) data, kterou musí dnes vyřešit sám uživatel. Otázkou je, zda a jak automaticky zkombinovat popisy z obou dokumentů, případně identifikovat případné nekonzistence. Problém je o to závažnější, pokud dokument *URI* je součástí *URL1* (dokument *URL1* definuje resource *URI* pomocí konstruktu *@rdf:ID* a je tedy implicitním zdrojem pro popis resource *URI*).

[1] v příspěvku bude užíváno počeštěného resource tak, aby bylo zamezeno možné záměně se zdrojem (poskytovatelem či dokumentem) dat

Nalezení nejlepšího popisu resource je přitom velmi žádoucí. Vzhledem k distribuované povaze webu mohou zdroje dat odkazující původně na popis resource v dokumentu *URL1* aktualizovat referenci dokument *URL2,* pokud k datům od poskytovatele dokumentu *URL2* začnou chovat větší důvěru (například proto, že dlouhodobě prezentuje kompletnější informace).
Z uvedeného příkladu je vidět, že princip zavedení jednoznačných identifikátorů objektů na sémantickém webu sebou přináší vedle kladných aspektů (snadnější indexace a agregace) i projevy negativní, na které si uživatelé webových technologií musejí zvyknout. Dnes v reálném provozu není překvapením, když dvě různé webové stránky prezentující popis stejného resource uvádějí jinou skupinu vlastností či dokonce si v hodnotách některých vlastností odporují. Softwarový agenti schopni automatického zpracování obsahu takových stránek se s tím musejí, stejně jako uživatelé, relevantním způsobem vypořádat.
Příspěvek proto studuje nové míry a způsoby pro porovnávání webových zdrojů, které by pomocí přímých měr mohly upřednostnění dat některých dokumentů, v obecnějším pojetí zdrojů, prezentující aktuálnější a kompletnější data napomoci.

2 Stav problematiky - systémy pro správu důvěry

Systémy pro budování důvěry se velmi často využívají pro zlepšení nevyhovující situace především v bezpečnosti v obecně distribuovaných prostředích (jako jsou například Servisně orientovaná prostředí [2], Semantické výpočetní gridy [3], [4] multi-agentní společnosti [5], (Sémantický) web [1], výpočetní gridy, peer-to-peer sítě [6]). Základní myšlenkou, která provází systémy pro budování důvěry, je umožnit komunikujícím uzlům vybudovat si síť důvěryhodných vazeb se svým okolím. Pokud jsou takovéto vazby k dispozici, pak mohou být používány pro stanovení, které uzly budou pro komunikaci vhodnější, tedy u kterých je riziko zneužití poskytnutých dat či služeb nejmenší. Systémy pro správu důvěry lze obecně rozdělit na tři základní skupiny:

1. Systémy založené na politikách [7,8,9]. Jejich hlavním cílem je za použití předem definovaných politik umožnit entitám odvozovat míru důvěry. Základním nedostatkem je nutnost používat společný a standardizovaný jazyk pro popis politik a také ne vždy přímočaré a jednoduché stanovení politik.
2. Reputační systémy [10,11]. Reputační systémy odvozují důvěru na základě historického a také současného chování entity.
3. Systémy využívající sociální sítě [12]. Jedná se o způsob řízení důvěry, kdy je k odvození míry důvěry mezi entitami, použita sociální síť a metody pro hledání odpovídajících struktur známých např. z SNA [13] (Social Network Analysis).

Současně lze na systémy pro budování důvěry nahlížet z pohledu způsobu, jakým je důvěra spravována:

- Centralizovaný přístup [14], [15], kde určité jednotky mají zodpovědnost za zprostředkování služeb, resp. za budování důvěry mezi komunikujícími uzly. Na druhou stranu centrální jednotka představuje slabé místo systému.
- Distribuované řízení důvěry [16], [17], [18], [19]. Hlavním přínosem se snaha odstranit slabé místo centralizovaného přístupu – centrální jednotku spravující důvěru a tedy umožnit jednotlivým uzlům budovat si vlastní důvěru na své okolí, či umožnit data o důvěryhodnosti sdílet a tím eliminovat některé formy útoků.

3 Reputace zdrojů

V následující části bude definován nový způsob, jakým lze za pomoci reputačního systému umožnit nalezení důvěryhodnějších (přesnějších a aktuálnějších) dokumentů dat na internetu. Důvěryhodnost dokumentu pak lze odvozovat z jeho reputace, důvěryhodnost zdroje pak z důvěryhodnosti jím prezentovaných dokumentů. Nazveme-li množinu zdrojů prostředím, níže uvedený reputační systém pak spravuje důvěryhodnost v prostředí.
Dokument je možné považovat za důvěryhodný, pokud (rozšíření z [20])

a. sdílí s ostatními dokumenty stejná data,
b. potvrzuje ostatním dokumentům data,
c. ostatní dokumenty validují[2] data prezentovaná dokumentem,
d. neprezentuje nekonzistentní data.

První parametr *a* předpokládá, že zdroje v prostředí zabývající se podobnou problematikou budou popisovat i podobné[3] či stejné resource. Tedy tento parametr obráží stupeň, jak moc daný dokument odpovídá prostředí. Předpokládá se, že míra bude nabývat větších hodnot pro konzervativní dokumenty prezentující základní data, na kterých se shodují i ostatní zdroje v prostředí.
Zatímco první parametr popisuje statické vlastnosti zdroje, další dva parametry *potvrzování* a *validace* jsou ryze dynamické. Dynamika je volena velikostí časového okna, ve kterém je dokument hodnocen. Zatímco parametr potvrzování *b* sleduje dokumenty externích zdrojů, které rychleji reagují na změnu reality (lze je tedy perspektivně využít k přebírání dat), parametr validace *c* hovoří o dokumentech jiných zdrojů, které jsou (následně) schopny data prezentovaná dokumentem (který je prezentoval dříve) potvrdit.
Posledním parametrem *d* je *nekonzistence*, jejímž úkolem je penalizovat dokumenty, pokud prezentují nekonzistentní data. Poznamenejme, že jakákoliv změna reality se v datech projeví nekonzistencí oproti pomalejším zdrojům. Taková penalizace však musí být dočasně kompenzována výše uvedenými dynamickými parametry.

3.1 Centralizovaný přístup

Přístup uvedený v [20] předpokládá, že důvěryhodnost zdrojů je spravována na úrovni prostředí, tedy centralizovaně. Kombinace uvedených parametrů je rezistentní proti mnohým typům útoků [21] a experimentálně bylo ověřeno, že takové hodnocení zdrojů odpovídá subjektivnímu pozorování vlastností zdrojů lidským uživatelem.
Navržený systém bohužel není odolný proti útoku podobnému budování linkových farem u vyhodnocování PageRanku. Tento typ útok je svázán s minimální cenou za replikaci dat na webu. V případě výpočtu důvěryhodnosti dokumentu je scénář útoku následující:

- Mějme dokument *D* presentovaný zdrojem *S*, který sdílí data s ostatními zdroji a požívá vysoké reputace.
- Předpokládejme, že tento zdroj *S* si bude chtít přes dokument *D* svou současnou reputaci svévolně navýšit.
- Z toho důvodu zdroj vytvoří svojí vlastní kopii prezentující kopii dokumentu *D* (případně vytvoří kopii pouze dat, která jsou prezentována výhradně dokumentem *D*, ale nejsou presentována jinými zdroji s vysokou reputací).

---

[2] Pojmem validace budeme v příspěvku myslet skutečnost, že jeden zdroj ověří správnost prezentovaných dat

[3] Podobné z pohledu atributů či sémantiky

- Reputace tohoto nového zdroje nebude vysoká, neboť jeho data nebudou následně potvrzována. Nicméně reputace původního dokumentu bude zvýšena o potvrzená data tímto nově vytvořeným zdrojem.
- Po uplynutí časového okna (kdy podvodné zvýšení parametru vymizí) je vytvořený zdroj zrušen a postup je opakován.

Efekt celého útoku je možné umocnit zvýšením počtu nově vytvářených zdrojů.
Obecná obrana proti tomuto útoku je v centralizovaném prostředí komplikovaná. Lze sice sledovat nově vytvořené zdroje ve zvláštním režimu po dobu delší nežli je časové okno dynamických parametrů, avšak tento způsob lze obejít systematickým budováním několika takových zdrojů, jehož projevy nejsou v prostoru uvedených parametrů odlišitelné od korektního chování reálných zdrojů.

3.2 Decentralizovaný přístup

Přirozenou alternativou k centralizovanému řešení systému pro správu důvěryhodnosti zdrojů je distribuovat správu důvěryhodnosti na úroveň zdrojů. V tomto řešení každý zdroj buduje a následně spravuje svoji vlastní síť důvěry ke spolupracujícím zdrojům, na které odkazuje (zdroj bude motivován odkazovat na relevantní zdroje prezentující (rozšiřující) jím prezentovaná data) a zároveň sledovat zdroje, které odkazují na něj. Prostředí se v tomto případě redukuje na okolí zdroje.
Na rozdíl od centralizovaného přístupu vytvoření nové kopie libovolným zdrojem, který je již hodnotícím zdrojem registrován, nepovede k tíženému efektu (viz předchozí odstavec), neboť hodnotící zdroj nemá žádnou motivaci pro spolupráci s (horší) kopií sebe sama.
Pro potřeby zdroje je distribuovaný výpočet důvěryhodnosti zdrojů ve svém okolí přímočařejší, neboť reputace jsou počítány na základě zdrojem prezentovaných dat, nikoliv proti datům prezentovaných všemi zdroji v prostředí. Na rozdíl od centralizovaného přístupu tedy není vyžadován „jeden centrální etalon pravdy“. Distribuovaný přístup tak více odpovídá filozofii webových zdrojů a přesněji odpovídá i chování uživatelů. I sami různí uživatelé důvěřují některým webovým zdrojům různě.
Decentralizované správa reputace velmi dobře poslouží k relevantnímu doplňování vlastních dat (zejména pokud zdroj ressource definuje pomocí @rdf:ID) daty jiných zdrojů. Uživatelům zdroje je poskytnut nejvýše možný kompletní důvěryhodný popis (což se sekundárně pozitivně projeví v důvěryhodnosti uživatelů v tento zdroj), avšak nesmí to být na úkor správnosti (konzistence) dat (což by uživateli bylo vnímáno negativně). Tedy zdroj sám o sobě se snaží působit preventivně proti útokům.
Používání pouze decentralizovaného přístupu však předpokládá, že uživatel má vybraný zdroj jím požadovaných dat, kterému důvěřuje. Tento předpoklad je splněn pro úzce specializované zdroje (státní správa, tiskové agentury, odborná periodika) nebo obecně uznávané zdroje (Wikipedia).
Uvedený předpoklad však splněn v okamžiku, kdy uživatel data hledá za účelem jejich prvotního zjištění (tj. a priori neví, který zdroj je v této oblasti relevantní). K takovému uspořádání zdrojů však je nutně potřeba centralizované autority. Z tohoto důvodu je vhodné udržovat vedle decentralizované správy a správu centralizovanou anebo se pokusit centralizovaně aproximovat výsledky decentralizovaného přístupu (v experimentální části přezásobením poměrem mezi objemem dat prezentovaných dokumentem/zdrojem proti objemu dat v prostředí).

4 Experimentální část

Návrh parametrů byl experimentálně ověřen na reálných zdrojích, konkrétně na zdrojích prezentující on-line přenosy tenisových zápasů:

- sports.espn.go.com
- uk.eurosport.yahoo.com
- www.livescore.com
- www.tipsport.cz
- sports.yahoo.com

Stav tenisového zápasu byl popsán atributy:

- jméno prvního hráče,
- jméno druhého hráče,
- jména případných spoluhráčů (čtyřhra),
- status (aktuální set, začátek zápasu, finální výsledek, atp.),
- výsledky jednotlivých setů.

Příslušné dokumenty uvedených zdrojů byly periodicky stahovány (minimální perioda stahování byla 2minuty). Vzhledem k tomu, že nekonzistentní data byla rovnoměrně rozložena mezi zdroje (cca 2% z objemu zdrojem prezentovaných dat), jsou na následujících grafech uvedeny pouze průběhy parametrů sdílení, validace a potvrzování. Z těchto parametrů byla vypočtena celková reputace podle

$$r = a * \frac{(1+b)}{2} * \frac{(1+c)}{2}$$

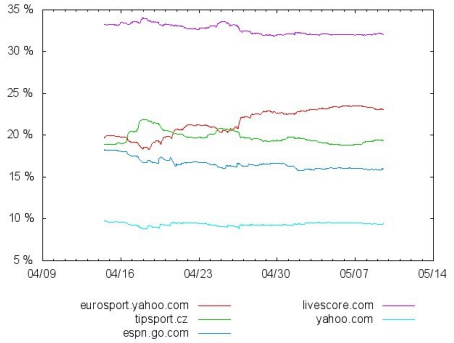
Výpočet je prováděn jak centralizovaně, tak decentralizovaně. Centralizovaný výpočet parametru sdílení je normalizován počtem unikátních trojic v prostředí, u decentralizovaného přístupu pak počtem trojic hodnotícího dokumentu. Parametry validace a potvrzení jsou vztaženy k počtu dokumentem sdílených trojic.

4.1 Centralizovaný přístup

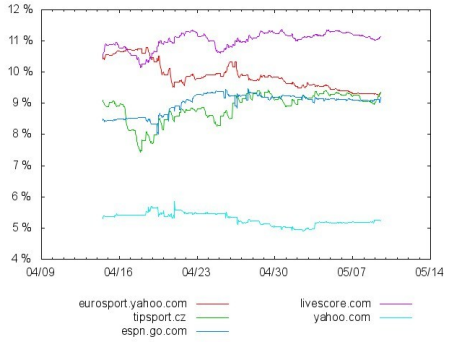
Průběh reputace jednotlivých zdrojů (u každého zdroje byl uvažován pouze jeden dokument, což umožňuje důvěryhodnost dokumentu zobecnit na důvěryhodnost zdroje) při centralizované správě reputace je uveden na Obr. 1.a , Obr. 1.b a Obr. 1.c, ze kterých je následně vypočtena reputace (Obr. 1.d).

Z průběhů vyplývá, že pořadí zdrojů je většinou neměnné, nejlepším zdrojem je označen livescore.com, nejhorším pak yahoo.com (prezentující výsledky offline). V zachyceném časovém okně lze pozorovat souboj mezi eurosport.yahoo.com a tipsport.cz u počtu sdílených dat a mezi tipsport.cz a espn.go.com u potvrzovaných dat. Díky tomu, espn.go.com mnohem více validuje než eurosport.yahoo.com, ve výsledné reputaci je uvedené chování promítnuto do souboje mezi espn.go.com a tipsport.cz.

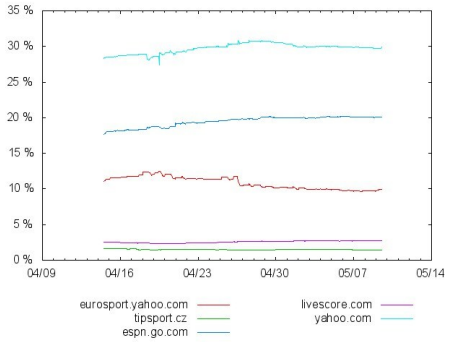
Výsledky experimentu lze interpretovat tak, že uživateli neseznámeného s problematikou tenisových zápasů lze doporučit zdroj livescore.com následovaný zdrojem eurosport.cz. Tento výsledek odpovídá i subjektivnímu, člověkem provedenému, hodnocení.



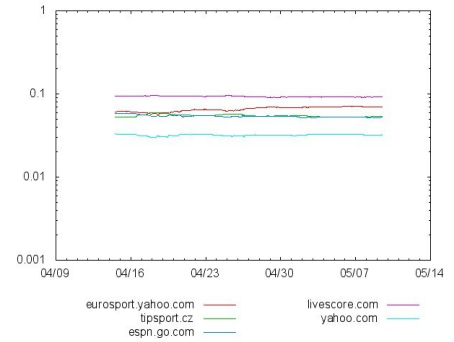
Obr. 1. a Centralizovaný parametr sdílení



Obr. 1. b Centralizovaný parametr potvrzování



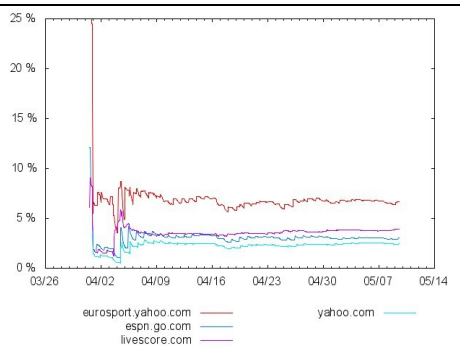
Obr. 1. c Centralizovaný parametr validace



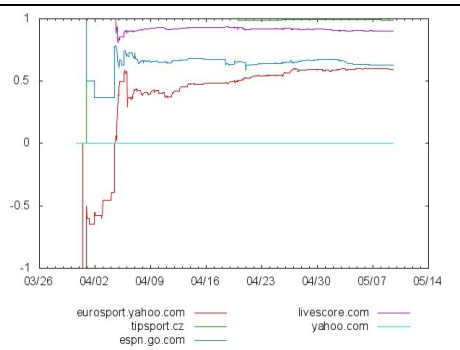
Obr. 1. d Centralizovaný výpočet reputace

4.2 Decentralizovaný přístup

Podobné charakteristiky vykazují i zdroje při použití decentralizovaného přístupu, kdy zdroje počítají charakteristiky proti svým vlastním datům. Tato změna je poměrně dobře patrná na zdroji tipsport.cz (Obr. 2. a ukazující decentralizovaný parametr sdílení), který prezentuje výsledky všech důležitých turnajů, zatímco ostatní zdroje prezentují pouze podmnožinu těchto turnajů.



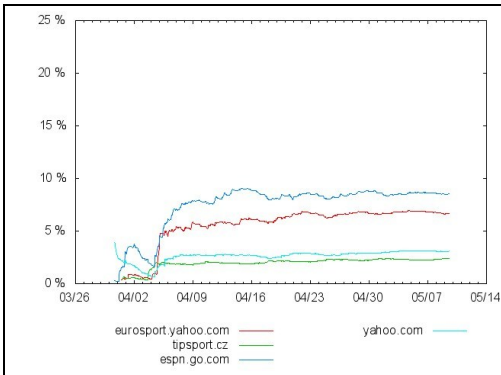
Obr. 2. a tipsport.cz – decentralizované sdílení



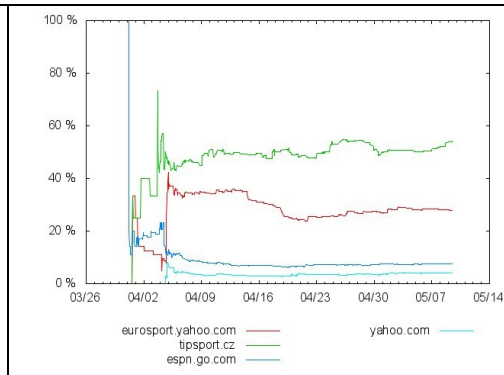
Obr. 2. b yahoo.com – decentralizované vyhodnocení kausality

Z grafů na Obr. 2. b (rozdíl mezi potvrzováním a validací) vyplývá, že i samotný zdroj sports.yahoo.com ví, že data přebírá od ostatních zdrojů (je pomalejší).

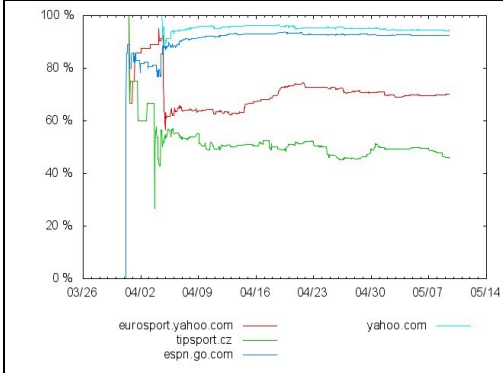
Na Obr. 3. a, Obr. 3. b, Obr. 3. c a Obr. 3. d je uveden průběh decentralizované důvěryhodnosti počítané zdrojem livescore.com (vyhodnocen jako nejlepší z centralizovaného pohledu). U toho zdroje je vidět poměrně velká vazba na zdroj espn.go.com, který mu často data potvrzuje, navíc s ním sdílí hodně dat).



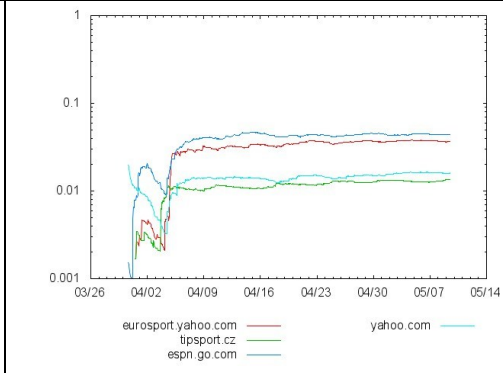
Obr. 3.a livescore.com – decentralizované sdílení



Obr. 3.b livescore.com - decentralizované potvrzování



Obr. 3.c livescore.com - decentralizovaná validace

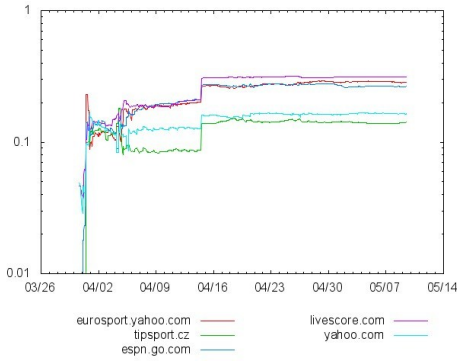


Obr. 3.d livescore.com - decentralizovaná reputace

V případě potřeby použité centralizované reputace je možné agregovat dílčí decentralizované reputace zdrojů. Z porovnání centralizovaného výpočtu a agregovaného výpočtu z decentralizovaných reputací vyplývá, že v tomto konkrétním případě dochází k zachování pořadí zdrojů, avšak jednotlivé poměry mezi reputacemi zdrojů se odlišují. Jedním z důvodů tohoto chování je různý počet prezentovaných dat jednotlivými zdroji.

4.3 Aproximace výsledků decentralizovaného přístupu na centralizovaný

Za předpokladu, že zdroje hodnotí své okolí otevřeně (tedy zveřejňují své vypočtené hodnocení zdrojů), lze aproximovat centralizovanou důvěryhodnost z decentralizované správy např. pomocí průměrování. Takový výsledek zachycuje graf na Obr. 4.



Obr. 4. Odhad centralizované reputace na základě reputací decentralizovaných

Z obrázku plyne, že pořadí lepší poloviny zdrojů zůstává zachováno; pro tyto zdroje je možné centralizovanou důvěryhodnost nahradit za aproximaci důvěryhodnosti získané decentralizovaně. Naopak poměry mezi hodnoceními a hůře hodnocené zdroje vykazují jiné charakteristiky. Toto chování je pravděpodobně způsobeno hůře porovnatelnými charakteristikami zdrojů, potenciálním řešením je obohatit použitý triviální výpočet průměrováním sofistikovanějším způsobem.

Z inženýrského pohledu je však důležité, že pořadí lépe hodnocených zdrojů (což je předmětem otázky) je u obou přístupů shodné.

5 Závěr

Moderní webové zdroje mohou prezentovat data ve strojově zpracovatelném formátu. Vzhledem k tomu, že veškerá taková data jsou veřejně přístupná, i zdroje sami o sobě je mohou využívat například k doplňování svých vlastních dat. Takové chování je však spojeno s nutností udržovat mezi zdroji důvěryhodné vazby.

Článek porovnává možnosti centralizovaného a decentralizovaného výpočtu důvěryhodnosti dokumentů. Zatímco centralizovaný způsob provádí výpočet reputace zdroje proti datům všech zdrojů (tedy ideálně vyžaduje jednu „centrální pravdu“), u decentralizovaného řešení je reputace počítána proti datům, které prezentuje konkrétní zdroj provádějící výpočet svých reputací okolních spolupracujících zdrojů. Návrh centralizovaného přístupu je velmi náročný na robustnost, neboť každý zdroj bude chtít obdržet co největší reputaci. Ochránit takové řešení proti veškerým typům útoků, jak ukazuje praxe z obdobných problémů na webu, je prakticky nemožné, naopak vytváření rozdílných typů útoků díky minimální ceně za kopírování dat je velmi snadné. U decentralizovaného systému tento problém odpadá, neboť by zdroj chybným výpočtem reputace svého okolí poškozoval primárně sám sebe. Navíc váhy jednotlivých kritérií může mít každý zdroj nastaven odlišně.

Decentralizovaný přístup je využitelný pro uživatele, pakliže již sami mají stanoven svůj důvěryhodný zdroj dat - reputace okolních zdrojů spravované uživatelem preferovaným zdrojem pak poslouží k doplnění či ověření dat tohoto zdroje. Naopak centralizovaný způsob se pokouší nalézt nejdůvěryhodnější zdroj, tedy napomoci uživateli s výběrem preferovaného zdroje (je využitelný zejména při hledání informací o neznámých objektech).

Experimentální část příspěvku ukazuje dynamický vývoj jednotlivých sledovaných parametrů u zdrojů prezentující výsledky tenisových zápasů a pokouší se pomocí aproximace získat z distribuovaného pojetí reputace výsledek srovnatelný s centralizovaným řešením. Ukazuje se, že v tomto konkrétním případě pořadí zdrojů hodnocených centralizovaně a agregovaně z distribuovaných výpočtů je podobné, poměry mezi jednotlivými zdroji a hodnocení horších zdrojů jsou však odlišné. V praxi lze tedy agregaci využít pouze jako odhad, reálné nasazení takového přístupu však může být komplikováno faktem, že zdroje nebudou chtít publikovat své vlastnoručně vypočtené reputace.

Literatura

1. Tim Berners-Lee, Dieter Fensel, James A. Hendler, Henry Lieberman, Wolfgang Wahlster: *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potentia*. The MIT Press; New Ed edition, 2005
2. Zaki Malik, Athman Bouguettaya: RATEWeb: Reputation Assessment for Trust Establishment among Web services. The VLDB Journal, Vol. 18, No. 4., pp. 885-1.911, 2009.
3. Shalil Majithia, Ali Shaikh Ali, Omer F. Rana, David W. Walker: Reputation-Based Semantic Service Discovery. Proceedings of the 13th IEEE InternationalWorkshops on Enabling Technologies: Infrastructure for Collaborative Enterprises,pp. 297-302, June 14-16, 2004.
4. Florian Skopik, Daniel Schall, Schahram Dustdar: Modeling and mining of dynamictrust in complex service-oriented systems. Distributed Systems Group, Vienna Uni-versity of Technology, Austria, 2010.
5. Jordi Sabater, Carles Sierra: Reputation and social network analysis in multi-agent systems. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, Bologna, Italy, July 15-19, 2002.
6. Donovan Artz, Yolanda Gil, *A survey of trust in computer science and the Semantic Web, Web Semantics: Science*, Services and Agents on the World Wide Web, Volume 5, Issue 2, Software Engineering and the Semantic Web, June 2007, Pages 58-71, ISSN 1570-8268, DOI: 10.1016/j.websem.2007.03.002.
7. J. Basney, W. Nejdl, D. Olmedilla, V. Welch, and M. Winslett. *Negotiating trust on the grid*. In 2nd WWW Workshop on Semantics in P2P and Grid Computing,New York, USA, may 2004.
8. G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok. *Semantic web languages for policy representation and reasoning: A comparison of kaos, rei and ponder*. In Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, Florida,USA, Oct. 2003.
9. L. Kagal, T. Finin, and A. Joshi. *A policy based approach to security for the semantic web*. In Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, Florida,USA, Oct. 2003.
10. Despotovic Z., Aberer K.: P2P reputation management: Probabilistic estimation vs. social networks, *Computer Networks, Volume 50, Issue 4, Management in Peer-to-Peer Systems, 15 March 2006, Pages 485-500, ISSN 1389-1286, DOI: 10.1016/j.comnet.2005.07.003*.

11. K Hoffman, D Zage, C Nita-Rotaru: *A survey of attack and defense techniques for reputation systems*, ACM Computing Surveys, 2008
12. S. Garfinkel: *Pgp: Pretty good privacy*. O'Reilly & Associates, Inc., 1995.
13. Stanley Wasserman, Katherine Faust: *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*, Cambridge University Press; 1 edition (November 25, 1994)
14. A. Singh, L. Liu: TrustMe: Anonymous management of trust relationships in decentralized P2P systems. IEEE International Conference on Peer-to-peer Computing, pp. 142-149, 2003.
15. Minaxi Gupta, Paul Judge, Mostafa Ammar: A reputation system for peer-to-peer networks. Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video ACM New York, USA, 2003. ISBN: 1-58113-694-3. DOI: 10.1145/776322.776346.
16. Sepandar D. Kamvar, Mario T. Schlosser, Hector G. Molina: The Eigentrust algorithm for reputation management in P2P networks. Proceedings of the 12th International Conference on World Wide Web, pp. 640{651, 2003. DOI: 10.1145/775152.775242.
17. Li Xiong, Ling Liu: PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 7., pp. 843-857, 2004.
18. Xiaomei Liu, Li Xiao: hiREP: Hierarchical Reputation Management for Peer-to-Peer Systems. Parallel Processing, 2006. ICPP 2006. International Conference on Parallel Processing, pp. 289-296, August 2006. DOI: 10.1109/ICPP.2006.48.
19. Jerey Gerard, Hailong Cai, Jun Wang: Alliatrust: A Trustable Reputation Management Scheme for Unstructured P2P Systems, Advances in Grid and Pervasive Computing. Lecture Notes in Computer Science, Volume 3947/2006, pp. 115-125,2006. DOI: 10.1007/11745693 12.
20. Řimnáč, Martin - Špánek, Roman: Hodnocení datových zdrojů pomocí reputačního systému. Datakon 2009. Praha : Oeconomica, 2009 - (Chlapek, D.) S. 117-126. ISBN 978-80-245-1568-7.
21. M. Řimnáč, R. Špánek: *Automated Incremental Building of Weighted Semantic Web Repository*. Published in Foundations of Computational Intelligence Volume 6 Data Mining Series: Studies in Computational Intelligence , Vol. 206 , Abraham, A.; Hassanien, A.-E.; Carvalho, A.P. de L.F. de; Snášel, V. (Eds.), 2009, X, 402 p. 130 illus., Hardcover, ISBN: 978-3-642-01090-3

Annotation:

Distributed Measurement of Web source quality

The paper descries an approach for evaluation of a quality of data provided by data sources on the Internet using a reputation system. The approach is designed to be fully distributed and experiments show that such approach can be used to simplify user decision about which source is more reliable.

Poděkování:

Práce byla podpořena výzkumným centrem: Pokročilé sanační technologie a procesy 1M0554, Ministerstva Ministerstvo školství, mládeže a tělovýchovy České Republiky.

Dátovody a filtre alebo Správca procesov: ktorá integračná architektúra je „lepšia“?

Pavol MEDERLY[1,2], Pavol NÁVRAT[1]

[1]*Ústav informatiky a softvérového inžinierstva FIIT STU v Bratislave*
Ilkovičova 3, 842 16 Bratislava
{mederly,navrat}@fiit.stuba.sk
[2]*Centrum informačných technológií Univerzity Komenského v Bratislave*
Šafárikovo námestie 6, 818 06 Bratislava

Abstrakt. Pri tvorbe integračných riešení sa môžeme stretnúť s rôznymi prístupmi. Pri integrácii s využitím posielania správ sa často používajú architektúry Dátovody a filtre (angl. Pipes and Filters) a Správca procesov (angl. Process Manager). V príspevku využijeme prípadové štúdie získané z literatúry a z vlastnej aplikačnej praxe na porovnanie vlastností týchto dvoch architektúr. Zameriame sa na vlastnosti viditeľné v čase návrhu, ako aj v čase vykonávania.

Klíčová slova: integrácia aplikácií, architektonické vzory, dátovody a filtre, správca procesov.

1 Úvod

Integrácia aplikácií je proces, cieľom ktorého je dosiahnuť, aby nezávisle vyvinuté aplikácie boli schopné zmysluplne spolupracovať. Existujú rôzne prístupy k integrácii; jedným z nich je posielanie správ (angl. messaging), ktorému sa venujeme v tomto príspevku. Vďaka viacerým svojim dobrým vlastnostiam, najmä čo sa týka umožnenia uvoľnenia väzieb medzi integrovanými aplikáciami, je posielanie správ rozšíreným prístupom k integrácii [2].

Dva významné architektonické vzory používané pri tvorbe integračných riešení založených na posielaní správ sú *Dátovody a filtre* (angl. Pipes and Filters) a *Správca procesov* (angl. Process Manager). Hlavný rozdiel medzi nimi spočíva v spôsobe riadenia činnosti integračného riešenia: v prvom prípade je riadenie dané spôsobom usporiadania jednotlivých filtrov a dátovodov; v druhom prípade je riadenie zabezpečované špecializovaným komponentom, ktorý sa nazýva správca procesov.

Každý z uvedených architektonických vzorov významne ovplyvňuje vlastnosti konkrétnych integračných riešení, ktoré sú na ňom založené. Ide jednak o vlastnosti viditeľné v čase vývoja, ako je napr. zložitosť, pochopiteľnosť alebo modifikovateľnosť systému, a tiež o vlastnosti viditeľné v čase vykonávania, ako je napríklad efektívnosť, spoľahlivosť alebo ľahkosť administrácie (správy). Zjednodušene povedané, vzor Správca procesov je výhodný z hľadiska vývoja, kým vzor Dátovody a filtre má viaceré dobré vlastnosti viditeľné v čase vykonávania. V predkladanom príspevku sa prostredníctvom prípadovej štúdie prevzatej z [2] a [5], ako aj štúdie vychádzajúcej z integračného projektu realizovaného na Univerzite Komenského v Bratislave pokúšame zhodnotiť a porovnať vlastnosti riešení založených na týchto dvoch architektonických vzoroch. Porovnávame ich zároveň s prístupom využívajúcim nami vyvinutú metódu MD/CP (z angl. Messaging

Designer using Constraint Programming) umožňujúcu automatizáciu návrhu integračných riešení prostredníctvom spĺňania ohraničení [3, 4], ktorá sa snaží kombinovať dobré vlastnosti správcu procesov v čase vývoja s dobrými vlastnosťami dátovodov a filtrov v čase vykonávania.

Štruktúra príspevku je nasledujúca: V časti 2 stručne predstavíme obe vyššieuvedené architektúry. Časť 3 je venovaná opisu prípadových štúdií. Časť 4 obsahuje výsledky porovnania spolu s diskusiou a porovnaním s podobným experimentom opísaným v [5]. V záverečnej časti uvádzame plánované smery budúceho výskumu.

2 Dátovody a filtre a Správca procesov

2.1 Dátovody a filtre

Ako samotný názov naznačuje, integračné riešenia založené na architektonickom vzore Dátovody a filtre pozostávajú z dvoch druhov prvkov: *Filtre* predstavujú komponenty spracovávajúce správy a *dátovody* sú kanály, ktoré tieto filtre spájajú. Každý filter má množinu vstupných kanálov – jeden, viacero, prípadne aj žiadne – a množinu výstupných kanálov: opäť, jeden, viacero, prípadne žiadne. Kanály môžu byť realizované v operačnej pamäti, prostredníctvom sprostredkovateľa správ (angl. message broker), prípadne ľubovoľným iným vhodným mechanizmom, ako je napr. protokol HTTP alebo SOAP.

Funkčnosť integračného riešenia založeného na tomto architektonickom vzore je teda určená štruktúrou filtrov a dátovodov, ktoré ich spájajú. Jednotlivé filtre sú *autonómne*: každý z nich samostatne spracúva prichádzajúce správy a výsledky spracovania posiela ďalej. V systéme neexistuje centrálny prvok, ktorý by tento tok správ riadil.

V prípade, že sú filtre a spojenia medzi nimi dobre navrhnuté, prináša takéto usporiadanie – v porovnaní s usporiadaním, kedy je celé integračné riešenie sústredené v jednom (monolitickom) komponente – viaceré výhody. Sú to najmä flexibilita riešenia, opakovaná použiteľnosť filtrov, ľahká distribuovateľnosť a paralelizovateľnosť [1, 2, 4].

2.2 Správca procesov

Pri použití architektonického vzoru Správca procesov je činnosť integračného riešenia riadená špeciálnym komponentom, nazývaným správca procesov. Tento komponent vykonáva programátorom vytvorený opis integračného riešenia, zachytený vo forme procesu, resp. procesov. Proces je spravidla definovaný ako množina aktivít a prechodov medzi nimi. Naraz je aktívnych obvykle viac *inštancií procesu*.[1] Pre každú inštanciu procesu správca uchováva aktuálny stav, t.j. informáciu o práve vykonávanej aktivite (aktivitách), hodnoty premenných a podobne.

Typickými aktivitami sú volanie externého komponentu (služby) alebo práca s premennými procesu. Udalosti, ktoré majú za následok vykonanie prechodu medzi aktivitami, sú napríklad ukončenie vykonávania aktivity, prijatie správy, uplynutie určeného času, výskyt výnimky a podobne.

Správca procesov môže byť súčasťou architektúry Dátovody a filtre. V takom prípade si ho môžeme predstaviť ako špecifický druh filtra – prípad vzoru Smerovač správ (angl. Message Router) [2]. S ostatnými filtrami je správca procesov spojený dátovodmi.

[1] Inštanciou procesu sa rozumie jeho konkrétne vykonanie; v prípade procesu spracovania žiadostí o pôžičku je inštanciou napr. spracovanie žiadosti podanej Jánom Malým dňa 20.5.2011 o 14:23.

Všetky dobré vlastnosti architektúry Dátovody a filtre uvedené v časti 2.1 ostávajú zachované: riešenie založené na vzore Správca procesov je flexibilné, komponenty sú opakovane použiteľné, a – za predpokladu vhodných prenosových kanálov – sú volané komponenty ľahko distribuovateľné a paralelizovateľné.

Kľúčovou výhodou vzoru Správca procesov v porovnaní s predchádzajúcim vzorom je fakt, že správca procesov eviduje všetky inštancie procesov, ktoré spravuje. Toto poskytuje nasledujúce prínosy:

- *Jednoduchá správa systému* – správca systému má k dispozícii aktuálny prehľad inštancií jednotlivých procesov, čo je významné pri odhaľovaní problémov, optimalizácii výkonu integračného riešenia a podobne. Zároveň môže správca systému jednotlivé inštancie priamo ovplyvňovať: pozastaviť vykonávanie, zmeniť stav, resp. inštanciu procesu zrušiť.
- *Jednoduché programovanie* – programátor má v ľubovoľnom okamihu k dispozícii celý stav procesu, t.j. všetky údaje, ktoré boli na vstupe procesu, resp. ktoré boli počas jeho vykonávania vytvorené volanými komponentmi.[2] Môže tiež programovo zasahovať do vykonávania procesu, napr. celú inštanciu procesu v prípade potreby zrušiť.

Pri použití vzoru Dátovody a filtre je situácia iná. V tomto prípade nie je jednoduché získať informáciu o inštanciách procesov, ani s už raz spustenými inštanciami manipulovať.

Čo sa týka práce s údajmi, vzor Dátovody a filtre nepozná koncept premennej procesu. Základnou dátovou jednotkou je správa prechádzajúca niektorým z dátovodov. Správa sa najčastejšie skladá z hlavičky, tela a (na niektorých integračných platformách) príloh. Ak programátor potrebuje prenášať údaje medzi filtrami, ktoré nie sú priamo spojené dátovodom, musí zaistiť, aby filtre, cez ktoré správy nesúce tieto údaje prechádzajú, ich nezrušili, prípadne neželaným spôsobom nemodifikovali. Riešenie tohto problému často vyžaduje detailne poznať funkčnosť jednotlivých filtrov a predstavuje významnú časť práce na vytváraní integračného riešenia.

Za nevýhody architektonického vzoru Správca procesov v porovnaní so vzorom Dátovody a filtre sa považujú: [2]

- *Výkonnosť systému* – tým, že správca procesov je centrálny komponent, cez ktorý prechádza prakticky celá komunikácia, predstavuje potenciálne úzke hrdlo obmedzujúce výkon systému ako celku. Použitie tejto architektúry navyše často zabraňuje možnosti využiť efektívne prostriedky vnútroprocesovej komunikácie – táto situácia je názorne zobrazená na obr. 4 a 5.
- *Spoľahlivosť systému* – pri zlyhaní správcu procesov sa zastaví vykonávanie všetkých procesov, ktoré obsluhuje. (V prípade, že správca procesov uchováva stav inštancií procesov v trvalom úložisku, po obnovení jeho fungovania sa vykonávanie procesov obnoví od naposledy uloženého stavu.)

Pri použití vzoru Dátovody a filtre je tento centrálny prvok eliminovaný, t.j. výkonnosť ani spoľahlivosť systému nie je týmto spôsobom obmedzená. Naopak, použitím viacerých nezávislých inštancií filtrov a sprostredkovateľov správ je možné relatívne jednoducho dosiahnuť požadovanú mieru výkonnosti a spoľahlivosti.

Napriek existencii všeobecného konsenzu, že architektonický vzor Dátovody a filtre poskytuje vyššiu výkonnosť integračného riešenia v porovnaní so vzorom Správca

[2] Je to podobná situácia, ako keď má programátor v tradičnom programovacom jazyku k dispozícii všetky lokálne premenné v rámci vykonávania konkrétnej procedúry, funkcie, resp. metódy.

procesov, boli nedávno publikované výsledky experimentov, ktoré naznačujú opak [5]. Tento fakt bol jednou z motivácií pre vykonanie meraní uvedených v tomto príspevku.

3 Prípadové štúdie

Ako prvú prípadovú štúdiu pre porovnanie architektonických vzorov Dátovody a filtre a Správca procesov sme zvolili scenár sprostredkovania pôžičiek (angl. Loan Broker), pôvodne publikovaný v [2] a prevzatý (s úpravami) v [5]. Pre potreby našej štúdie sme ho tiež mierne upravili.

Podstatou scenára je vytvorenie integračného riešenia pre sprostredkovateľa pôžičiek. Úlohou tohto riešenia je spracovávať prichádzajúce požiadavky na pôžičky: zaistiť ohodnotenie bonity klienta (GetCreditRating), zistenie možností pre štátnu podporu („hypotéka pre mladých“, GetSupportPossibilities), získanie ponúk z jednotlivých bánk (PrepareBankXRequest, BankX) a napokon realizovať výber najvýhodnejšej ponuky (SelectBestOffer).

Symbolicky je funkcionalita riešenia zachytená na obr. 1. Použitý je jazyk metódy MD/CP – ide o doménovo špecifický jazyk pre oblasť integračných riešení založených na posielaní správ, ktorého syntax voľne vychádza zo syntaxe jazykov, ako sú C alebo Java.

```
process GetLoanQuote(in Xml request, out Xml offer)
{
    Xml status = execute("Validate.xsl", request);

    if (xpath "$status/Status != 'ok'")
        reject "Validation failed" sending request, status;

    Xml rating = GetCreditRating(request);
    Xml support = GetSupportPossibilities(request);
    fork-and-join parallel
    {
        {
            Xml bank1Request = execute("PrepareBank1Request.xsl",
                request, rating, support);
            Xml bank1Response = execute("Bank1.esbws", bank1Request);
        }
        {
            Xml bank2Request = execute("PrepareBank2Request.xsl",
                request, rating, support);
            Xml bank2Response = execute("Bank2.esbws", bank2Request);
        }
        {
            Xml bank3Request = execute("PrepareBank3Request.xsl",
                request, rating, support);
            Xml bank3Response = execute("Bank3.esbws", bank3Request);
        }
    }
    Xml offer = SelectBestOffer(bank1Response, bank2Response, bank3Response);
    return offer;
}
```

Obr. 1. Požadovaná funkcionalita integračného riešenia pre sprostredkovateľa pôžičiek.

Ako realizačnú platformu sme zvolili Progress Sonic ESB (Enterprise Service Bus) vo verzii 8.0.1, vzhľadom na niekoľkoročné skúsenosti s používaním tohto nástroja.

V Sonic ESB, podobne ako aj pri iných integračných platformách, sú jednotlivé komponenty (hovorí sa im aj integračné služby, resp. jednoducho služby) umiestnené v kontajneroch. Každý kontajner predstavuje proces operačného systému, v rámci ktorého virtuálny stroj jazyka Java vykonáva kód príslušných služieb.

Služby, s ktorými sme pracovali, sme rozdelili do dvoch kontajnerov. Služby, ktoré sú externé vzhľadom k sprostredkovateľovi (t.j. služby bánk – Bank1, Bank2, Bank3) sme umiestnili v kontajneri Banks na vyhradený počítač a sprístupnili ich protokolom HTTP/SOAP. Služby realizované sprostredkovateľom pôžičiek (GetCreditRating, GetSupportPossibilities, SelectBestOffer) sme umiestnili v kontajneri LoansServices a sprístupnili ich prostredníctvom interného komunikačného mechanizmu Sonic ESB. Všetky služby boli len „náhradou" skutočných služieb – namiesto naozajstného spracovania realizovali jednoduchú transformáciu XML dokumentov; toto však považujeme pre potreby porovnania architektonických štýlov za postačujúce.

Následne sme vyššieuvedený proces implementovali tromi spôsobmi:

1. V jazyku BPEL (Business Process Execution Language), ktorý je v súčasnosti štandardným jazykom pre opis integračných procesov. Ako správcu procesov sme použili Progress Sonic BPEL Server 8.0.1. Tento nástroj je vhodný o.i. preto, že je integrovaný s prostredím Sonic ESB, čo mu umožňuje realizovať volanie služieb rovnako efektívne, ako je to v prípade natívnych integračných procesov vytvorených v Sonic ESB (implementácia č. 2).
2. Druhá implementácia využívala architektonický vzor Dátovody a filtre a bola realizovaná v jazyku platformy Sonic ESB.
3. Tretia implementácia využívala metódu MD/CP. Táto metóda pracuje so zdrojovým kódom na úrovni abstrakcie blízkej opisu procesu v prostredí vzoru Správca procesov (t.j. kódom až na detaily totožným s tým, ktorý je uvedený na obr. 1), pričom na základe neho vytvára integračné riešenie s architektúrou Dátovody a filtre. V našom prípade sme generovali kód pre platformu Sonic ESB, t.j. v tom istom jazyku ako implementácia č. 2.

Ako druhú prípadovú štúdiu sme použili jednoduchý integračný scenár z prostredia Univerzity Komenského v Bratislave, ktorého účelom je poskytnúť podklady pre zobrazenie týždenného jedálneho lístka, pričom k dispozícii máme webovú službu poskytujúcu jedálny lístok na jeden deň.

4 Realizované experimenty

4.1 Vývoj integračného riešenia

Náročnosť vývoja integračného riešenia charakterizujeme dvomi spôsobmi: prvým je subjektívne hodnotenie, druhým pokus o objektivizáciu pomocou merania rozsahu vytvoreného kódu. V nasledujúcom opise sa zameriame sa najmä na prvú prípadovú štúdiu – sprostredkovateľ pôžičiek (v prípade druhej štúdie sú výsledky podobné).

Subjektívne ako najnáročnejšiu hodnotíme implementáciu prostredníctvom Sonic ESB, teda použitie architektúry Dátovody a filtre. V tomto konkrétnom prípade je hlavným dôvodom nutnosť zaistiť tok údajov medzi nesusednými komponentmi, napríklad prenos hodnoty premennej „request" na viaceré miesta systému. (Na obr. 2b sú naznačené časti technického riešenia, ktoré bolo potrebné vytvoriť.)

V súlade s očakávaniami sa tento problém nevyskytoval pri architektúre Správca procesov (implementácie BPEL a MD/CP). V prípade implementácie č. 1 boli zdrojom istých komplikácií obmedzenia jazyka BPEL, napríklad to, že volanie externej služby vyžaduje práve jednu vstupnú premennú a jednu výstupnú premennú, t.j. ak má služba viac parametrov, potrebujeme ich v kóde BPEL najprv umiestniť do jednej, na tento účel

vytvorenej premennej. Taktiež vyššia flexibilita tohto jazyka, konkrétne napr. oddelenie údajov o nasadení volaných služieb od ich abstraktného opisu, spôsobila nutnosť vytvárať a používať viaceré konštrukty („partner link“, „port type“, „operation“), ktorých použitie je síce priamočiare a nepredstavuje zdroj významnejších komplikácií, napriek tomu však vytvorenie riešenia spomaľuje. Tento fakt je viditeľný na obr. 2a.

Zápis v jazyku metódy MD/CP sa ukázal ako najjednoduchší. Do istej miery je to dané relatívne úzkym zameraním tejto metódy. Univerzálnosť jej použitia bude ešte potrebné overiť na väčšej množine prípadových štúdií.

Pokúsili sme sa tiež o objektivizáciu týchto pozorovaní. Komplikáciou bol fakt, že prvé dve implementácie využívajú grafické editory – v prvom prípade sú to editory pre jazyk BPEL a pre súvisiace jazyky (WSDL, opis nasadenia, konfigurácia služby), v druhom prípade je to editor procesov pre Sonic ESB a konfiguračná konzola. Tretia implementácia (MD/CP) používa výlučne textový jazyk bez grafickej reprezentácie.

Rozhodli sme sa preto mieru zložitosti vývoja odhadnúť pomocou počtu symbolov, ktoré musí programátor pri vývoji použiť. V textovom jazyku za symboly považujeme napr. kľúčové slová, identifikátory (názvy procesov, dátových typov, premenných), textové reťazce (napr. sieťové adresy, odkazy na súbory a podobne). V grafickom jazyku za symboly považujeme takisto identifikátory, textové reťazce, ale tiež voľbu z viacerých možností, výber nástroja z palety atď. Nepočítame tie symboly, ktoré sú predvyplnené editorom, s výnimkou situácií, kedy ponechanie prednastavenej hodnoty predstavuje explicitné návrhové rozhodnutie. Príklad spôsobu určovania počtu potrebných symbolov je uvedený na obr. 2; hviezdičkou sú označené započítané výskyty symbolov.

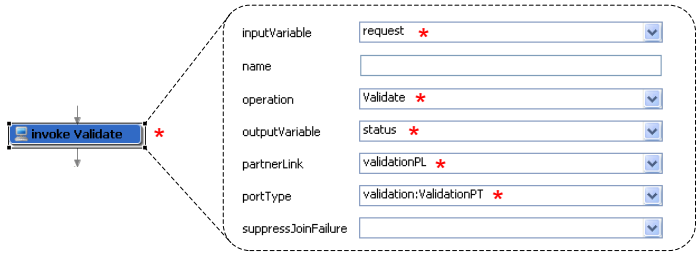
Výsledky sú zhrnuté v grafe na obr. 3. Použité symboly sme rozdelili do troch kategórií:

- *tok riadenia* – sem patrí zaistenie vykonania príslušných služieb a ostatné činnosti ovplyvňujúce spôsob vykonávania integračného riešenia, napr. vetvenie a spájanie toku riadenia;
- *tok údajov* – práca s údajmi v rámci integračného riešenia, napr. deklarácie a používanie premenných (BPEL, MD/CP), práca so správami, resp. ich časťami (ESB);
- *nasadenie* – určenie konkrétnych kontajnerov, vlákien, adries na komunikáciu, kanálov a podobne, potrebných na správnu činnosť integračného riešenia.

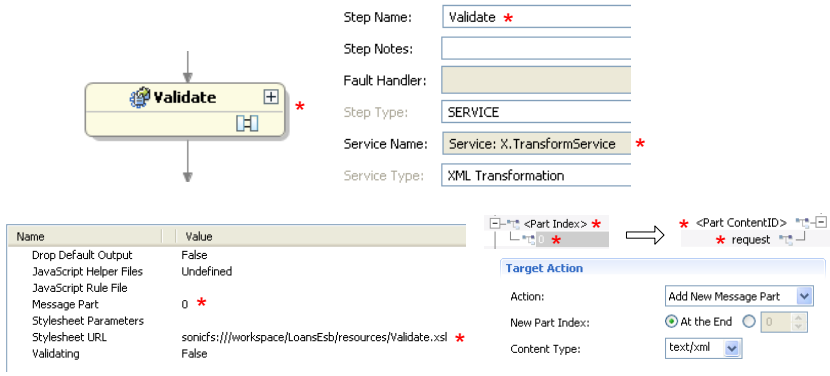
4.2 Porovnanie výkonu

Druhý experiment bol zameraný na porovnanie výkonu, presnejšie povedané priepustnosti jednotlivých implementácií. Použili sme nasledujúce prostredie:

- Jadro integračného riešenia a služby poskytované sprostredkovateľom pôžičiek v kontajneri LoansServices boli prevádzkované na počítači Dell Latitude E6400 s procesorom Intel Core2 Duo P8600, 2,4 GHz, 4 GB operačnej pamäte a operačným systémom Microsoft Windows XP SP3 (32-bit).
- Webové služby bánk boli prevádzkované na počítači s procesorom AMD Athlon64 3400+, 2,2 GHz, 2 GB operačnej pamäte a operačným systémom Microsoft Windows XP SP3 (32-bit).



(a) BPEL (správca procesov)

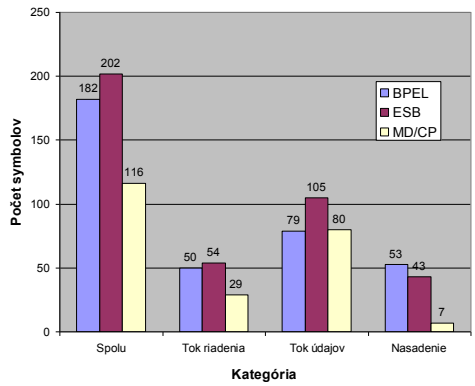


(b) Sonic ESB (dátovody a filtre)

```
Xml status = execute ("Validate.xsl", request);
```

(c) MD/CP

Obr. 2. Ukážka spôsobu počítania symbolov potrebných na vyjadrenie volania služby Validate v jednotlivých programovacích jazykoch.



Obr. 3. Počty symbolov potrebných na implementáciu integračného riešenia v jednotlivých programovacích jazykoch.

Na meranie priepustnosti sme použili soapUI, čiže rovnaký nástroj, aký bol použitý v [5]. SoapUI generuje správy v súlade s určeným profilom (doba trvania testu, počet vlákien, časový interval medzi správami, obsah správ) a meria okrem iného celkovú priepustnosť volanej služby, vyjadrenú počtom spracovaných správ za sekundu.

Použili sme viacero testovacích scenárov, líšiacich sa veľkosťou spracovávaných správ. Každý test bol opakovaný viackrát, pričom do úvahy sme vzali aritmetický priemer jednotlivých výsledkov. Doba trvania jedného testu bola 60 sekúnd. Výsledky experimentu sú zhrnuté v tabuľkách 1 a 2.

Tab. 1. Priepustnosť jednotlivých implementácií pre prípadovú štúdiu Sprostredkovateľ pôžičiek.

	Správy 1 KB	Správy 10 KB	Správy 10 KB [5]
BPEL (správca procesov)	7,2	4,5	55,4
ESB (dátovody a filtre)	23,9	13,0	32,6
MD/CP (dátovody a filtre)	19,6	13,2	-
Pomer ESB / BPEL	**3,32**	**2,89**	**0,59**

Tab. 2. Priepustnosť jednotlivých implementácií pre prípadovú štúdiu Jedálny lístok.

	Správy 1 KB	Správy 10 KB
BPEL (správca procesov)	7,0	3,6
ESB (dátovody a filtre)	16,0	7,6
MD/CP (dátovody a filtre)	9,4	5,4
Pomer ESB / BPEL	**2,29**	**2,11**

Výsledky vykonaných experimentov potvrdzujú všeobecné očakávanie, že použitie integračného vzoru Dátovody a filtre vedie k vyššej efektívnosti než použitie vzoru Správca procesov. Pri štúdii Sprostredkovateľ pôžičiek je dôvod rozdielu zrejmý z obr. 4 a 5 znázorňujúcich prechod správ systémom. Vidíme, že v prípade ESB a MD/CP je väčšina komunikácie realizovaná vnútri kontajnera, t.j. využívajú sa efektívne lokálne volania jazyka Java. V prípade BPEL prebieha pomerne veľa komunikácie medzi kontajnermi.

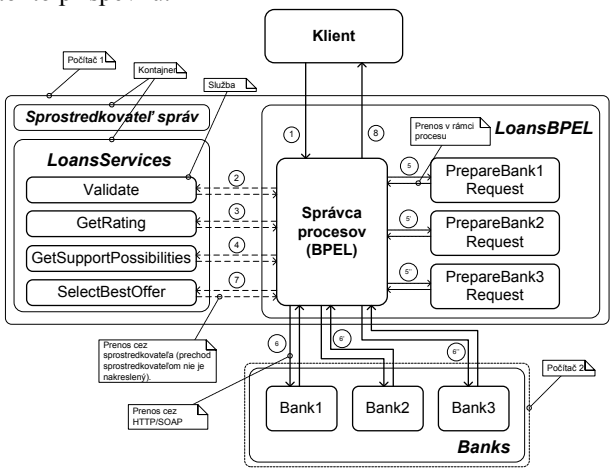
Pri prípadovej štúdii Jedálny lístok boli trasy správ pre oba architektonické vzory rovnaké, rozdiel bol však v spôsobe práce s údajmi. Prejavila sa tu skutočnosť, že jazyk BPEL je orientovaný na prácu so súbormi XML, pričom pri riešeniach implementovaných v Sonic ESB sme mohli využiť efektívnejšiu manipuláciu s údajmi ako s textovými reťazcami. (Tento rozdiel teda nesúvisí s podstatou porovnávaných architektúr, na druhej strane však vypovedá o praktickej použiteľnosti technológií typických pre tieto architektúry).

Nižšia priepustnosť riešenia využívajúceho MD/CP oproti riešeniu v Sonic ESB je daná tým, že pri Sonic ESB sme mohli využiť niektoré črty platformy, ktoré v súčasnosti metóda MD/CP nepodporuje, a tak dosiahnuť vyššiu efektívnosť riešenia.

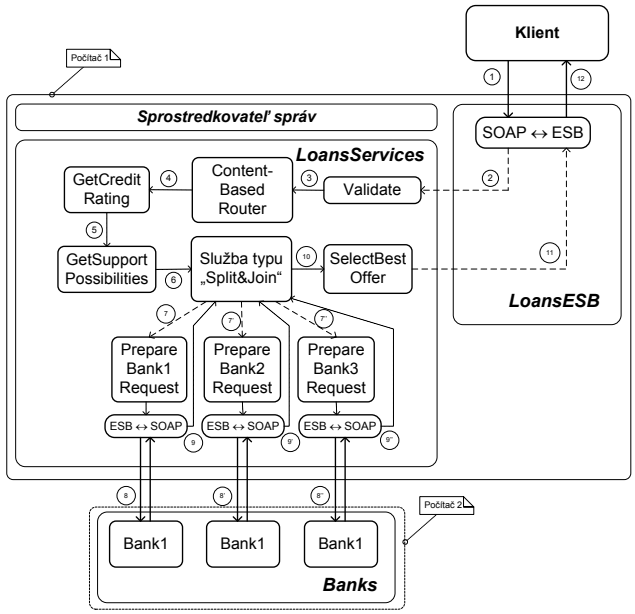
Nevieme, prečo autori [5] prišli k zásadne odlišným výsledkom. Príčina bude pravdepodobne v konkrétnej integračnej platforme, ktorú použili, prípadne v ich integračnom scenári – je možné, že tento scenár neumožnil využiť distribuované spracovanie poskytované architektúrou Dátovody a filtre, a tak do popredia vystúpili rozdiely dané efektívnosťou fungovania použitých infraštruktúrnych produktov (IBM WebSphere ESB a IBM WebSphere Process Server).

Záverom poznamenajme, že výber architektúry integračného riešenia sa spravidla opiera o množstvo ďalších aspektov, nielen o jednoduchosť vývoja a efektívnosť vykonávania

riešenia. (Niektoré z nich sme naznačili v časti 2.) Hlbšie preskúmanie týchto aspektov je však mimo rámca tohto príspevku.



Obr. 4. Tok správ v architektúre Správca procesov v prípadovej štúdii Sprostredkovateľ pôžičiek (implementácie ESB a MD/CP).



Obr. 5. Tok správ v architektúre Dátovody a filtre v prípadovej štúdii Sprostredkovateľ pôžičiek (implementácie ESB a MD/CP).

5 Záver

V príspevku sme prostredníctvom dvoch prípadových štúdií porovnali vybrané vlastnosti architektonických vzorov Dátovody a filtre a Správca procesov. Vykonané pozorovania

potvrdili očakávanie, že riešenia založené na vzore Správca procesov sú jednoduchšie z pohľadu vývoja, avšak menej efektívne počas behu. Zároveň sme demonštrovali, že pri použití metódy MD/CP dokážeme skĺbiť jednoduchosť vývoja s efektívnosťou počas behu.

Ako ukazuje porovnanie s [5], pozorované výsledky sú závislé na použitej integračnej platforme. Preto v budúcnosti plánujeme zopakovať tu uvedené, resp. podobné experimenty aj pre iné integračné nástroje, ako je napr. Apache Camel a Apache ODE (Orchestration Director Engine).

Plánujeme sa tiež ďalej venovať metóde MD/CP. V prvom rade by sme chceli overiť jej vlastnosti na širšej množine integračných problémov z aplikačnej praxe a zistené poznatky využiť na jej ďalší vývoj. Radi by sme tiež zdokonalili spôsob kvantitatívneho určovania náročnosti vývoja integračných riešení, jednak návrhom a realizáciou experimentov vedúcich k zisteniu skutočnej náročnosti, vyjadrenej napr. v človeko-hodinách práce, a tiež nájdením metrík, ktoré by umožnili s dostatočnou presnosťou náročnosť vývoja odhadnúť. Plánujeme tiež overiť vplyv použitia tejto metódy na udržiavateľnosť vytvorených riešení; prvé výsledky v tejto oblasti sú povzbudzujúce.

Poďakovanie: Prezentovaná práca bola čiastočne podporená grantom VEGA VG1/0508/09 a je čiastkovým výsledkom projektu Výskum metód získavania, analýzy a personalizovaného poskytovania informácií a znalostí, ITMS 26240220039, v rámci Operačného programu Výskum a vývoj spolufinancovaného Európskym fondom regionálneho rozvoja.

Literatúra

1. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: *Pattern-Oriented Software Architecture. A System of Patterns.* John Wiley & Sons, Chichester, 1996.
2. Hohpe, G., Woolf, B.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Pearson Education, Inc., Boston, 2004.
3. Mederly, P., Návrat, P.: Construction of Messaging-Based Integration Solutions using Constraint Programming, In: *Advances in Databases and Information Systems*, LNCS 6295, Springer, Heidelberg (2010), 579-582.
4. Mederly, P., Návrat, P.: Automatizovaný návrh integračných riešení založených na posielaní správ, In: *Datakon 2010: Proceedings of the Annual Database Conference,* University of Ostrava (2010), 121-130.
5. Scheibler, T., Leymann, F., Roller, D.: Executing Pipes-and-Filters with Workflows. In: *Proc. of Fifth International Conference on Internet and Web Applications and Services*, IEEE Computer Society, Los Alamitos (2010), 143-148.

Annotation:

Pipes and Filters or Process Manager: which integration architecture is „better“?

There are many approaches to enterprise application integration. When using messaging, the most common architecture patterns are Pipes and Filters and Process Manager. In this paper we use two case studies, taken from the literature and real-life integration projects in order to compare properties of these two styles. We take into account design-time as well as run-time properties, mainly the effort needed to create solutions and resulting solutions’ performance.

Získávání informací o závislostech mezi projekty v softwarových ekosystémech platformy Java

Antonín PROCHÁZKA[1], Mircea LUNGU[2], Karel RICHTA[3]

[1]*Katedra softwarového inženýrství, FIT ČVUT v Praze*
Kolejní 550/2, 160 00 Praha 6
prochan2@fit.cvut.cz

[2]*Institut für Informatik, Universität Bern*
Neubrückstrasse 10, CH-3012 Bern
lungu@iam.unibe.ch

[3] *Katedra softwarového inženýrství, MFF UK v Praze*
Malostranské nám. 25, 118 00 Praha 1
richta@ksi.mff.cuni.cz

Abstrakt. Znalost původu a smyslu zdrojového kódu softwarového ekosystému je kritická jak pro organizaci, která ekosystém vlastní, tak pro vývojáře, kteří pracují na evoluci ekosystému. Modelem řízený vývoj (MDD) a modelem řízená architektura (MDA) obsahují několik důležitých zdrojů informací, které tuto znalost pokrývají. Jedním z nich je popis závislostí mezi projekty a jinými moduly ekosystému. Mnoho způsobů pro popis závislostí mezi projekty existuje již dnes. Přesto se ale během evoluce ekosystému nevyužívají nebo se neudržují aktuální. Tento problém lze vyřešit dvěma způsoby. První je donucení vývojářů k ručnímu udržování těchto informací. To je ale časově náročné, nekomfortní a přivádí to do systému nový zdroj chyb. Druhá možnost je využití reverzního inženýrství. I pro tuto cestu existují hotová řešení. Jedním z nich je model závislostí mezi projekty nazvaný Ecco a jeho sada metod pro reverzní inženýrství závislostí mezi softwarovými ekosystémy založenými na jazyce Smalltalk. Toto řešení vyvinula skupina okolo Mircea Lungu. Náš příspěvek uvádí možnost využití těchto metod pro získávání informací o závislostech mezi projekty v softwarových ekosystémech založenými na platformě Java. Díky rozdílným typovým konvencím v těchto jazycích také můžeme srovnat rozdílnost tohoto přístupu mezi staticky a dynamicky typovanými jazyky.

Klíčová slova: Modelem řízený vývoj, modelem řízená transformace, softwarové ekosystémy, reverzní inženýrství, závislosti mezi projekty.

1 Úvod

Valná většina nástrojů pro modelování a vývoj softwarových projektů se dnes soustředí na jednotlivé projekty. Projekty ale běžně existují v systému, ve kterém spolu spolupracují, vyvíjejí se a tím pádem na sobě navzájem závisí. Každý vývojář pracující na takovém systému přispívá k vývoji alespoň jednoho z jeho projektů a o některých dalších má znalost jako uživatel jejich veřejných rozhraní. Takové systémy nazýváme *Softwarové Ekosystémy*.

Termín ekosystém pochází z biologie, kde je definován jako *základní funkční jednotka v přírodě, ve které jsou v přímém vztahu všechny živé složky s fyzikálními i chemickými*

faktory prostředí[1]. V kontextu softwarového inženýrství definujeme softwarový ekosystém jako *soubor softwarových projektů, které se vyvíjejí a navzájem rozvíjejí ve stejném prostředí [2]*. Příkladem softwarového ekosystému může být společnost s vlastním interním softwarem, open-source komunita, výzkumná skupina atd.

Pohled na software jako na ekosystém odkrývá široké spektrum důležitých informací, které pomáhají jak manažerům vést své softwarové týmy, tak i jednotlivým vývojářům, kteří díky těmto informacím dokáží lépe a snadněji porozumět jejich práci. K tomu pomáhá i možnost soustředit se při získávání informací buď na projekty, nebo i na členy týmů, které se účastní jejich vývoje.

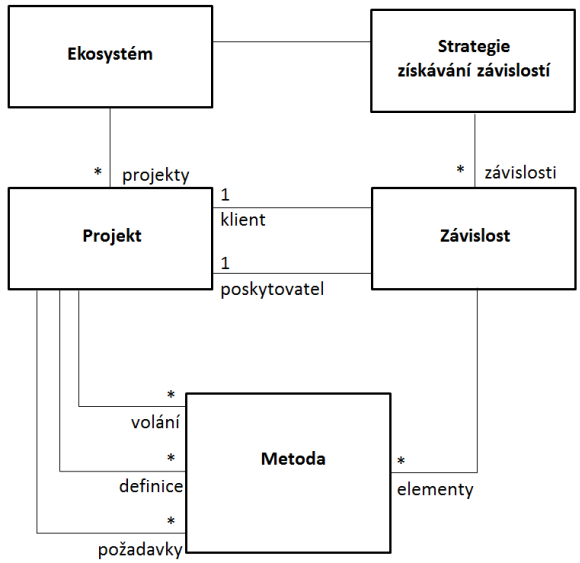
V našem příspěvku se zaměřujeme na analýzu projektů a jejich vztahů uvnitř softwarových ekosystémů. Rozšiřujeme předchozí práci Lungu a spol. [4], která se zaměřuje na získávání informací o závislostech mezi projekty softwarových ekosystémů založených na platformě Smalltalk. Naše práce směřuje k ekosystémům na platformě Java. Lungu a spol. zveřejnili několik pohledů, které pomáhají porozumět softwarovým ekosystémům díky automatickému získávání informací a generování navzájem souvisejících přehledů o jejich struktuře a evoluci. Jeden z jejich nejdůležitějších pohledů je pohled na strukturální závislosti mezi projekty. Vývojáři díky němu mají lepší možnost porozumět aplikačním rozhraním (API) různých projektů, méně duplikují kód a více využívají již hotových implementací a snadněji provádějí refactoring systémů.

Abychom mohli informace o závislostech mezi projekty získat a prezentovat, potřebujeme nejprve v dostupných zdrojích nalézt informace o jejich vztazích. V některých ekosystémech jsou tyto vztahy popsané v konfiguračních souborech. Tyto konfigurační soubory nebo podobné zdroje ale ne vždy existují nebo často nejsou udržovány aktuální. Jedinečným zdrojem informace proto většinou bývá samotný zdrojový kód. Sestavení modelu Ecco z informací dostupných ve zdrojovém kódu probíhá ve dvou fázích. Nejprve se sestaví individuální modely pro každý projekt a následně se tyto modely propojí v jeden. V naší práci bychom chtěli použít stejný nebo obdobný princip pro ekosystémy platformy Java. Jako referenční zdroj informací, proti kterému budeme ověřovat naše výsledky, jsme si zvolili Project Object Model systému Maven. Tomuto projektu přikládáme velkou důležitost, jelikož valná většina informačních systémů je v dnešní době psaná v jazyce Java. Navíc zde máme několik dalších jazyků, které se kompilují pro JVM, což umožní ještě širší využití naší myšlenky. Nalezení spolehlivé techniky pro automatické získávání informací o závislostech mezi projekty je základ pro následný výzkum a vývoj nových technik a nástrojů pro podporu údržby a vývoje softwarových ekosystémů. Cílem tohoto příspěvku je uvést možnost využití metod pro získávání informací o závislostech mezi projekty v softwarových ekosystémech založenými na platformě Java.

2 Model Ecco

V současnosti existuje několik metod pro popis závislostí mezi projekty. Jednou z nich je model Ecco vyvíjený Lungu a spol. [4]. Ecco je velmi odlehčený model vyjadřující čistě jen závislosti mezi projekty. (Viz obr. 1.)

[1] Slovník cizích slov



Obr. 1. Model Ecco

Model Ecco sestává z pěti součástí:

Ekosystém. Ekosystém je v pojetí modelu Ecco množina projektů a jejich závislostí.

Projekt. Pod pojmem *projekt* se v tomto příspěvku rozumí jakýkoli autonomní celek v rámci ekosystému. Např. v ekosystému linuxové distribuce budou projekty nazývány jednotlivé balíky. Přesné vymezení kontextu projektu záleží na povaze konkrétního ekosystému. Projekt se může skládat z dalších částí, modulů. Moduly každého projektu v ekosystému definují *metody* a jiné *metody* volají. Modul může volat vlastní metody nebo jiné metody svého projektu. Stejně tak může volat metody, které jeho projekt nedefinuje. Takové metody nazýváme požadavky. Požadavek v páru s projektem poskytujícím požadovanou metodu vytváří závislost.

Závislost. Existují projekty, které volají metody, jež jim poskytují jiné projekty v ekosystému. Takovému vztahu říkáme závislost. Závislost se skládá ze tří částí. Klient je projekt, který volá metody, které definuje projekt nazvaný poskytovatel. Metody v rámci jedné závislosti se nazývají elementy.

Strategie získávání závislostí. Pro naplnění modelu daty existuje několik metod a postupně přibývají další. Těmto metodám říkáme strategie získávání závislostí.

Tyto strategie můžeme rozdělit do dvou skupin. První skupina je založena na reverzním inženýrství. Právě těmito strategiemi se v naší práci zabýváme. Do druhé skupiny patří strategie, které získávají data z jiných modelů. Tyto strategie nám pomáhají měřit úspěšnost našich metod reverzního inženýrství, protože nám dopředu dokáží prozradit, jaké výsledky bychom měli očekávat.

Strategie pro získávání závislostí jsou přímou součástí modelu Ecco, což nám umožňuje srovnávat různé strategie mezi sebou. V současnosti je implementováno několik metod pro získávání informací o závislostech mezi projekty z ekosystémů založených na platformě

Smalltalk [4]. K nim patří strategie pro získávání informací ze Squeak Universe[2], která těmto strategiím slouží jako referenční.

V našem příspěvku uvádíme základní poznatky, které pomohou vytvořit nové strategie založené na reverzním inženýrství pro získávání informací o závislostech mezi projekty ekosystémů založených na platformě Java.

3 Super-repositáře

Zdrojové kódy a další informace o jednotlivých projektech jsou uchovávány ve verzovaných repositářích. Pro softwarové ekosystémy definujeme pojem super-repositář jako *kolekci všech verzovaných repositářů pro množinu softwarových projektů [3]*. V nich pak hledáme data popisující závislosti mezi projekty.

Některé repositáře obsahují informace o závislostech mezi projekty explicitně. Takové repositáře nám poskytnou referenční informace pro ověření efektivity našich technik založených na reverzním inženýrství. Následně naše ověřené techniky mohou pomoci s udržování těchto explicitních informací v aktuálním stavu. Největší přínos těchto technik ale přijde u projektů, kde tyto informace vůbec dostupné nejsou. Příkladem takového repositáře, který umožňuje explicitně popisovat závislosti mezi projekty založenými na platformě Java, je Apache Maven.

3.1 Apache Maven

Abychom mohli ohodnotit efektivitu našich metod založených na reverzním inženýrství, potřebujeme vědět, jaké výsledky bychom měli očekávat. Z tohoto důvodu jsme si zvolili repositáře Apache Maven jako zdroj dat pro naše případové studie. Maven je nástroj pro vývoj softwaru soustřeďující se na projekty. Jeho datové struktury obsahují rozličné informace ke každému projektu. Tyto informace poskytuje uživatel. Využitím těchto informací tento nástroj obhospodařuje sestavení, reporting a dokumentaci nad projekty [1].

3.2 Project Object Model

Celý Maven je založený na technologii nazvané Project Object Model (POM). Každý projekt má vlastní dokument POM. Dokument POM za pomocí XML popisuje vše relevantní ke svému projektu: vývojáře, kteří na něm pracují, cesty ke zdrojovým souborům, požadované knihovny, kompilátor, správce dokumentace, systém evidence chyb a další. Základní dokument POM s minimem informací může vypadat takto:

```
<project ...>
    <modelVersion>4.0.0</modelVersion>
    <groupId>cz.cvut.fit.swing</groupId>
    <artifactId>my-project</artifactId>
    <version>1.0</version>
</project>
```

V naší práci je nejpodstatnější část tohoto modelu popis závislostí mezi projekty.

[2] Repositář platformy Smalltalk s explicitně definovanými závislostmi mezi projekty

3.3 Popis závislostí mezi projekty

Project Object Model nám mimo jiné poskytuje i explicitní informace o závislostech mezi projekty spravovanými repositářem Maven. Pro Maven je tato informace užitečná během kompilace a sestavování projektů. My z ní můžeme těžit během hodnocení výsledků našich technik založených na reverzním inženýrství.

4 Získávání informací o závislostech z POM

Abychom byli schopni měřit efektivitu technik pro získávání informací o závislostech mezi projekty založených na reverzním inženýrství, potřebujeme vědět, jaké závislosti mezi projekty ve skutečnosti opravdu existují. To je jedna z věcí, kterou nám POM poskytuje. Tyto informace o závislostech mezi projekty spravovanými repositářem Maven musejí být vyčteny z různých částí POM. V následujícím textu popíšeme jednotlivé části POM, které nás zajímají a vysvětlíme jak jejich informace zužitkovat k našemu prospěchu.

4.1 Závislosti

Pokud jeden projekt přímo závisí na jiném, je tato informace zaznamenána v části Dependencies. Tato informace se pak nachází v dokumentu POM toho projektu, který požaduje nějakou funkcionalitu, tedy z pohledu modelu Ecco v klientském projektu. Každá závislost je popsána několika atributy – viz příklad.

```
<dependencies>
        <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.0</version>
                <type>jar</type>
                <scope>test</scope>
        </dependency>
        ...
</dependencies>
```

GroupID, ArtifactID, Version. Tato trojice přesně určuje jaký projekt a v jaké jeho verzi vyžadujeme. Využijeme jí jako jednoznačný identifikátor klientského projektu.

Type. Typ je název balíku, ve kterém je obsažen projekt poskytující požadovanou funkcionalitu. Může to být název libovolné technologie – neexistuje žádný omezený výčet. V naší práci se budeme zajímat o balíky Java Archive (JAR). Tento atribut tedy využijeme jako filtr.

Scope. Tento atribut dělí závislosti do 5 skupin. Každá skupina nám zároveň poskytuje informaci o tranzitivitě popisované závislosti. Transitivní závislost znamená, že pokud klient A vyžaduje projekt B, který vyžaduje poskytovatele C, bude C společným poskytovatelem pro A i pro B a tím pádem bude existovat i závislost mezi A a C.

Compile Scope je výchozí skupina reprezentující běžné projekty, které jsou dostupné včetně zdrojového kódu a jsou potřebné pro úspěšné sestavení klientského projektu. Závislosti v této skupině *jsou tranzitivní*.

Test Scope je obdoba předchozí skupiny, ale její projekty jsou potřebné pouze k testování systému. Stejně tak tyto závislosti *jsou tranzitivní.*

Provided Scope je skupina předkompilovaných projektů, které jsou dodané v rámci SDK (Software Development Kit), kontejneru nebo jinou cestou. Tyto závislosti *nejsou tranzitivní*.

Runtime Scope je obdoba předchozí skupiny, ale dodání poskytovatelů je očekáváno až za běhu. Stejně tak tyto závislosti *nejsou tranzitivní*.

Systém Scope je další obdoba, ale vyžaduje explicitní dodání požadovaných závislostí. Tyto závislosti také *nejsou tranzitivní*.

Protože budeme pouze projekty obsažené v daném ekosystému, budeme se zajímat pouze o projekty z první skupiny, případně z druhé, pokud budeme chtít náš záběr rozšířit i na testovací projekty.

4.2 Výjimky

Transitivní závislosti mohou způsobovat nechtěné chování. Abychom tomu zamezili, můžeme v POM specifikovat výjimky, čímž tyto závislosti eliminujeme. Každá výjimka je definována dvěma již známými atributy *GroupID* a *ArtifactID*.

```
<dependencies>
        <dependency>
                <groupId>org.apache.maven</groupId>
                        <artifactId>maven-embedder</artifactId>
                        <version>2.0</version>
                        <exclusions>
                                <exclusion>
                                <groupId>org.apache.maven</groupId>
                                <artifactId>maven-core</artifactId>
                        </exclusion>
                </exclusions>
        </dependency>
        ...
</dependencies>
```

Význam výjimek pro náš případ je zřejmý. Tyto výjimky musíme brát v potaz a závislosti, jichž se týkají, neuvažovat.

4.3 Dědičnost

POM nám umožňuje uspořádat projekty do stromové struktury. Každý potomek potom dědí vlastnosti svého předka včetně závislostí, pokud taková vlastnost není v potomkovi předefinována. Potomek může vypadat např. takto.

```
<project ...>
    <modelVersion>4.0.0</modelVersion>
    <parent>
            <groupId>cz.cvut.fit.swing</groupId>
            <artifactId>my-parent</artifactId>
            <version>2.0</version>
            <relativePath>../my-parent</relativePath>
    </parent>
    <artifactId>my-project</artifactId>
</project>
```

Dědičnost je pro nás zajímavá ze dvou důvodů. Za prvé samotná dědičnost vyjadřuje závislost a tak o ní i musíme uvažovat. Za druhé se závislosti předka stávají i závislostmi potomka, pokud není v potomkovi řečeno jinak.

4.4 Agregace

Sestává-li projekt z několika modulů, jsou tyto moduly spravovány Mavenem jako separátní projekty a následně jsou agregovány do tzv. Multi-Module projektu, který může vypadat takto.

```
<project ...>
    <modelVersion>4.0.0</modelVersion>
    <groupId>cz.cvut.fit.swing</groupId>
    <artifactId>my-multi-module</artifactId>
    <version>2.0</version>
    <modules>
        <module>my-module</module>
        <module>another-module</module>
    </modules>
</project>
```

Agregace budeme z našeho pohledu považovat za jinou formu závislosti.

4.5 Super-POM

Aby nebyl každý dokument POM příliš obsáhlý, definuje se tzv. Super-POM dokument, který obsahuje informace společné pro všechny ostatní POM dokumenty. Ve výchozím Super-POM dokumentu se informace o závislostech, výjimkách, dědičnosti ani agregaci nevyskytují. Super-POM je ale konfigurovatelný, takže je třeba vždy nejprve načíst jeho informace.

5 Získávání závislostí z JAR

Cílem tohoto příspěvku je popsat základní principy potřebné pro získávání informací o závislostech mezi projekty softwarových ekosystémů založených na platformě Java. Nejprve je třeba získat informace o typech referencovaných ve zkoumaných projektech. Následně s touto informací pracujeme, abychom získali konečný výsledek.

5.1 Java Bytecode

Softwarové ekosystémy založené na platformě Java mohou sestávat z projektů napsaných v různých jazycích. Stejně tak nemusíme mít vždy k dispozici zdrojové kódy. Vždy ale máme k dispozici společný základ – Java Bytecode. Ten nám dá dostatečný přehled bez ohledu na původní programovací jazyk a to i bez dostupného zdrojového kódu. Každá třída platformy Java je kompilována do Java Bytecodu – do souborů s příponou .class a zabalena spolu s ostatními do Java Archivu – souboru .jar, což je klasický komprimovaný balík [6].

Soubor .class poskytuje dostatečné množství informací, abychom byli schopni vyčíst jak definované metody, tak jejich volání. To lze ukázat na příkladu. Mějme následující třídu napsanou v jazyce Java.

```
import java.awt.*;
import java.applet.*;

public class DocFooter extends Applet {
        String date;

        public void paint(Graphics g) {
                g.drawString(date + " by ",100, 15);
        }
}
```

Pokud tuto třídu zkompilujeme (např. příkazem javac) a následně „disassemblujeme“ (např. příkazem javap), získáme následující výstup.

```
public class DocFooter extends java.applet.Applet {
        java.lang.String date;
        public DocFooter();
        public void paint(java.awt.Graphics);
}
```

Takto získáme plně kvalifikovaný název každého typu použitého v deklaraci třídy. Dále potřebujeme znát také typy použité v definicích metod. Výpis příkazu javap s parametrem c pak pro metodu paint vypadá takto.

```
public void paint(java.awt.Graphics);
        Code:
        0:      aload_1
        1:      new #8; //class java/lang/StringBuilder
...
        32:     getfield #7; //Field email:Ljava/lang/String;
        35:     sipush  290
        38:     bipush  15
        40:     invokevirtual #13;
//Method java/awt/Graphics.drawString:
(Ljava/lang/String;II)V
        43:     return
}
```

I když jde pouze o obdobu assembleru, z uvedených komentářů můžeme vyčíst vše potřebné. Nyní máme veškeré informace pro provedení našich metod pro naplnění modelu Ecco založených na reverzním inženýrství.

5.2 Metody založené na reverzním inženýrství

Jak již bylo řečeno, máme nyní k dispozici několik metod pro získávání informací o závislostech mezi projekty softwarových ekosystémů založených na platformě Smalltalk. Protože jsou Smalltalk i Java objektově orientované platformy, můžeme se pokusit využít znalostí získaných z platformy Smalltalk na platformě Java. Pokud to bude třeba, poskytne nám práce s existujícími technikami zkušenosti k vytvoření nových technik, které by více reflektovali specifika platformy Java.

Vlastní výzkum jednotlivých metod je předmětem naší další práce. Na první pohled se podobné metody mohou zdát poměrně přímočaré, ale z výzkumu Lungu a spol. je patrné, že o netriviální úkol.

6 Vyhodnocení výsledků

I když pracujeme s exaktními informacemi, získávání informací o závislostech mezi projekty v softwarových ekosystémech je stále empirická disciplína. Proto potřebujeme nástroj pro měření a srovnávání našich výsledků. V našem případě využijeme známé míry pro získané informace *přesnost*, *úplnost* a *míra F* [5], které pro náš případ přizpůsobil Lungu a spol. [4].

Díky informaci získané z POM můžeme získané informace rozdělit na relevantní, resp. nerelevantní, které jsou, resp. nejsou obsaženy v POM. Informace získané metodami založenými na reverzním inženýrství rozdělíme na získané a nezískané. Tím získáváme 4 statistické kategorie, jak je vidno z tab. 1.

	Relevantní (TP ∪ FN)	Nerelevantní (FP ∪ TN)
Získané (TP ∪ FP)	Pravdivě pozitivní (TP)	Falešně pozitivní (FP)
Nezískané (FN ∪ TN)	Falešně negativní (FN)	Pravdivě negativní (TN)

Tab. 1. Statistické množiny získaných informací

Metriky jsou pak definovány dle (1). Přesnost (P) je poměr počtu získaných závislostí, které jsou relevantní, úplnost (R) je poměr počtu relevantních závislostí které jsou získané. Míra F je vážený harmonický průměr přesnosti a úplnosti a vyjadřuje celkovou efektivitu měřené metody.

$$P = \frac{|TP|}{|TP \cup FP|} \qquad R = \frac{|TP|}{|TP \cup FN|} \qquad F = \frac{2PR}{P+R} \tag{1}$$

Srovnání hodnot těchto metrik pro každou metodu nám následně dá celkový přehled o jejich efektivitě.

7 Závěr

V úvodu jsme přiblížili pojem softwarový ekosystém. Zmínili jsme jeho definici a význam. Také jsme uvedli informaci k předchozí práci Lungu a spol., která je základem pro náš výzkum zaměřený na platformu Java.

Z naší studie je zřejmé, že pokus o znovupoužití technik pro získávání informací o závislostech mezi projekty platformy Smalltalk pro platformu Java má smysl. Nalezli jsme zdroj informací, jak pro tyto metody, tak pro metriky na měření efektivity těchto metod.

Znalosti uvedené v tomto příspěvku nám dávají solidní základ pro náš další výzkum. Víme kam a jak uložit informace, odkud je získat a jak zjistit, zda jsou získané informace správné. Informace umíme získat z projektů napsaných v libovolném jazyce kompilovatelného do Java Bytecodu a to i bez dostupnosti jejich zdrojového kódu. Díky informacím získaným z výzkumu Lungu a spol. také budeme moci srovnat rozdíly v technikách mezi staticky a dynamicky typovanými jazyky. Náš další výzkum se nyní

bude zabývat jednotlivými metodami pro získávání informací o závislostech mezi projekty softwarových ekosystémů založených na platformě Java.

Poděkování

Rádi bychom poděkovali za finanční podporu grantu Studentské Grantové Soutěže ČVUT v Praze v rámci projektu „Modelování, extrakce a procházení závislostí v softwarových ekosystémech jazyku Java“, číslo grantu SGS11/086/OHK3/1T/18. Částečně byl rovněž podpořen grantem GAČR 201/09/0990.

Literatura

1. Apache Foundation: *Maven Project*. http://maven.apache.org/, 2002.
2. Lungu, M.: *Reverse Engineering Software Ecosystems*. Ph.D. thesis, University of Lugano (2009)
3. Lungu, M., Lanza, M., Girba, T., Heeck, R.: Reverse engineering super-repositories In: Proceedings of the 14th Working Conference on Reverse Engineering. pp. 120-129. IEEE Computer Society, Washington, DC, USA (2007)
4. Lungu, M., Robbes, R., Lanza, M.: Recovering inter-project dependencies in software ecosystems. In: Proceedings of the IEEE/ACM international conference on Automated software engineering. pp. 309-312. ASE '10, ACM, New York, NY, USA (2010), ACM ID: 1859058
5. Manning, C., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press New York, NY, USA (2008)
6. Oracle: *Java SE Documentation*, http://download.oracle.com/javase/6/docs/, February 2010

Annotation:

Retrieving Inter-Project Dependencies from Java-Based Software Ecosystems

Understanding the legacy of code in a software ecosystem is critical for the organization that is the owner of the ecosystem as well as for individual developers that work on particular systems in the ecosystem. Model driven development (MDD) and model driven architecture (MDA) techniques contain several important categories of information which cover this knowledge. One of them is description of how do projects or other modules of our software ecosystem depend on each other. Today we have many ways to describe these inter-project dependencies that help us maintain this information. On the other hand there's none of them used or it's not updated by anyone during an evolution process. There are two solutions in this case. Developers can describe the dependencies by hand. This can be painful and error prone process. Another solution is recovering the dependencies using some reverse-engineering process. There are some existing technologies today. One of them is an Ecco model of inter-project dependencies with a set of methods for recovering the dependencies from Smalltalk based software ecosystems developed by Lungu et al. This paper discusses a potential of this technology for recovering inter-project dependencies from Java based software ecosystems.

Geoinformatické modelování rozpukaných oblastí

Blanka MALÁ[1], David TOMČÍK[2]

[1]*Ústav nových technologií a aplikované informatiky, FM TUL*
Studentská 2, 461 17 Liberec
blanka.mala@tul.cz

[2]*Ústav nových technologií a aplikované informatiky, FM TUL*
Studentská 2, 461 17 Liberec
david.tomcik@tul.cz

Abstrakt. Geoinformatické modely rozpukaných oblastí jsou požadovaným vstupem do hydrogeologických a transportních modelů. Tyto matematické modely jsou využívány např. pro řešení scénářů kontaminace podzemních vod z úložišť nebezpečného vysoce aktivního odpadu. Pro tyto hydrogeologické modely je vždy nutné zpracovat reálná geologická a hydrogeologická data modelované oblasti, vytvořit tzv. geometrický model, modelovou síť a tyto naplnit daty. Model oblasti, ve kterém chceme kromě tvarů povrchu a horninového složení vystihnout také zlomy a rozpukání hornin ve vertikálním směru a vrstevnatost či rozpukání horizontálního průběhu, je velmi komplexní a složitý. Lze modelovat v různém rozlišení a podrobnosti, která se promítá buď do geometrického modelu nebo do modelové sítě, vzniká tedy celá množina modelů navázaných na zdrojová data. Proto byla vytvořena metodika pro předzpracování dat, postupy transformace vstupů do geometrického modelu a modelové sítě, naplnění modelové sítě daty. Zpracování reálných dat o území vyžaduje nasazení GIS a přístupů geoinformatického modelování společně se speciálními aplikacemi pro propojení GIS a generátoru sítí. Tyto aplikace jsou vytvořeny pro automatizaci celého procesu generování složitých modelů rozpukaného horninového prostředí. Využívají databáze a nástroje GIS. Byla vytvořena rovněž samostatná aplikace pro generování geometrických modelů a jejich zápis v kódu generátoru 3D sítí. Článek vysvětluje základní pojmy pro pochopení problematiky, postupné kroky tvorby modelů rozpukaných oblastí, vysvětluje realizované automatizované řešení pomocí aplikace Convert2Geo významně doplněné o algoritmy pro zpracování složitých datových struktur vystihujících speciální geologické charakteristiky území. Zásadním krokem byla realizace organizace dat, která umožnila aplikovat vlastnosti modelovaných objektů již na úrovni geometrického modelu. Tyto vlastnosti pak přebírá generovaná 3D síť, kterou je pak možné operativně naplňovat daty potřebnými pro následné matematické simulace.

Klíčová slova: modelování, geometrický model, modelová síť, geologické charakteristiky, GIS, Convert2geo

1 Úvod

Geoinformatické modely se uplatňují při studiu procesů v horninovém prostředí. Ty jsou studovány v souvislosti s využitím geologického prostředí pro ukládání nebezpečného odpadu, např. vyhořelého jaderného paliva. Pro řešení scénářů vývoje kontaminace podzemních vod při úniku nebezpečných látek z průmyslu, skládek atd. se používají hydrogeologické a transportní modely [4]. Specifická je situace při modelování podzemních úložišť nebezpečných vysoce aktivních odpadů, kdy je uvažováno úložiště v granitických

horninách, které mají ovšem složitou geologickou strukturu s výskytem puklin, puklinových zón na rozhraní bloků, také puklin malého rozsahu a mikropuklin. Tyto však vytváří potenciální cesty pro migraci kontaminace z úložiště a jsou podstatné pro analýzu.

V daném území nás proto zajímají veškeré objekty a jevy na povrchu i pod zemským povrchem, které mají přímý nebo nepřímý vliv na proudění vody v podzemí i na povrchu, na jeho rychlost a směr, na celkovou bilanci vody v území, na možnost šíření kontaminace, která vzhledem ke konfiguraci území a vlastnostem proudění vznikne. Tyto charakteristiky jsou zpracovány do speciálního geoinformatického modelu území (tvořeného geometrickým modelem a modelovou sítí), který je nezbytnou součástí hydrogeologických a transportních modelů.

Pro zpracování veškerých dat popisujících zájmovou oblast do přesně definovaných formátů a struktur jsou využívány geoinformační technologie. Je vytvořen geometrický model a modelová síť, které nesou všechny potřebné charakteristiky území. Tyto jsou dále naplněny daty definujícími chování objektů v hydrogeologickém modelu. Hydrogeologické a transportní modely jsou založeny na výpočtech pomocí metody konečných prvků a vyžadují na vstupu geometrické modely a modelové sítě, které speciálním způsobem diskretizují zájmové území. Uvažujeme plně 3D modely (objemové), proto pracujeme nejen v GIS (geografický informační systém) softwarech, ale i programech pro objemové modelování společně s vyvíjenými aplikacemi pro specifické úlohy.

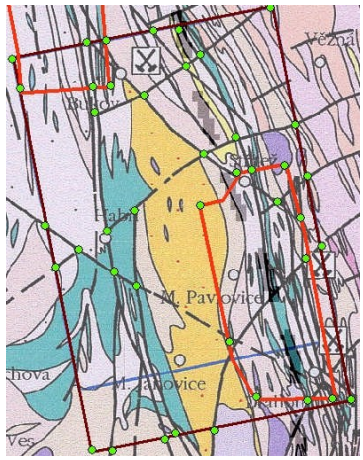
2 Charakteristika modelovaných území

V dalším textu se budeme zabývat výstavbou geoinformatických modelů (geometrických modelů a modelových sítí). Soubory geometrického modelu a modelové sítě fungují jako vstupní soubory do hydrogeologických a transportních modelů. Jsou nezbytné pro konkrétní výpočty v simulované oblasti. Pro výpočty pro danou zájmovou oblast se vždy musí vytvořit geometrický model a modelová síť této oblasti, která respektuje geologické a geografické charakteristiky daného území. Jakkoli jsou zájmová území odlišná, postupy výstavby geometrických modelů a modelových sítí a jejich naplňování daty lze zobecnit, vytvořili jsme postupy a metodiku, jak tyto modely vytvářet efektivně a automaticky.

Modelované území je vždy vymezeno předem. Je dána jeho hranice horizontální (např. na mapě), ale také je vymezeno do hloubky. Zájmové území je popsáno pomocí různorodých dat. Jsou k dispozici mapy v analogové a digitální formě, data týkající se hydrologie území - vodních toků, povodí, průběhu rozvodí, dále data popisující povrch reliéfu. Je rovněž nutno zpracovat geologická data - horninové složení, tektonické oblasti, puklinové zóny, rozhraní hornin, fyzikální vlastnosti hornin, vlastnosti puklin v horninovém prostředí, nutno zahrnout hydrogeologická data popisující studny, prameny, geologické vrty, výšky hladiny podzemní vody a potřebné výsledky měření, důlní mapy, geologické profily a data v tabulkách či databázích, která popisují oblast pomocí atributů vybraných objektů a jevů v území. Nezřídka je nutné zařadit rovněž data týkající se socioekonomických aktivit způsobujících kontaminace povrchu či data o objektech a jevech, které mohou být kontaminací ovlivněny.

Na modelovaném území nás zajímá horninové složení, průběh rozhraní jednotlivých druhů hornin, průběh puklin vertikálních i horizontálních, horizontální rozvrstvení hornin, konfigurace povrchu reliéfu, říční síť, výskyt pramenů, studní, poloha uskutečněných geologických vrtů (Obr. 1). Tyto jevy mají vliv na směr, rychlost a změny směru a rychlosti proudění vody v oblasti. Pro transportní úlohy se musíme zabývat také zdroji určitého typu kontaminace. Geometrické charakteristiky zájmové oblasti jsou následně zapracovány do

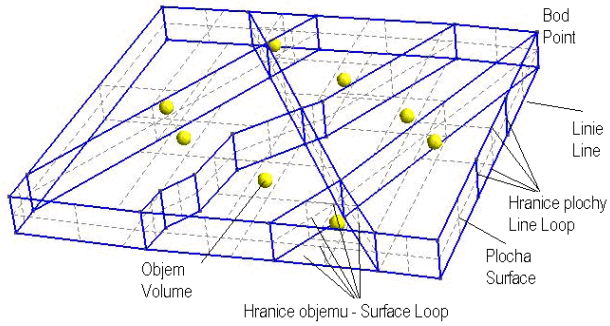
geometrického modelu a modelové sítě, atributy (vlastnosti objektů) pak slouží k naplnění geometrického modelu a modelové sítě daty.



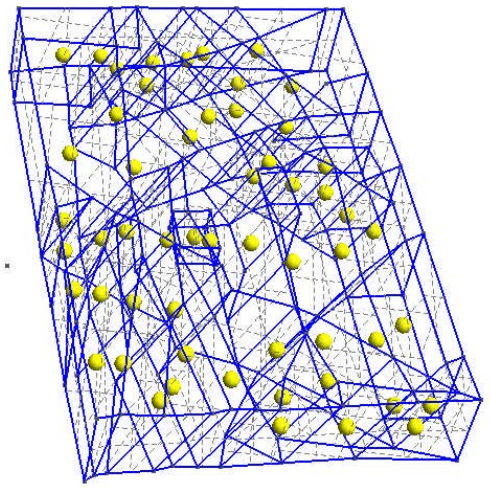
Obr. 1. Příklad vstupních dat – geologická mapa, puklinové zóny, vymezení zájmové oblasti.

3 Geometrické modely a modelové sítě

Geometrický model popisuje modelovanou oblast z pohledu geometrie území. Jedná se o objemový 3D model. Je složen z elementů geometrického modelu: 0D – bod, 1D - linie, 2D – rovina, 3D - objem. 1D elementy mohou být i složité – spline, 2D elementy mohou být také např. přímkové plochy, části povrchu koule, atd. Každý geometrický element má speciální definice ve zdrojovém kódu geometrického modelu. Nejblíže tomuto chápání geometrického modelu je drátový model, z toho důvodu používáme také CAD přístup k tvorbě modelů (Obr. 2). Geometrické modely vytvářené na základě reálných dat jsou obvykle složité. Např. blok rozpukané horniny s puklinami jak vertikálního tak horizontálního průběhu. Tyto pukliny rozdělují jediný blok horniny na mnoho objemů (Obr. 3).

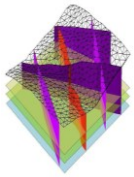


Obr. 2. Geometrický model a jeho elementy.

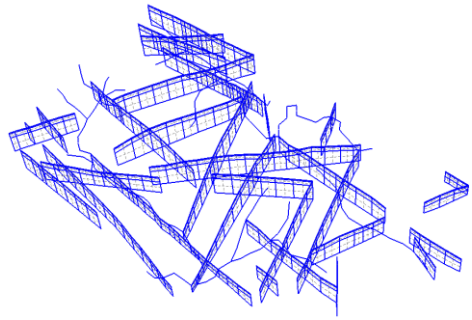


Obr. 3. Objemy geometrického modelu definované podle rozpukání oblasti.

Dále je geometrie oblasti ovlivněna dalšími jevy - konfigurací terénu, říční sítí, průběhem hranic povodí, polohou hladiny podzemní vody a dalšími. Které objekty a jevy budou zahrnuty do geometrie modelu závisí na účelu modelu, na požadovaném rozlišení a přesnosti modelu [2]. Geometrický model neurčuje pouze dekompozici na dílčí objemy a povrchy v prostoru, ale také definuje jejich množiny v souladu s jejich fyzikálními vlastnostmi. Tedy jeden tzv. fyzikální objem je tvořen množinou všech objemů, které mají danou vlastnost důležitou pro budoucí matematické modelování (vlastnost je např. druh horniny, propustnost dané horniny, porozita). Stejně tak definujeme fyzikální plochy (např. podle vlastností rozhraní hornin, charakteristik puklin, aj.) nebo fyzikální linie (vodní toky, rozvodí, aj.) (Obr. 4 a Obr. 5).



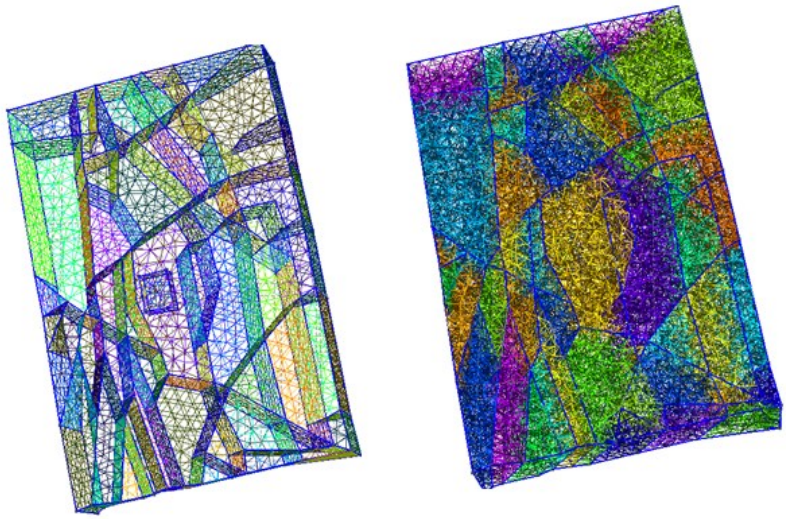
Obr. 4. Vizualizace vertikálních a horizontálních puklin části modelu. Barevně odlišené rozdílné fyzikální vlastnosti.



Obr. 5. Svislé pukliny a vodní toky v modelu – fyzikální plochy a fyzikální linie.

Modelová síť je model tvořený 2D a 3D elementy v prostoru. 2D elementy jsou trojúhelníky, 3D elementy jsou čtyřstěny. Objekty uložené v originálním GIS, které je

nutné do geometrického modelu zařadit, jsou postupně diskretizovány a je odvozován geometrický model. Modelová síť pak vyplní plochy a objemy geometrického modelu konečným počtem trojúhelníků a čtyřstěnů. Jak je na základě geometrického modelu vytvořena modelová síť, závisí na rozlišení modelu, požadavcích na výsledný počet elementů sítě. Dle požadovaného rozlišení nebo počtu elementů sítě je nastavena délka strany/hrany elementu a ta spolu s geometrickým modelem určuje výslednou hustotu, přesnost, kvalitu, počet elementů modelové sítě (Obr. 6).



Obr. 6. Modelová síť. Vlevo pouze 2D síť pro plochy modelu, vpravo kompletní 3D síť.

4 Postupy modelování rozpukaných oblastí

4.1 Předzpracování dat pro geometrické modely

Konstrukce geometrického modelu a následně modelové sítě má několik postupných kroků. Předpokládejme, že geografický informační systém pro danou lokalitu je vytvořen a naplněn potřebnými daty (vytvoření GIS popsáno v [2] a [5]). Dalším krokem je definování požadavků na geometrický model a modelovou síť. Jedná se o definování velikosti ve vertikálním a horizontálním směru, definování přesné hranice modelu a klasifikací objektů tvořících hranici, určení hustoty modelové sítě a délky strany/hrany elementu sítě pro určené části modelu. Pro geometrický model jsou vybrány objekty s vlivem na proudění a transport v dané oblasti (viz. kap. 1) a jejich výčet je v kap. 2.

Parametry budoucího modelu ovlivňují stupeň generalizace dat v GIS – výběr, zjednodušování tvarů, spojení tříd objektů a také generalizaci atributů. Dalším krokem je předzpracování dat v GIS a získání dat v takovém tvaru a s definováním těch vztahů, které jsou potřebné pro nástroje zajišťující výstavbu geometrického modelu a modelové sítě. Předzpracování dat v jednotlivých krocích je podrobně popsáno v [2]. Výsledný geometrický model je geometricky zjednodušený, ale topologické vztahy musí zůstat zachovány. Jedná se o sousednost, zachování relativní vzdálenosti objektů, které jsou ve

vztahu nebo mají společný dopad na výsledky v hydrogeologických a transportních modelech. Jsou také připraveny atributy pro naplnění geometrického modelu a modelové sítě daty. Atributy jsou exportovány z GIS databáze v textovém formátu, který lze připojit jako další vstupní soubor do hydrogeologických modelů společně se souborem modelové sítě. Veškerých požadavky na modely jsou definovány ve spolupráci s jejich uživateli – modeláři využívajícími tyto geoinformatické modely.

Výstavba geometrického modelu probíhá od bodů (0D elementy), proto při předzpracování dat jsou prvně vytvářeny bodové množiny, pomocí kterých jsou popsány všechny objekty vstupující do geometrie modelu. Každý bod nese informaci o absolutní poloze v modelu, příslušnosti k elementu geometrie a jevu, který je popisován, informaci o vztazích k okolí a také nese atributy všech jevů, ve kterých se vyskytuje (např. bod leží na hranici modelu, která je zároveň vodním tokem, leží na povrchu v určité nadmořské výšce, náleží do určitého objemu horniny, která má své fyzikální vlastnosti a zároveň leží na zlomu, který také má své vlastnosti). Na tomto základě mohou elementy vyšší dimenze získat relevantní informaci (plochy zlomů, objemy hornin, atd.). Výsledkem tohoto předzpracování dat jsou soubory obsahující informace o bodech a jejich vlastnostech a ty slouží jako základ pro vytvoření geometrického modelu.

4.2 Výstavba geometrického modelu a modelové sítě

Obecně výstavba geometrického modelu začíná z bodů, které jsou spojeny do linií, plochy jsou definovány hranicí tvořenou liniemi, objemy jsou definovány hranicí tvořenou plochami (obr. 2). Navíc jsou definovány fyzikální skupiny linií (např. linie vodních toků), ploch (plochy tektonických zlomů s danou vlastností) a objemů (jednotlivé bloky horniny stejné vlastnosti).

Pro zpracování modelové sítě je používán software GMSH [1]. Jedná se o generátor prostorových sítí. Do tohoto generátoru vstupuje geometrický model ve formátu geo. Důvod pro použití GMSH je dán požadavky na matematické modelování metodou konečných prvků v hydrologických a transportních modelech vyvíjených na TUL [4]. Plochy a objemy jsou vyplněny konečným počtem elementů (Obr. 6). Hustotu sítě lze ovlivnit nastavením parametrů v geometrickém modelu. Počet elementů sítě závisí na požadované hustotě sítě a nastavené délce strany/hrany elementu. Výsledný formát msh souboru modelové sítě obsahuje seznam všech bodů tvořících vrcholy elementů sítě a jejich souřadnic v prostoru, seznam všech elementů sítě roztříděných podle dimenze elementu a příslušnosti k fyzikálním skupinám. Podle této příslušnosti k fyzikálním skupinám lze elementům přiřadit konkrétní vlastnosti.

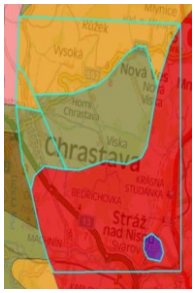
5 Automatizovaná výstavba geometrických modelů

V odstavci 4 byla popsána koncepce tvorby odvozených geoinformatických modelů, které jakožto vstupní soubory pro hydrogeologické modely mají danou strukturu a celý model musí být popsán předem daným způsobem. V rámci efektivní tvorby geometrických modelů a modelových sítí bylo vyvinuto automatizované řešení popsané dále.

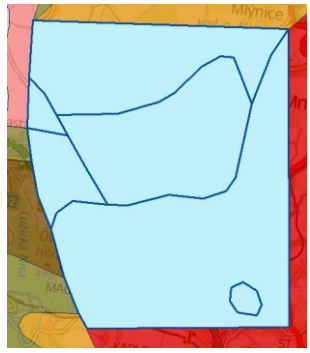
5.1 Automatizované předzpracování dat

Zásadní prvky v modelu rozpukaných oblastí mají charakter linií. Jedná se hlavně o hranice modelu, rozhraní hornin uvnitř modelu, geologické zlomy, vodní toky (Obr. 7). Tyto linie jsou vybrány v datových vrstvách originálního GIS a každá linie je vhodně označena

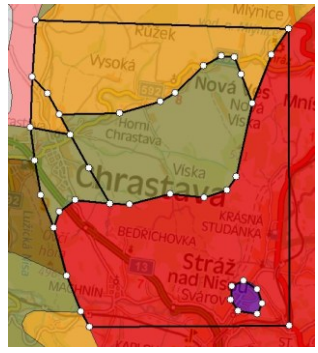
odpovídajícím atributem příslušnosti, který je zaznamenám do položky Feature v databázi. Pomocí nástroje pro rozdělení linie ve vrcholech (např. v ArcGIS funkce „Split Line in Vertices“) jsou linie důsledně rozděleny na dílčí části v místech jejich zlomů. Následně jsou identifikovány všechny základní cykly v grafu již rozdělených linií pomocí nástroje vytvoření polygonu („Feature to Polygon“) (Obr. 8). Na základě této identifikace je získána informace o počtu a rozsahu spojitých částí území (reprezentují např. bloky hornin). Jednotlivé polygony jsou následně očíslovány dle předem stanovených kritérií. Dále pomocí nástroje topologického překrytí („Intersect“) lze označit linie, které dohromady tvoří tyto základní cykly neboli ohraničují dané polygony.



Obr. 7. Linie popisují geologické charakteristiky oblasti.



Obr. 8. Polygony základních cyklů.



Obr. 9. Body vytvořené na pozicích vrcholů linií.

Z vrcholů rozdělených linií je vytvořena množina bodů („Feature Vertices to Points“) (Obr. 9). Údaje o bodech jsou zapsány do databázové tabulky (vlastnosti polohy, příslušnosti k typu linie) a přidány jsou souřadnice x, y, z. Pro každý bod (a jeho souřadnice x,y) může existovat více hodnot z, pokud se jedná o geometrický model, ve kterém je zohledněna také horizontální rozpukanost. Souřadnice z je v této fázi konstanta, teprve po vytvoření geometrického modelu je aplikován digitální model reliéfu v rámci generování modelové sítě a tím je modelován povrch reliéfu či tvary povrchu horizontálních rozhraní v různých hloubkách (př. na Obr. 4). Takto zpracovaná data o bodech a liniích uzavírajících polygony (Tab. 1 a Tab. 2) jsou vstupními soubory aplikace Convert2Geo [8] pro generování geometrického modelu.

5.2 Automatizované řešení výstavby geometrického modelu rozpukané oblasti

Za účelem automatizace výstavby geometrie je vyvíjena aplikace Convert2geo. Tato aplikace načítá databázové tabulky vytvořené v rámci předzpracování dat v GIS jako vstupní data a na výstupu zapisuje do souboru geo zdrojový zápis výsledné 2D nebo 3D geometrie. Požadovaný formát vstupních dat do aplikace Convert2geo je podrobněji popsán v následujících tabulkách (Tab. 1 a Tab. 2).

Body	{ID; X; Y; Z, F}
ID	Identifikátor bodu
X	Souřadnice X
Y	Souřadnice Y
Z	Souřadnice Z
F	Označení fyzikální skupiny složené z položek (*Typ*, *Index*)

Tab. 1. Schéma vstupních dat vrstvy bodů.

Linie	{ID; PID1; PID2; G}
ID	Identifikátor linie
PID1	Index prvního vrcholu linie
PID2	Index druhého vrcholu linie
G	Označení cyklu a fyzikální skupiny složené z (*Cyklus*, *Typ*, *Index*)

Tab. 2. Schéma vstupních dat vrstvy linií.

Aplikace vstupní data načítá do interní paměti a následně provádí výstavbu geometrie. Geometrický model je stavěn od bodu přes linie, plochy až k objemům. Zásadním krokem je identifikace duplicitních bodů. Zjednodušeně se dá říci, že všechny body vstupních dat jsou duplicitní, to je zapříčiněno jejich vytvořením v průběhu předzpracování dat z GIS. Z hlediska určování fyzikálních skupin je to výhodné, protože bod je duplicitní tolikrát, kolikrát tímto bodem prochází nějaká linie. Konkrétně pokud jde o průnik dvou puklin, víme, že jde o bod průniku a známe přesně identifikaci protínajících se puklin. V případě výstavby modelu však tyto duplicitní body musí být nahrazeny jedním zástupným bodem, aby bylo možné korektně vystavět geometrii.

Nahrazování duplicitních bodů je řešeno tím způsobem, že jsou prohledávány souřadnice XY všech bodů a pokud je nalezena shoda dvou duplicitních bodů, identifikační index druhého bodu je přepsán identifikačním indexem prvního bodu. Za tímto účelem je vytvořena Tabulka indexů bodů, kde v prvním sloupci je uveden původní index bodu a ve druhém sloupci je uveden jeho nový index [6]. Tento způsob nám zaručuje pohodlný přístup od původních duplicitních dat k novým neduplicitním, na základě kterých může být vystavěna geometrie modelu.

Podobný problém s duplicitními daty je i v případě linií, zde jde o důsledek toho, že potřebujeme vědět, které linie tvoří cyklus. A z teorie grafů je zřejmé, že každou linií může procházet více cyklů i v případě že se jedná o základní cykly [7]. Proto jsou linie duplikovány pro každý cyklus, který jimi prochází. I v tomto případě je vytvořena Tabulka indexů linií. Princip je podobný, index druhé linie je nahrazen indexem první, pokud mají některé dvě linie stejné oba dva zástupné body vrcholů. Postup je tedy takový, že vrcholy linie (jako body načtené ze vstupních dat) se nejdříve převedou na zástupné body a až po té se porovnávají.

Tyto zástupné body respektive linie mají svůj význam i v dalších částech výstavby geometrie, kde jsou využívány rovněž pro stavbu ploch a objemů. Výstavba ploch je založena na hledání posloupnosti linií (hran) za sebou jdoucích tak, aby byly správně orientovány vůči ploše, kterou ohraničují (návaznost), což se porovnává opět podle zástupných bodů. V případě horizontálních ploch se při určování linií, které plochu ohraničují, využívá údaj z atributu G konkrétně položka *Cyklus*, jak je uvedeno v tab. 2. Stejný princip se uplatňuje i v případě výstavby objemů, kde se hledají vertikální plochy ohraničující daný objem.

Na konci transformace jsou v souboru geometrického modelu vytvořeny fyzikální skupiny, neboli skupiny elementů u kterých si definujeme stejné fyzikální vlastnosti např. propustnost. K určení, do jaké fyzikální skupiny daná linie, plocha nebo objem patří, slouží položky *Typ* a *Index* atributu G nebo F pro linie respektive body, uvedené v předchozích

tabulkách. Toto označení je složené z předem definovaných hodnot oddělených podtržítkem, např. V případě linií, může být v atributu G uložen řetězec „1_2_1“, jedná se o tři hodnoty pro každou z položek (*Cyklus*, *Typ*, *Index*). V případě bodů v atributu F se jedná pouze o dvě položky (*Typ*, *Index*) oddělené podtržítkem, např. „2_1“. Každá hodnota těchto položek má svůj určitý význam a každá položka může být definována určitým rozsahem nebo omezením. Položka *Cyklus* je omezená počtem nalezených základních cyklů v grafu, a proto může nabývat pouze hodnot od 1 do N, kde N představuje počet nalezených cyklů. Položka *Typ* má pevně definovaný rozsah a význam jednotlivých hodnot, jichž může nabývat. Jednotlivé hodnoty a jejich významy jsou zobrazeny v tabulce (Tab. 3), ze které je patrné, že rozsah položky je od 1 do 4.

1	vnější hranice modelu
2	hranice uvnitř modelu
3	geologický zlom
4	vodní toky

Tab. 3. Seznam přípustných hodnot položky *Typ*.

Třetí a poslední položkou je *Index*. Její význam je v počítání a označení instancí každé ze čtyř položek *Typ*. To znamená, že pokud máme v modelu např. pět výskytů různých geologických zlomů, bude položka *Index* nabývat hodnot od 1 do 5 pro tento daný typ. Položka *Index* je tedy přímo podřízená položce *Typ*. To neplatí u položek *Cyklus* a *Typ,* kde mezi nimi není žádný vztah.

Fyzikální skupiny se prostřednictvím aplikace Convert2geo vytváří zcela automatizovaně na základě těchto předem specifikovaných vlastností, které jsou definovány již na počátku v originálním geoinformačním systému a nesou si je linie vytvářené v rámci automatizovaného předzpracování dat. Návaznost na originální GIS a původní data je tak zachována.

6 Závěr a zhodnocení

Automatizovaná výstavba geometrických modelů a modelových sítí rozpukaných oblastí je velmi efektivní z hlediska časového, kdy během krátké doby může být vytvořeno mnoho variant geometrických modelů a sítí s různou konfigurací puklinových zón a zpracovaných v různém detailu. Díky těmto variantám lze použít tyto variantní vstupy do hydrogeologických a transportních modelů a v reálném čase testovat vliv výskytu jednotlivých puklin a rozhraní na výsledek rychlosti proudění a transportu v horninovém prostředí.

Literatura

1. Geuzaine, C., Remacle, J.-F. *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.* International Journal for Numerical Methods in Engineering, Volume 79, Issue 11, pages 1309-1331, 2009
2. Malá, B., Pacina, J., Capeková, Z.: Účelově odvozované modely v procesu předzpracování dat pro tvorbu geometrie modelových sítí. In: *SIMONA 2009*. TUL, Liberec (2009), 87-93.
3. Malá, B., Pacina, J.: Creation of model meshes for hydrogeologic modeling. In. *GIS Ostrava 2011*. VŠB-TU Ostrava (2011), 13 pp.
4. Maryška, J., Královcová, J., Hokr, M., Šembera, J.: Modelování transportních procesů v horninovém prostředí. Technická Univerzita v Liberci (2010). 303 pp.
5. Maryška, J., Malá, B.: Výstavba modelové sítě a její naplnění hodnotami z GIS SURAO a stanovení počátečních podmínek pro různé varianty migrace. *Dílčí závěrečná zpráva projektu Výzkum procesů pole vzdálených interakcí HÚ vyhořelého jaderného paliva a vysoce aktivních odpadů.* TUL Liberec (2008), 52 s.
6. Sedgewick, R: *Algoritmy v C, Části 1-4*. SoftPress, 2003.
7. Sedgewick, R.: *Algorithms in Java, Part 5 (Graph Algorithms).* Princeton University, 2004.
8. Tomcik, D.: Convert2geo [online]. 2010 . *Convert2geo*. Dostupné z WWW: <https://sites.google.com/site/convert2geo/home>.
9. Tomčík, D., Malá, B.: Automation in construction of 3D models and meshes. In: *DATAKON 2010*, Petr Šaloun (ed.), Mikulov, 16-19. 10. 2010, University of Ostrava (2010) , pp. 153 to 157.

Poděkování
Tato práce byla realizována za podpory Ministerstva průmyslu a obchodu České republiky v rámci projektu „MPO“ FR-TI1/362.
Tato práce byla realizována v rámci státní dotace České republiky v programu specifických výzkumů 7822/115 podporovaném Ministerstvem školství.

Annotation:

Geoinformatics modelling of the cracked rocks area

The main point of this paper lies in the methods of the creation of the 2D/3D geometry and mesh models. Especially, the paper is focussed on the modelling of the area, where the feature is characterised by both horizontal and vertical geology features such as rock cracks or boundaries. The models are usually created with a different resolution and detail, hence a general methodology has been created, that contains a data preprocessing in the GIS, a creation of the geometric model by Convert2geo and a generation of the mesh model by GMSH. Important aspect of this process is its ability to define some physical characteristics, already before the geometric model is created.

Případové studie

Nejnovější vlastnosti databázové technologie Informix

Jan MUSIL

IBM Česká republika
V Parku 2294/4, 148 00 Praha 4
jan_musil@cz.ibm.com

Abstrakt. Rodina databázových serverů IBM Informix se neprezentuje pouze jako rodina serverů pro transakční zpracování, ale má i významnou podporu pro analytické zpracování. Společně s verzí Informix 11.7 přichází s unikátní technologií Informix Warehouse Accelerator, která umožňuje zrychlit analytické operace sto až tisíci násobně. Tato technologie patří do skupiny tzv. databázových technologií třetí generace. Další unikátní technologií je podpora zpracování dat pravidelných i nepravidelných časových řad. K dispozici je zde zdarma nativní podpora správy a ukládání těchto dat co nejefektivněji jak z pohledu výkonnosti, tak z pohledu potřebných diskových prostorů. Následující příspěvek podává stručný přehled obou technologií.

Klíčová slova: Informix, relační databáze, datové sklady, časové řady.

1 Úvod

Databázové technologie Informix patří již tradičně mezi stálice databázového trhu přinášející s každou novou verzí významné nové vlastnosti a vylepšení. Pro tuto studii jsme mezi mnoha vybrali dvě oblasti, které považujeme za unikátní. První oblastí je zcela nový koncept správy dat datových skladů a druhou oblastí je zpracování dat pořizovaných v pravidelných i nepravidelných časových řadách. Výběrem těchto oblastí si dovolujeme demonstrovat skutečnost, že ne vždy je pro některé typy aplikací (databází) vhodné striktně relační ukládání a zpracování, a že nerelační ukládání některého typu dat může poskytovat podstatně vyšší výkonnost a úsporu v diskových kapacitách, ovšem při plné transparentnosti vůči existujícím aplikacím a při zachování SQL přístupu takto (nativně) ukládaných dat.

2 Datové sklady

2.1 Co je problémem ?

Pokud pro provoz a správu dat datového skladu používáme relační databázové stroje, můžeme pro optimalizaci přístupu k zpracovávaným datům použít například partitioning tabulek, partitioning databází, můžeme maximalizovat paralelní zpracování jak na databázové úrovni, tak s použitím masivně paralelní HW architektury, můžeme přidělit maximum zdrojů databázového serveru a podobně. I přes veškerou snahu však stále narážíme na omezení, která vedou k tomu, že analytické dotazy do datových skladů se zpracovávají

dlouho, prakticky je nelze spustit současně s transakčním zpracováním na jednom serveru a v neposlední řadě je třeba stále tyto analytické dotazy optimalizovat tak, aby plán přístupu k datům byl, pokud možno, co nejefektivnější.

Obecně lze říci, že databázové technologie tzv. druhé generace, které se používají v drtivé většině, a které ukládají data po jednotlivých záznamech a pro přístup k nim je potřeba velkého množství vstupně výstupních operací na disku, nejsou příliš vhodné pro analytické dotazy. Dle blogu Freda Ho ze společnosti IBM [1], který se odvolává na studii IDC [2], získávají stále více na aktuálnosti databáze třetí generace, kde si můžeme dovolit zapomenout na koncept diskového partitioningu, koncept různých indexovacích strategií a koncept buffer managementu, protože tyto nové systémy nám nabídnou zpracování analytických dotazů v paměťových databázích s vícejádrovými procesory a klastrovanými servery s vysoce komprimovanými daty uloženými na disku nikoliv po záznamech, ale po jednotlivých sloupcích.

Nástup databázových technologií třetí generace lze podle stejné studie IDC [2] očekávat do pěti let. IDC odhaduje následující charakteristiky databázových technologií třetí generace.

1. Většina dat v datových skladech bude uložena po sloupečcích
2. Většina OLTP databází bude bude udržována částečně nebo zcela v paměti
3. Většina databázových serverů, které zpracovávají rozsáhlé databáze budou škálovatelné horizontálně prostřednictvím klastrování
4. Většina problémů se sběren dat a jejich reportování bude řešena databázemi, které nemají již vůbec formalizované datové schéma

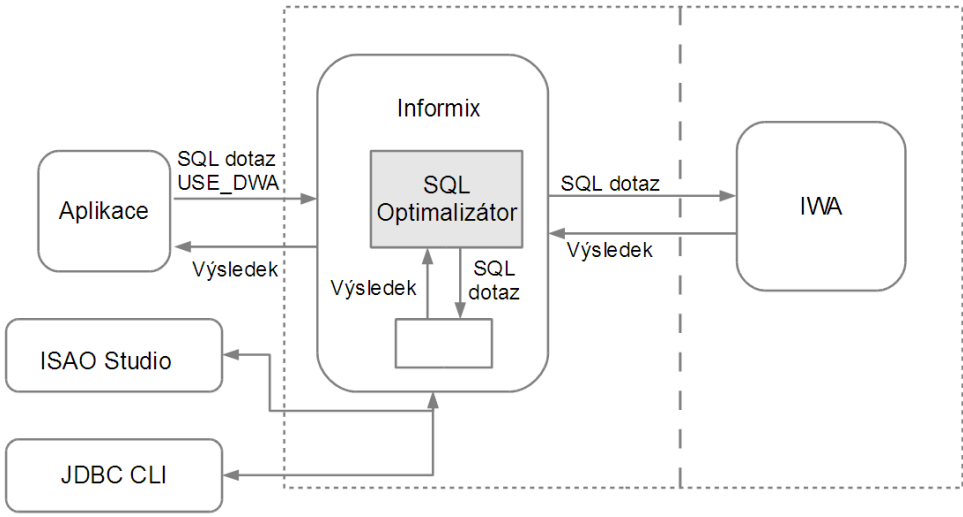
2.2 Databázové technologie třetí generace prakticky

Praktickou realizaci konceptu databázových technologií třetí generace nabízí IBM pro své zákazníky, kteří provozují datové sklady na databázové platformě Informix. Rozhodnutí poskytnout stávajícím a budoucím IBM Informix zákazníkům technologii podporující rychlé zpracování analytických dotazů vyplývá také ze skutečnosti, že 40% současných Informix zákazníků provozuje aplikace datových skladů. Nová technologie databázové platformy třetí generace se jmenuje Informix Warehouse Accelerator (IWA) a jak je patrné již z názvu technologie, hlavním cílem je zrychlit stávající analytické dotazy, zpracovávané aktuálně na konvenční (druhogenerační) databázové platformě Informix.

Vedle již zmíněné mimořádné výkonnosti, dle praktických zkušeností zákazníků se dotazy zrychlily sto až tisícinásobně (!), jsou dalšími výhodami plná transparentnost řešení ve vztahu k původní aplikaci, což znamená, že analytický dotaz se pošle původnímu databázovému serveru jako před tím, optimalizátor ovšem rozhodne, zda se dotaz bude zpracovávat lokálně (konvenčně) nebo se pošle akcelerátoru (viz. Obrázek č. 1). Pokud se dotaz zpracovává v akcelerátoru, pak požadovaná data jsou poslána zpět původnímu databázovému stroji a ten je zprostředkuje aplikaci. Z tohoto pohledu nejsou třeba naprosto žádné změny na straně aplikace, ani žádné rekonfigurace stávajícího databázového serveru. Další důležitou výhodou je oddělení transakčního zpracování od analytického mezi dva různé počítače, případně mezi dvě různé instance databázového serveru, pokud vše běží na jednom počítači – o této alternativě lze ale uvažovat spíš v případě testovacího nebo demonstračního režimu. V neposlední řadě je třeba zmínit, že nová technologie nevyžaduje

žádné speciální ladění, případně optimalizaci dotazů, vytváření indexů nebo distribučních statistik optimalizátoru.

Podporovanými datovými schámaty jsou hvězdice a sněhové vločky (tedy vzájemně svázané tabulky faktů a tabulky dimenzí). Akcelerovat, tedy přenést data a jejich strukturu z tradiční relační databáze do IWA lze buď pro celé schéma datového skladu nebo jeho podmnožinu. Akcelerované schéma s daty na úrovni IWA se nazývá *data mart*.



Obrázek č. 1: Architektura řešení Informix Warehouse Accelerator

2.3 Konfigurace a architektura řešení

Konfigurace Informix Warehouse Acceleratoru (dále jen IWA) je velmi jednoduchá. Po nainstalování software na provozní platformu, kterou je 64bit Intel s operačním systémem Linux RHEL 5 nebo SLES 5, se provede několik jednoduchých konfiguračních kroků. Konfigurace spočívá v určení diskové oblasti, kde budou uložena akcelerovaná data datového skladu, dále pak se určí počet uzlů tohoto databázového serveru třetí generace, paměť přidělená každému uzlu a na závěr způsob komunikace s původním databázovým serverem (vzdálená nebo lokální).

Z pohledu procesů, v prostředí IWA běží pouze dva typy procesů (uzlů). Prvním typem jsou koordinační uzly, zodpovědné za komunikaci s relačním databázovým serverem, zpracování požadavků a koordinaci druhého typu uzlů, což jsou tzv. pracovní uzly. Koordinační uzly jsou také zodpovědné za administraci systému. Je pevně stanoveno, že jeden koordinační uzel může spravovat maximálne pět pracovních uzlů. Například tedy při stanovení celkového počtu 7 uzlů, budou automaticky spuštěny 2 koordinační uzly a 5 pracovních uzlů. Míra paralelizmu zpracování analytických dotazů je pak dána počtem pracovních uzlů. Pracovní uzly komunikují pouze s koordinačními uzly.

Jak koordinačním uzlům, tak pracovním uzlům je přidělena sdílená paměť. Takto přidělená sdílená paměť by měla být tak velká, aby zde bylo možné načíst kompletní data potřebná pro analytický dotaz. Jak si vysvětlíme dále, tato data jsou komprimována vysoce účinným komprimačním algoritem. Dle doporučení, pracovním uzlům, které spravují veškerá data, potřebná pro analytický dotaz, je vhodné přidělit dvě třetiny celkové paměti počítače. Koordinačním procesům, které nevyžadují příliš paměti, lze přidělit 5 až 10% dostupné paměti.

Při akceleraci dat potřebných pro analytický dotaz, jsou transformovaná data (způsob jejich transformace bude popsán později) rozdělena mezi jednotlivé pracovní uzly následujícím způsobem. Tabulka faktů, která je pochopitelně nejrozsáhlejší, je rovnoměrně rozdělena mezi všechny pracovní uzly, čímž je zajištěn paralelizmus zpracování dotazů na „první úrovni“. O další úrovni paralelismu budeme hovořit později. Vzhledem k tomu, že tabulky dimenzí neobsahují velký objem dat, každý uzel má k dispozici úplnou jejich kopii.

Administrace a inicializace instance IWA se provádí prostřednictvím jednoduchého řádkového příkazu. Definici data martů a akceleraci dat lze provádět buď z prostředí grafického Smart Analytics Studia (ISAO), vytvořeného na platformě Eclipse nebo z příkazové řádky. Aby mohl být analytický dotaz akcelerován, musí být všechny tabulky, se kterými pracuje, součástí data martu, jehož data jsou transformovaná do IWA. Definici data martu lze provádět manuálně z grafického nebo řádkového prostředí, ovšem po analýze toho, s jakými tabulkami analytický dotaz pracuje. Aby nebylo nutné provádět tuto analýzu manuálně, lze vytvořit strukturu data martu automaticky po automatické analýze SQL analytických dotazů.

2.4 Transformace dat

Po vytvoření data martu, tedy struktury schématu, který je potřeba pro akcelerované analytické SQL dotazy, je třeba transformovat data z původní databáze do IWA. Data jsou při transformaci jednak převedena z řádkového ukládání do položkového, jednak komprimována pomocí Huffmanovo algoritmu do binárního tvaru.

Jelikož se při analytických dotazech provádí agregace nad tabulkovými položkami, je výhodnější právě použít položkové ukládání dat, protože pro agregace potřebujeme co nejrychleji zpracovat co nejvíce hodnot jednotlivých položek. Při řádkovém ukládání dat je třeba načíst kromě potřebné hodnoty příslušné položky i ostatní nepotřebné hodnoty zbylého záznamu, což je pro analytické dotazy nevýhodné. Ukládání po záznamech je naopak výhodnější pro transakční zpracování, kdy potřebujeme zpracovat informaci uloženou v celém nebo v části jednotlivého záznamů.

U binární komprese pomocí Huffmanova algoritmu je několik položek tabulky spojeno do buňky a tyto buňky jsou na základě frekvenční analýzy zakódovány binárními hodnotami. Čím je frekvence výskytu hodnot v buňce vyšší, tím kratší je binární hodnota, kterou jsou data buňky zakódovaná. Kódování do binárního tvaru umožňuje vysokou míru komprese dat, která je nutné pro uložení dat data martu do paměti počítače, kde běží IWA. Druhou výhodou binární komprese je skutečnost, že agregace a vyhodnocení podmínek analytického dotazu se provádí pomocí „Single Instruction Multiple Data“ (SIMD) paralelismu přímo v registrech Intelové platformy, na které je IWA instalován. Intel Xeon plat-

forma má 128bitové registry. Pokud má buňka kratší délku (což zaručuje extrémní komprese Huffmanovým algoritmem), pak lze paralelně jednou instrukcí zpracovat více záznamů najednou. Tím zajistíme paralelismus „druhé úrovně“.

2.5 Výhody řešení

První zásadní výhodou celého řešení je jednoduchá integrace do stávající databázové infrastruktury. Na straně původního databázového serveru není třeba provádět žádné změny v konfiguraci, veškerá konfigurace se provádí při implementaci IWA řešení. Další výhodou je, že není třeba cokoliv měnit ve stávající aplikaci. Aplikace stále komunikuje s původním databázovým serverem, jehož optimalizátor rozhodne, zda se bude dotaz akcelerovat, či nikoliv. Významnou implementační výhodou je skutečnost, že integrace IWA a jeho spuštění nijak neohrozí dostupnost dat. Pokud by z nějakého důvodu nebylo možné dotaz akcelerovat, provede se lokálně, jako dříve. Zákazníci se tedy nemusí obávat, že nasazováním řešení může dojít z nějakého důvodu k výpadku v dostupnosti dat.

Zásadní výhodou je pak několikanásobné zrychlení provádění analytických dotazů. Máme k dispozici statistiky porovnání doby provedení analytického zpracování bez IWA a s IWA. Zpracování trvající od minut do desítek minut a i déle bez akcelerátoru, trvala několik sekund s akcelerátorem.

3 Zpracování dat v časových řadách

3.1 Přehled technologie

Problém ukládání dat pořizovaných v pravidelných nebo nepravidelných časových řadách je obvykle v tom, že jejich relační ukládání vyžaduje velmi „štíhlou“ a „vysokou“ relační tabulku.

„Štíhlostí“ se zde myslí velmi malý počet položek, definovaných pro tento typ dat. Položkami obvykle bývají časový okamžik (časové razítko) pořízení datového záznamu, subjekt, se kterým je svázaná získaná hodnota v daném časovém okamžiku a získané hodnoty. Subjekty pak mohou být například měřidla nějakých hodnot (např. měřidlo spotřeby elektrického proudu domácnosti nebo akcie, obchodované na burze) a hodnotami pak mohou být například naměřené hodnoty spotřeby elektrického proudu domácnosti v daném časovém okamžiku nebo hodnoty akcií, a to např. aktuální hodnota v daném časovém okamžiku, maximální, minimální a průměrná hodnota akcie za nějaký časový interval, např. jeden den, kdy se obchoduje. „Výškou“ se pak myslí velmi vysoký počet záznamů v dané tabulce.

Aby byla doba zpracování nad takovými daty alespoň částečně uspokojující, musí být nad takovou tabulkou vytvořen index, a to buď pouze nad některými záznamy nebo dokonce indexy nade všemi záznamy takové tabulky. Vezměme si příklad měřidla elektrického proudu, který posílá elektrárenské společnosti každých 15 minut aktuální hodnoty spotřeby proudu každé z miliónů domácností, připojených na takový odečet. Pokud se ptáme na spotřebu proudu všech domácností za určité časové období, pak musíme indexovat položku s

jednotlivými časovými údaji. Pokud se ptáme na spotřebu elektrického proudu konkrétní domácnosti za určité období, pak musíme navíc indexovat položku s identifikátory měřidel (tedy domácností). Pokud potřebujeme vyhledat domácnost z určitého regionu s maximální spotřebou v průběhu určitého časového období, pak musíme indexovat všechny tři položky. Pokud vezmeme v úvahu, že vedle diskového prostoru pro vlastní data potřebujeme také diskové prostory pro indexy, pak požadavek na velikost diskových kapacit může být poměrně vysoký.

Problém výkonnosti a optimalizace diskového prostoru, potřebného pro ukládání dat časových řad, řeší technologie Informix TimeSeries.

Tato technologie dovoluje efektivně interpretovat a ukládat data pořizovaná v časových řadách. Hlavními znaky této technologie jsou:

1. optimální ukládání časových řad v nativním formátu, které vyžaduje podstatně méně diskových prostorů než v případě „klasického“ relačního ukládání
2. výběry, agregace a statistické výpočty jsou podstatně rychlejší než v případě dotazů při klasickém relačním ukládání dat
3. dostupnost mnoha funkcí na SQL úrovni pro interpretaci a manipulaci dat v časových řadách
4. plná podpora objektového přístupu k datovému typu TimeSeries

3.2 Porovnání tradičního relačního a nativního ukládání dat časových řad

Výhody technologie Informix TimeSeries spočívají jednak v úspoře diskových prostorů potřebných pro ukládání dat v časových řadách, jednak v podstatně vyšší výkonnosti dotazů.

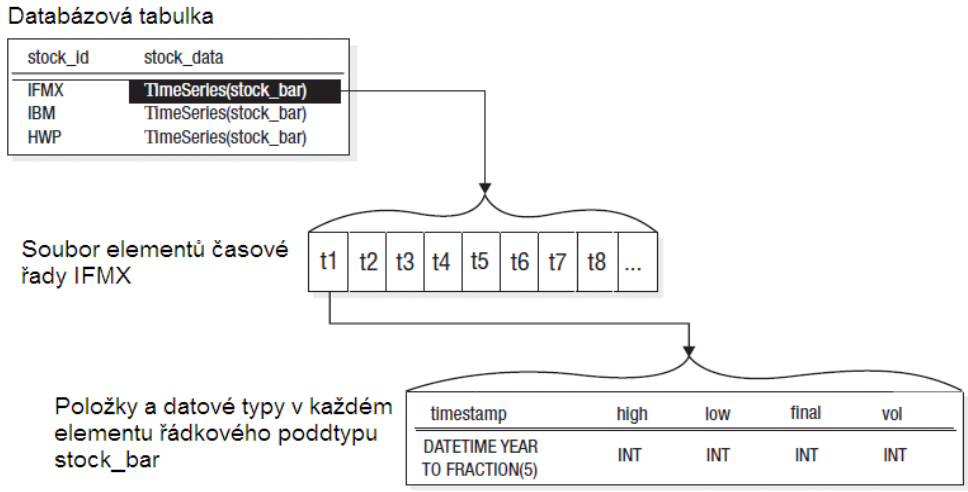
Srovnáme-li prostor potřebný pro uložení dat časových řad, technologie Informix TimeSeries potřebuje ve srovnání s tradičním uložením v relačních databázích v průměru asi ¼ diskového prostoru. Konkrétně data uložená v tradiční relační databázi potřebovala 1.3 TB, v případě technologie Informix TimeSeries to bylo 350 GB.

Z pohledu výkonnosti, byly provedeny následující testy. Load dat časové řady (1 milion záznamů) trval v případě tradiční relační databáze 7 hodin, v případě Informix TimeSeries technologie 18 minut. Různé příklady výběrů dat (agregace, statistické výpočty a podobně) trvaly v případě klasické relační databáze od 2 do 7 hodin, v případě technologie Informix TimeSeries to bylo od 25 vteřin do 6 minut.

3.3 Architektura řešení

Z pohledu architektury technologie Informix TimerSeries jsou všechna data určité časové řady uložena v jedné položce jednoho záznamu tabulky, přičemž v jedné tabulce může být uloženo více různých časových řad. Fyzicky jsou data uložena v optimalizovaném formátu v předem definovaném kontejneru (případně kontejnerech pro zajištění paralelního zpracování). Aby bylo možné ukládat skutečně pouze platné hodnoty časové řady, definuje se tzv. kalendář a kalendářové vzory, které určují, ve kterých časových okamžicích se pořizují platná data a kdy nikoliv. Kalendářovým vzorem může být například pracovní týden, kdy

lze určit, že všechna data jsou pořizována od pondělí do pátku a nikoliv o víkendu. Pravidelné časové řady pak nemusí ukládat NULL hodnoty pro období, kdy se data nepořizují, čímž se optimalizuje prostor potřebný pro uložení dat časové řady.



Obrázek č. 2: Architektura nativního ukládání dat časové řady

Jak můžeme vidět na obrázku č. 2, nově vytvořený nativní datový typ TimeSeries deklaruje data typu časové řady, přičemž tzv. subtyp použitý v deklaraci definuje strukturu ukládaných dat časové řady. Tento subtyp je definovaný jako složený datový typ „row“. Každý záznam časové řady musí mít povinně jako první položku časové razítko ve formátu DATETIME YEAR TO FRACTION(5). Tak lze ukládat data s přesností časového razítka na stotisícinu vteřiny. Hodnota časového razítka každého záznamu časové řady musí být unikátní. Struktura záznamu časové řady může být jinak zcela libovolná, například struktura pro popis hodnoty nějaké akcie může být následující. Časové razítko – DATETIME YEAR TO FRACTION(5), objem obchodů v časovém intervalu mezi dvěma záznamy – INTEGER, počáteční hodnota akcie v časovém intervalu – FLOAT, konečná hodnota akcie v časovém intervalu – FLOAT, maximální hodnota akcie během časového intervalu – FLOAT a další dle potřeby.

Technologie Informix TimeSeries dovoluje ukládat a zpracovávat data pořizovaná v pravidelných časových intervalech (pravidelné časové řady) a v nepravidelných časových řadách (nepravidelné časové řady). Hodnoty časových razítek u pravidelných časových řad se ukládají pouze ve formě offsetu k hodnotě časového razítka prvního záznamu. Tím dochází k další efektivitě využití prostoru potřebného k uložení časové řady. U nepravidelné řady je hodnota časového razítka uložena u každého záznamu.

Na úrovni SQL API jsou vedle funkcí pro správu kalendářů a kalendářových paternů k dispozici také funkce pro práci s vlastními daty časových řad. Tyto funkce dovolují získávat informace o definici časových řad, konvertovat časové řady mezi různými kalendářemi, vybírat a modifikovat jednotlivé záznamy nebo množiny těchto záznamů, vytvářet a plnit časové řady, provádět statistické, aritmetické a agregační operace nad časovými řadami a řada dalších. Kromě přístupu k časovým řadám prostřednictvím SQL API funkcí, lze vytvořit virtuální tabulku, která dovoluje záznam časové řady mapovat do relační struktury a tím pracovat s časovou řadou, jakoby byla uložena v tradičním relačním formátu. Pro psaní aplikací nad časovými řadami jsou k dispozici vedle SQL rozhraní, také JDBC ovladače a aplikační rozhraní pro psaní aplikací v jazyce C.

Informix TimeSeries rozšíření je bezplatnou součástí objektově relační databázové technologie Informix.

4 Závěr

Na závěr příspěvku je třeba konstatovat, že podpora zpracování analytických dotazů a podpora zpracování dat v časových řadách nejsou ani zdaleka jediné významné vlastnosti databázové technologie IBM Informix.

Jako další můžeme například jmenovat řešení vysoké dostupnosti, které má již dlouholetou tradici a svojí spolehlivostí a jednoduchostí patří mezi nejčastěji používané komponenty. Vedle řešení vysoké dostupnosti nabízejí databázové servery IBM Informix také replikaci dat na úrovni jednotlivých tabulek, včetně nadstavby Informix Flexible Grid. Technologie Informix Flexible Grid s podporou replikace dat nemá mezi ostatními dodavateli databázových technologií obdoby. Výhodou je plná integrace technologie do jádra databázového serveru, možnost konfigurace libovolné topologie bez ohledu na geografickou strukturu, automatická detekce a řešení konfliktních situací. Technologii Informix Flexoble Grid lze provozovat společně s klastry vysoké dostupnosti. Pro řízení přístupu pak slouží Connection Manager, který vybere ten nejvhodnější server v celé struktuře. Failover Arbitrator pak zajistí automatické přepnutí nejvhodnějšího záložního serveru do primárního módu, pokud dojde k pádu původního primárního serveru.

Tradičními výhodami databázové technologie Informix jsou jednoduchost implementace všech komponent, jednoduchost administrace, vysoká výkonnost, soulad s moderními bezpečnostními standardy, mimořádná stabilita a spolehlivost. Všechny tyto výhody pak vedou k oblibě této platformy u zákazníků.

Literatura

1. Fred Ho, *Fred Ho's Blog, The 3rd Generation of Database Technology*, IBM developerWorks (ibm.com/developerworks), 20.2.2010
2. Carl W. Olofson, *The Third Generation of Database Technology: Vendors and Products That Are Shaking Up the Market*, IDC, Feb 2010
3. *IBM Informix Warehouse Accelerator Administration Guide*, Informix Product Library Version 11.70, IBM Corporation 2010, 2011
4. *IBM Informix TimeSeries Data User's Guide*, Informix Product Library Version 11.70, IBM Corporation 2006, 2011

Annotation:

The document gives the overview of the most interesting technologies in version IBM Informix 11.7. Informix Warehouse Accelerator (IWA) significantly improves the response time of the analytical queries. IWA is the implementation of the 3rd generation database technology defined by IDC. Analytical queries are speed up 100 – 1000 times. The architecture of the solution allows to store the data sequentially by column (columnar data stores) instead of row oriented data store. The technology uses a high efficient compression which transform data into binary form which allows SIMD (Single-Instruction, Multiple-Data) parallelism of the operations run directly in registers. The main advantages of the technology are the application transparency, simple configuration and administration, no tuning, no indexes, no statistics for optimizer and also using of the inexpensive HW.

The other technology introduced in this paper is very efficient processing of the time series data. The technology enhanced the relational database technology by the new native data type which allows to store time series data much more better then in relational form and gives the significant query performance and the reduction of the storage. The time series data doesn't require no indexes. The technology is suitable for storing of the data produced by the meters (e.g. electricity consumption) or other devices which produce regular or irregular time series data and which require to be store very fast and efficiently and require the effective query processing.

Modelování procesů a služeb, aneb ETL procesy v praxi

Martin JANČEK

Komix s.r.o
Holubova 1, 155 00 Praha
jancek@komix.cz

Abstrakt. Vycházíme-li z obecného předpokladu, že proces představuje zobecnění pohledu na činnosti, které vytvářejí zákazníkovi přidanou hodnotu, určitě budeme mít pravdu, že na trhu existuje spousta nástrojů, které modelování těchto činností podporují. Jistě mi dáte za pravdu, že k tomu, aby bylo nutno vytvářet nástroje nové, by musel existovat pádný důvod. V následujícím textu bych rád popsal naše zkušenosti ze situací, ve kterých jsme se setkali s potřebou nejenom procesy modelovat, ale také dát zákazníkovi k dispozici nástroj, kterým si tyto procesy bude schopen nejenom popsat, ale také udržovat a měnit.

Klíčová slova: Extract, Transform, Load, SQL, Windows Workflow Foundation

1 Úvod

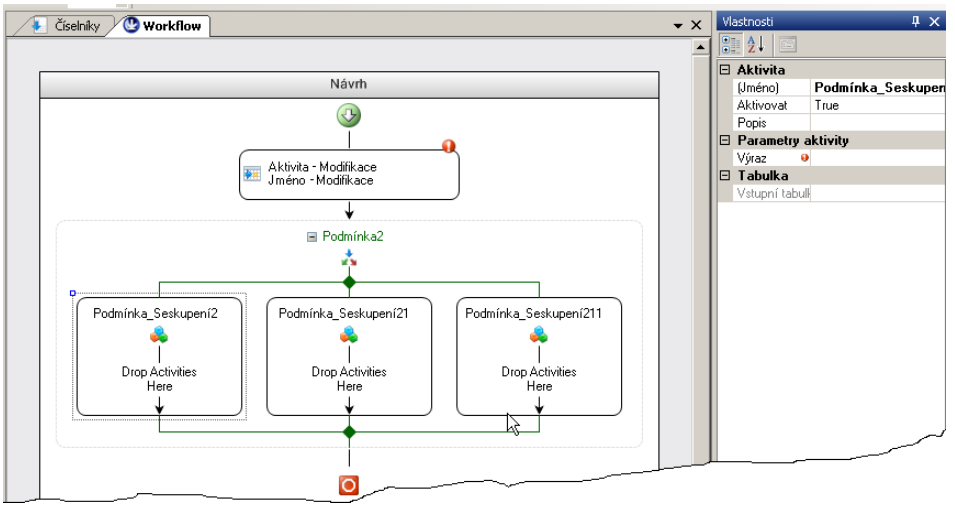
V následujícím textu bych rád popsal naše zkušenosti, ve kterých jsme se setkali s potřebou nejenom procesy modelovat, ale dát zákazníkovi k dispozici nástroj, kterým si tyto procesy bude schopen nejenom zachytit, ale také udržovat a měnit. Standardním východiskem projektu bylo zachytit tok zpracování dat, se kterými se v systému pracuje, a dosáhnout očekávaných výsledků na výstupech. Data pocházejí z různých zdrojů a jsou v různých formátech, kde předpokladem bylo navrhnout systém tak, aby se data po transformacích bez delšího prodlení dostala k odběrateli. Průběh požadovaného procesu tedy zodpovídá standardnímu ETL (Extract, Transform, Load) procesu, který bylo potřeba podpořit modelovacím nástrojem. Podíváme-li se z pohledu technologického konceptu Microsoftu, na kterém bylo nutno požadované řešení vytvořit, tak po krátkém váhání jsme zvolili technologii WF (Windows Workflow Foundation).

2 Hledání

Volbou vhodné technologie jak známo vyřešíte jenom jednu část problému. Zbývá již „jenom“ dané řešení v této technologii připravit. Pokud má tato kapitola název „Hledání“, myslím, že výstižnější název bych asi nenašel. Jedna stránka mince je mít technologii, která umí navržený proces provést, ale druhá strana mince, nemít kromě integrované vývojového prostředí nástroj, ve kterém by uživatel mohl proces vytvářet. S takovou “drobností“ jsme však již dle stanoveného konceptu tak trochu počítali a potřeba integrovat návrháře procesů do nástroje pro zákazníka přerostla v nutnost takového návrháře vytvořit.

3 Koncept

Vytvořením grafického nástroje (návrháře), vycházejícího z možností komponentního modelu platformy .NET a jeho integrací do systému, již bylo možné definovat komplexní příkazy, které příslušnou operaci vykonají. Spuštění těchto operací lze provést buď ručně přímo z aplikace, nebo automaticky dle předem definovaného plánu. Spuštění, a to buď automatické nebo ruční,umožňuje princip zachycení procesu v jednotném formátu XAML. Pro vytvoření samotného návrháře jsme využili technologií společnosti Microsoft jako jsou Windows Forms a Windows Workflow Foundation. Celý grafický návrh datového toku je vytvořen jako samostatná komponenta, kterou Windows Forms hostují.
Tento formát je vhodným nejenom pro opětovnou grafickou úpravu, ale také poskytuje možnosti vyhledávat v tomto formátu dle stanovených kritérií. Samotné modelování příkazů se provádí skládáním grafických prvků do základních programovacích konstrukcí jako jsou sekvence, podmínka a cyklus. Chování samotného příkazu se upravuje nastavením hodnot parametrů (vlastností) prvku použitím literálů, nebo výrazů a funkcí. Na následujícím obrázku je vidět náhled na základy modelování a jeho možnosti.

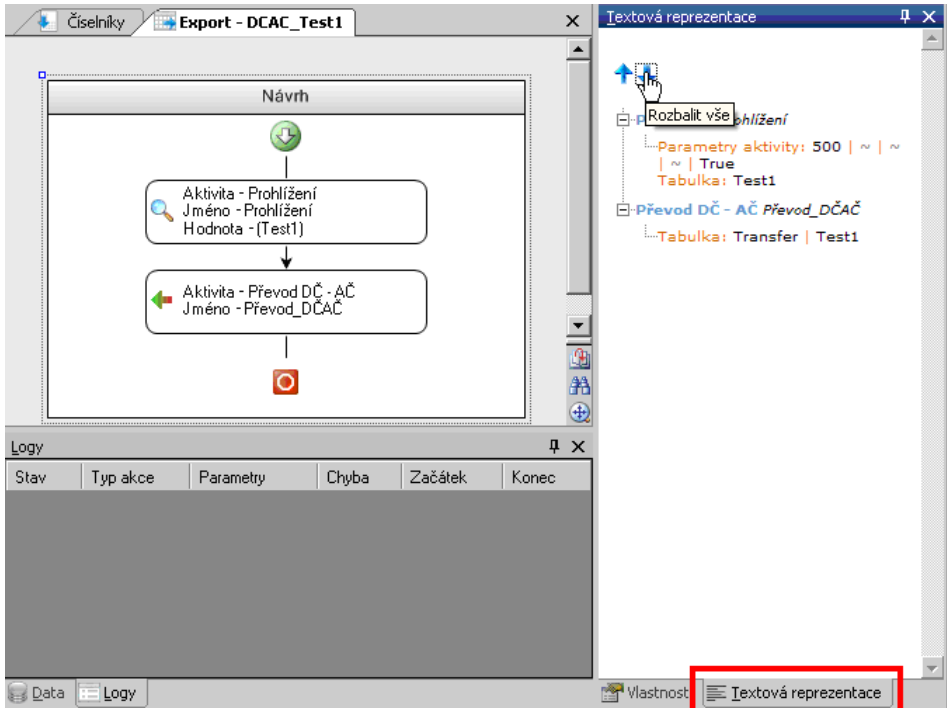


Obrázek 1 - Základní možnosti modelování

Základními stavebními prvky grafického modelování v našem případě jsou:

- **Aktivita** – základní předdefinovaná dílčí operace (např. "Přidat sloupec"), z níž se skládají komplexnější příkazy;
- **Kompozice** – komplexní, samostatně vykonavatelná operace vázaná na konkrétní číselník, sestavená z aktivit a případně dalších kompozic;
- **Workflow** – komplexní, samostatně vykonavatelná operace bez vazby na konkrétní číselník, sestavená z aktivit nebo kompozic.

Grafický náhled pro prohlížení modelovaných příkazů není jediným náhledem, který uživatel vidí, ale pro zpřehlednění je také k dispozici jeho textová reprezentace, viz následující obrázek.



Obrázek 2 – Textová reprezentace

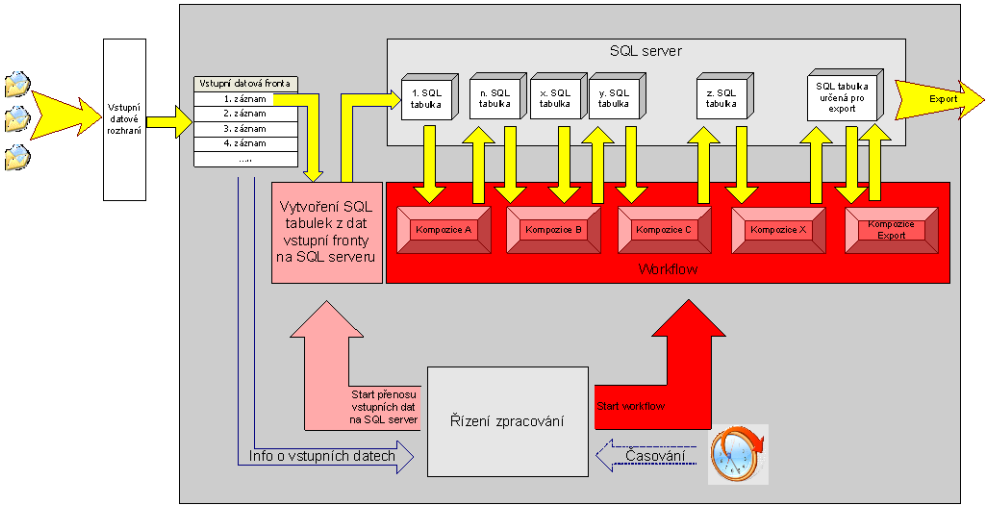
Tato reprezentace pomáhá uživateli v orientaci ve vytvořeném toku zpracování s možností přímé vazby mezi grafickým návrhem a právě touto textovou reprezentací. Tato vazba je vhodná zejména v případě složitějších procesů, kde uživateli vyhovuje čtení textové reprezentace, a pokud vyžaduje její vyhledání v grafickém kontextu, dané dosáhne pouhým kliknutím.

4 Řešení

Grafické rozhraní představené v předchozí kapitole , tvoří jenom jednu část celkového řešení Návrháře procesů, který má ještě následující části:

- **Vstupní rozhraní** – část systému, která zpracovává příchozí data v jednotném formátu XML. Pro zpracování dat ze vstupního rozhraní jsou definována pravidla, která umožňují tato data zpracovávat, jakmile se objeví na vstupním rozhraní dle předem definovaných kritérií (zpracování je možné nastavit také pro určený čas s danou periodou opakování) ;
- **Centrální systém** – hlavní část systému, která zodpovídá za samotnou realizaci funkčnosti a je přístupná jak klientovi, tak externím systémům připojeným přes definované rozhraní;
- **Výstupní rozhraní –** část systému, která zodpovídá za vystavení dat pro odběratele.

Celý systém jak bylo zmíněno má tři části, kde v první z nich dochází k zpracování příchozích XML. Toto vstupní rozhraní je svému okolí přístupno pomocí webové služby. Okolní systémy, zasílají data v nezměněném formátu, který se před zasláním na vstupní rozhraní transformuje do jednotného XML formátu. Další rozhraní je již na výstupu, které svým odběratelům poskytuje data (tabulky) k odběru. Toto rozhraní má implementovanou službu, umožňující odběrateli, který si předplatí odběr dostávat informace o datech, které jsou na tomto rozhraní pro něj k dispozici. Zpracování dat v takto navrženém systému je tedy průtokové bez zbytečných prostojů, pokud uživatel nedefinuje jinak. Na data, která se vyskytnou na vstupním rozhraní, se aplikuje pravidlo zpracování, dle kterého se v systému spustí přiřazený tok, který provede předem specifikované operace a data jsou navrženými operacemi transformována na výstup. Tento datový tok (posloupnost navržených operací) a jeho publikace pro automatické zpracování se navrhuje právě ve zmíněném návrháři.
Návrhář má navíc možnosti nejenom tok navrhovat, ale dále před jeho vlastním publikováním ho odladit na datech, která si uživatel připravil, případně do vstupů tohoto toku zadal. Na následujícím obrázku je zachycené celé zpracování dat, a to jak vstupů tak výstupů.



Obrázek 3 – Zpracování dat

V této části je vhodné, abych se zmínil o bloku „Řízení zpracování“, který řídí samotné zpracování toku dat a dále také popsal pravidla, kterými se tento blok řídí.
Funkce bloku „Řízení zpracování“ je následující: v bloku jsou uživatelem nastavené podmínky, za kterých se mají začít zpracovávat určitá data. Vstupem do tohoto bloku jsou:

- Informace o existenci určitých dat ve vstupní frontě
- Údaje časovače

Uživatel může nad těmito vstupními údaji nastavit různé kombinace podmínek, po jejichž splnění řídicí blok provede tyto úkony:

- Přenos a konverze vstupních dat (pokud byly zadány v podmínce) z datové fronty do SQL tabulek na SQL server aplikace.

- Spuštění zadaných workflow, která zpracovávají vstupní údaje.

V rámci definice pravidel je nutno definovat samotné úlohy (úloha je činnost, při které se za určitých splněných podmínek spustí jedno, nebo více workflow za sebou), které mají definovány atributy, dle kterých se tyto úlohy řídí. Dále bych se již dotknul samotné funkčnosti aplikace, a to v základních rysech. Pro podporu práce s daty bylo zapotřebí integrovat funkčnosti jako jsou například import, export, komentář, modifikace, práce s proměnnými, porovnání tabulek, různé druhy převodů dat, odeslání e-mailu, relace, vytvoření tabulky a další, které vytvářejí společně se základními funkcemi jako jsou cyklus, seskupení, podmínka, komplexní nástroj pro podporu modelování. Tyto aktivity jsou však jenom dílčí části celého systému, které v rámci modelování mohou využít předdefinovaných funkcí. Tyto funkce se rozdělují na:

- **Textové funkce –** pracují s textovými řetězci.
- **Konverzní funkce –** převádí datové typy hodnot.
- **Matematické funkce –** pracují s čísly.
- **Funkce data a času –** pracují s kalendářními daty a časem.
- **Ostatní –** poskytují doplňkové operace s hodnotami nebo proměnnými.

Formát funkcí vychází z jazyka T-SQL. Funkce, které nebylo možné pomocí tohoto standardu naplnit, jsme vytvořili ve formátu PowerShell. Právě v části funkcí PowerShell jsou zpřístupněné části z jazyka MS .NET. To, co uživatel v rámci vytváření toku dat určitě ocení, je nejenom možnost si celý tok zpracování vstupních dat graficky připravit, ale také to, že každá z částí, kterou do tohoto toku vkládá, má v sobě integrované validace, které uživatele upozorní na chybu v zápisu výrazu nebo parametrů. Optimalizace při zápisu znovupoužitelných funkčních bloků sestavených z jednotlivých aktivit, případně celých kompozic, které se vzájemně mezi sebou dají volat pomocí předávaných parametrů, dávají řešení bonus navíc, který uživatelé ocenili již z počátku práce se systémem.

5 Závěr

Správnost návrhu řešení a vhodnost vybraných technologií se nám potvrdily hned při prvních testech v rámci automatického zpracování dat. Tyto testy potvrdili, že systém bez problémů zpracovává stovky megabyte dat denně, a to bez zbytečných prostojů. Takto dostupný systém bylo již na začátku cílem vytvořit, tudíž bychom měli být spokojeni, ale jak se říká, stále je co zlepšovat.

Literatura

[1] KLEIN, Scoot. Professional WCF Programming: .NET Development with the Windows Communication Foundation. Indianapolis (Indiana): Wiley Publishing, Inc., 2007.

[2] PEIRIS, Chris. Pro WCF: Practical Microsoft SOA Implementation. Berkeley (California): Apress, Inc., 2007.

[3] Microsoft Corporation. Windows Communication Foundation: MSDN [online] Microsoft Corporation, c2007. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/ms735119.aspx>.

[4] Microsoft. WCF WCF Client Overview [online]. Microsoft Corporation, c2007. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/ms735103.aspx>.

Otevřená data veřejné správy

Dušan CHLAPEK, Jan KUČERA, Jindřich MYNARZ,
Marek OVEČKA, Martin TAJTL, Vojtěch SVÁTEK
Fakulta informatiky a statistiky, Vysoká škola ekonomická v Praze
nám. W. Churchilla 4, 130 67 Praha 3
chlapek@vse.cz, xkucj30@vse.cz, mynarzjindrich@gmail.com,
xovem02@vse.cz, xtajm02@vse.cz, svatek@vse.cz

Abstrakt. V článku jsou popsány přístupy, možnosti a zkušenosti řešení přípravy aplikace sémantických technologií při publikování dat veřejné správy v ČR. Článek se zabývá způsoby mapování dat, specifikaci metodiky využití standardizovaného nástroje pro katalogizaci digitálně zpracovatelných dat. Dále je proveden přehled aktivit zabývajících se propojováním dat ve veřejné správě ve vybraných vyspělých zemích světa a jsou diskutovány možnosti licencování dat veřejné správy ve světě a v ČR. Je provedeno vyhodnocení výsledků mapování dat veřejné správy, které proběhlo v březnu až květnu 2011. Tato mapování jsou zveřejněná v prostředí CKAN a budou dále rozšiřitelná a využitelná v rámci aktivit otevřené veřejné správy, jako je např. aktivita www.opendata.cz.

Klíčová slova: otevřená data, veřejná správa, ČR, katalog dat, linked open data, CKAN, IPSV.

1 Úvod

V posledních letech neustále roste objem dat, která veřejná správa na různých úrovních zveřejňuje. V zemích, kde tento proces nejvíce pokročil (zejména USA a Velká Británie), již tvoří významnou část vystavovaných dat data opatřená sémantickými metadaty odkazujícími na sdílené slovníky (ontologie). Takto obohacená data je možné do značné míry automaticky propojovat a vyhledávat mezi nimi souvislosti. Případová studie popisuje výsledky první etapy projektu řešeného v rámci interní grantové agentury VŠE Praha „Sémantické propojování dat ve veřejné správě“[1]. Projekt si klade za cíl teoreticky i prakticky ověřit možnosti aplikace sémantických technologií při publikování dat veřejné správy v ČR. Náplní první etapy je provést výchozí mapování dostupnosti důležitých typů dat veřejné správy v ČR (včetně porovnání se situací v "pokročilých" zemích), i s ohledem na formáty jejich poskytování a možnosti jejich transformace na formáty sémantické.

Otevřená data veřejné správy jsou pracovně definována jako data, která zveřejňuje s použitím informační technologií orgán veřejné správy ve formátu umožňujícím jejich další strojové zpracování. A šíření těchto dat není omezeno licenční ani jinou právní podmínkou.

Vzhledem k tomu, že v posledních době se objevuje celá řada aktivit, které pracují nebo se pokouší pracovat s daty zveřejňovanými veřejnou správou, považují autoři případové studie za důležité publikovat výsledky první etapy projektu, ve které proběhlo výchozí mapování dat veřejné správy, informovat o způsobu práce při mapování datových zdrojů, použitých nástrojích, vytvořené metodice a získaných zkušenostech. Předmětem mapování

[1] IGA VŠE, číslo projektu 10/2011

jsou data ve formátech, které umožňují další strojové zpracování. Současně se případová studie dotýká i problematiky licencování dat, která není v podmínkách ČR ještě dostatečně rozpracována a zejména orgány veřejné správy zveřejňují data bez udělení příslušných licenčních oprávnění pro jejich další použití. Na výchozí mapování dat je možno navázat a pokračovat v mapování dalších zdrojů dat. Současně namapované datové zdroje mohou posloužit pro ověřování nových technologií, aplikací a postupů nad daty, která jsou v současné době zveřejněna centrálními institucemi ČR.

Mapování otevřených dat probíhalo v těsné spolupráci s aktivitami Iniciativy za transparentní datovou infrastrukturu [1].

Výsledky mapování dat veřejné správy jsou zveřejněny v instanci nástroje CKAN pro Českou republiku [1].

2 Aktivity v oblasti otevřených dat v ČR a ve světě

Oblast veřejných dat je v současné době předmětem zájmu řady vládních i nevládních institucí, organizací a komunit (např. [7]). V této části příspěvku jsou shrnuty nejdůležitější aktivity, které využívají otevřená data v oblasti veřejné správy. Tyto aktivity v ČR i ve světě poslouží jako příklad a inspirace pro aplikační využití mapovaných otevřených dat veřejné správy.

2.1 Přehled aktivit v oblasti otevřených dat v ČR

V České republice probíhá v současné době celá řada aktivit, které se snaží zprůhlednit fungování veřejné správy, zajistit dohled veřejnosti nad činností politiků, úředníků, komisí, výběrových řízení, veřejných zakázek. V následující tabulce jsou v abecedním pořadí popsány hlavní aktivity dotýkající se problematiky využití otevřených dat veřejné správy v ČR.

Název aktivity	Popis a odkaz na webové stránky
Deník politika	Jedná se o projekt, který vytváří a rozvíjí databázi profilů všech politiků, politických stran, obecních i krajských zastupitelstev, institucí a voleb v České republice. Soustředí se na naprostou neutralitu. Projekt zastřešuje společnost EuroUniversum. *http://www.denikpolitika.cz/*
Inventura demokracie	Inventura demokracie je studentská iniciativa, jež vznikla k 20. výročí 17. listopadu. Zabývá se čtyřmi hlavními tématy a to mechanismy zadávání veřejných zakázek, zamezení praxe legislativních přílepků, depolitizace kontrolních institucí a omezení poslanecké a senátorské imunity. Skupina pořádá veřejné debaty a snaží se přispět k politickým tématům. *http://www.inventurademokracie.cz/*
JakHlasovali.cz	Stránky zabývající se hlasováním jednotlivých poslanců. Na stránkách lze nalézt analýzy od přítomnosti jednotlivých poslanců, přes jejich hlasování a souladu se stranickou příslušností a mnoho dalších zajímavých analýz. *http://www.jakhlasovali.cz/*
OpenData.cz	OpenData.cz je neformální iniciativa pro podporu budování transparentní datové infrastruktury české veřejné správy. Podílí se na ní lidé z různých organizací, například Matematicko-fyzikální fakulty Univerzity Karlovy, Vysoké školy ekonomické nebo Studia nových médií FF UK. *http://www.opendata.cz*
Otevřete.cz	Projekt je realizován Otevřenou společností, o.p.s a podporuje otevřenost veřejné zprávy. Zejména se věnuje veškeré tématice, která je

Název aktivity	Popis a odkaz na webové stránky
	spjata s uplatněním zákona 106/1999Sb. témata, které lze na webu nalézt jsou legislativa, soudní spory, knihovny a účastí na rozhodování. *http://www.otevrete.cz/*
Podněty.cz	Internetová databáze národních občanských podnětů jednotlivých občanů. Cílem je obec a stát bez podnětů, kde vše funguje tak jak má. Projekt je aktivní v Českém Krumlově, Praze, Chrudimi a v Holešově. K roku 2009 má sdružení 35 členů, kteří projekt rozvíjejí. *http://www.podnety.cz/*
PragueWatch	Projekt zobrazující mapu pražských kauz. Pod pojmem kauzy si lze například představit sporné příklady městského plánování, ohrožení kulturních památek nebo administrativní a finanční témata. Na webu jsou některé z informací také zpřístupněny na úředních deskách. Pro spolupráci s občany je využívána šablona v textovém editoru. Jedná se o aktivitu zcela politicky nezávislou. *http://praguewatch.cz/*
Průhledná radnice	Aktivita je zastřešena stranou Svobodných občanů. Touto aktivitou se snaží zprůhlednit činnosti zastupitelstev a kontrolovatelnosti veřejných výdajů, zakázek a závazků ze smluv. Snaží se tento projekt přenést oficiálně přes samosprávu na lokálních radnicích *http://www.pruhlednaradnice.cz/*
Rozklikávací rozpočet	Aktivita, které poukazuje na nepřehlednost zveřejňovaného návrhu státního rozpočtu v ČR a rozpočtů obecně. Jedná se o změť dokumentů. Snaží se tedy navrhnout přístup nebo metodiku k tomuto skrz všechny státní instituce. Na stránce je návrh členění avšak není na to navázána další aktivita. *http://rozklikavacirozpocet.cz/*
Rozpočet veřejně	Projekt je podobného zaměření jako "Rozklikávací rozpočet" se zaměřením na orientaci v rozpočtech obcí, regionů a státu. Na stránkách lze nalézt i Prototyp, který by měl být použit pro prezentaci sbíraných dat. *http://www.rozpocetverejne.cz/*
Veřejný dluh	Jednoduchý projekt, který zobrazuje míru zadlužení a jeho nárůst v každé sekundě v kontrastu s tímto je zde i počítáno HDP České republiky. Cílem je vést lidi k zamyšlení a diskuzi nad tímto tématem. *http://verejnydluh.cz/*
zIndex	Portál zabývající se hodnocením zadavatelů veřejných zakázek, který vznikl na základě nezávislého výzkumu. Na stránkách lze najít metodiku hodnocení, podle které jsou dále zpracovávány výstupy a poskytovány veřejnosti. *http://www.zindex.cz/*
Pilotní ověření zveřejnění vybraných číselníků ČSÚ	Aktivita ve spolupráci VŠE Praha, MFF UK a ČSÚ ohledně zveřejnění vybraných číselníků v RDF formátu. Vyhodnocení výhod a nevýhod nových technologií na pilotní ověření. *Zatím nemá webové stránky.*

Tabulka 1: Aktivity v oblasti využití otevřených dat v ČR

2.2 Přehled aktivit v oblasti otevřených dat ve světě

Výraznější tlak na otevřené zveřejňování dat veřejné správy se poprvé objevil před dvěma lety v USA (v rámci iniciativy „Open Government" prezidenta Obamy) a následně i ve

Velké Británii. S určitým zpožděním se přidávají další západoevropské země, jako je Irsko, Nizozemí nebo Francie.

V USA hraje zásadní roli portál http://www.data.gov/. Přes něj jsou přístupné *katalogizované datové kolekce* (v rozdělení na všeobecné a geografické), *aplikace* všeho druhu určené pro zpracování těchto dat (zejména běžné aplikace, widgety a syndikace RSS; celkem okolo 1000 položek), a dále pak mj. odkazy na další relevantní portály v USA i jinde.

Britský portál http://data.gov.uk/ je sice menší rozsahem, ale ještě o něco více akcentuje moderní technologie zpracování založené na konceptu *Linked Data* [6]. Zahrnuje (na adrese http://data.gov.uk/linked-data) několik koncových bodů ("endpoints") pro dotazování pomocí *jazyka SPARQL* – dotazovacího jazyka pro databáze sémantického webu, standardizovaného konsorciem W3C [14].

3 Licencování dat ve světě a v ČR

Problematika otevřených dat má dimenzi nejenom technologickou, ale i právní. Pokud hovoříme o datech, která jsou otevřená ve smyslu Open Knowledge Definition (dále jen Definice), tedy o situaci, kdy jsou data potenciálním uživatelům dostupná jako celek v podobě umožňující úpravy, jejich další šíření není omezeno, je povoleno provádění úprav dat spolu s šířením vzniklých modifikací a způsob využití dat není nikterak omezen (pro úplný výčet všech podmínek viz [11]), pak je třeba zajistit nejen technický přístup k datům ve vhodném formátu, ale potenciálním uživatelům je třeba také udělit patřičná oprávnění, aby skutečně mohli ve výše naznačeném smyslu s daty nakládat.

Právní ochrana dat se v závislosti na kontextu a povaze vlastních dat může v konkrétním případě řídit různými právními předpisy. Vzhledem k zaměření článku se zaměříme pouze na právní ochranu databází prostřednictvím práva autorského a zvláštního práva pořizovatele databáze, která v českém právním řádu upravuje zákon č. 121/2000 Sb. (autorský zákon, dále AutZ).

Dle autorského zákona je struktura databáze (volba a uspořádání jejího obsahu) chráněna jako autorské dílo souborné, jsou-li splněny podmínky vymezené v §2 odst. 2 AutZ. Databáze může být dále chráněna tzv. zvláštním právem pořizovatele databáze, které je upraveno v Hlavě III AutZ. Dle §88a odst. 1 AutZ „*přísluší zvláštní práva k databázi pořizovateli, pokud pořízení, ověření nebo předvedení obsahu databáze představuje kvalitativně nebo kvantitativně podstatný vklad bez ohledu na to, zda databáze nebo její obsah jsou předmětem autorskoprávní nebo jiné ochrany*". Databáze je pro tyto účely v §88 vymezena jako „*soubor nezávislých děl, údajů nebo jiných prvků, systematicky nebo metodicky uspořádaných a individuálně přístupných elektronickými nebo jinými prostředky, bez ohledu na formu jejich vyjádření*".

Podrobný rozbor obsahu autorského práva a zvláštního práva pořizovatele databáze jde nad rámec tohoto článku. Poznamenejme ale, že dle AutZ nelze s daty (databází) bez předchozího svolení autora nebo pořizovatele databáze nakládat volně ve smyslu Definice (je třeba např. souhlasu k šíření databáze, viz §14 a §94 AutZ). Aby databáze byla otevřená ve smyslu Definice i z právního hlediska, je třeba, aby autor, respektive pořizovatel databáze, udělil uživateli licenci, prostřednictvím které mu umožní výkon práv v Definicí požadovaném rozsahu.

Spojené státy Americké a Velká Británie patří mezi země s rozvinutými iniciativami v oblasti otevřených dat. Zároveň se jedná o země anglosaské právní kultury. V těchto zemích lze právní otevřenost dat řešit kromě udělení licence také věnováním dat do tzv. Public Domain. Díla, která jsou součástí Public Domain, nejsou dle [13] chráněna

autorským právem. Věnováním do Public Domain se dílo stává součástí Public Domain na základě rozhodnutí subjektu, kterému práva k dílu náleží a tento subjekt se tak svých právních nároků zříká. Thaney považuje odevzdání do Public Domain za vhodný způsob řešení právní otevřenosti dat, neboť se jejich poskytovatel zříká právních nároků, které by jinak otevřenosti mohly stát v cestě [15]. Autorka si nicméně uvědomuje, že věnování do Public Domain nemusí být ve všech právních řádech zcela realizovatelné, což je i případ právního řádu ČR, neboť v české právní úpravě se autor svých osobnostních a majetkových práv nemůže vzdát a nelze je ani převést (§ 11 odst. 4 a § 26 odst. 1 AutZ).

Za účelem publikace dat otevřených v souladu s Definicí vznikají standardizované licence a vzorová právní ujednání. Příkladem mohou být licence Open Data Commons Public Domain Dedication and Licence [8] a Open Data Commons Open Database License [9], které vznikly v rámci projektu Open Data Commons, jež je v současnosti jedním z projektů Open Knowledge Foundation [JK8]. Pro účely veřejné správy Velké Británie byla vytvořena licence Open Government Licence [16].

Během mapování datových zdrojů veřejné správy České republiky bylo zjištěno, že úřady zpravidla neuvádějí, za jakých podmínek lze s uveřejněnými soubory nakládat. Vhledem k této nejistotě ohledně práv a povinností vztahujících se k využívání datových souborů uveřejněných na webových stránkách úřadů veřejné správy nebyl žádný z balíčků datového katalogu (viz dále) označen některou z licencí pro otevřená data. V námi identifikované množině datových souborů se tak nepodařilo identifikovat data, která by bylo možno označit za otevřená data ve smyslu Definice.

4 Mapování otevřených dat v ČR

Tato kapitola se věnuje mapování otevřených dat, které proběhlo na ministerstvech a dalších úřadech veřejné správy České republiky. V kapitole je představen nástroj použitý pro vedení katalogu dat, stručně popsán postup provedení mapování a jsou zde shrnuty poznatky z analýzy mapovaných dat.

4.1 Zvolený nástroj

Pro mapování dat byl zvolen nástroj Comprehensive Knowledge Archive Network (CKAN). CKAN je software pro vytváření a správu katalogu dat. Software CKAN je navržen tak, aby s ním mohli pracovat jak lidé, tak i další programy a aplikace. Za tímto účelem CKAN disponuje webovým rozhraním pro interakci s člověkem (viz Obrázek č. 1) a aplikačním rozhraním (API). Pomocí API je možné vytvářet a upravovat balíčky a získat agregované informace o balíčcích, revizích a štítcích.

Záznamy o datech lze do CKAN vkládat, upravovat, vyhledávat a kategorizovat pomocí skupin a štítků (tagů). Uživatelé mohou k záznamům o datech (balíčkům) přidávat svoje komentáře nebo mohou balíčky hodnotit.

CKAN je vyvíjen jako svobodný/open source software. Díky tomu je možné software snadno a bezplatně získat. Zájemci také mohou software upravovat, případně se přímo zapojit do jeho vývoje.

Bližší informace o CKAN je možné nalézt na webových stránkách věnovaných tomuto softwaru [3], [2].

Open Knowledge Foundation (OKFN) [2] také provozuje instanci CKAN pro Českou republiku. Ta je dostupná na adrese cz.ckan.net.

Hlavními argumenty pro zvolení nástroje byla jeho dostupnost a možnost využít nástroj pro mapování dat i přispěvateli mimo řešitele projektu „Sémantické propojování dat ve

veřejné správě“. To je možné díky správě verzí balíčků, která umožňuje řídit příspěvky (úpravy) z řad veřejnosti a následné schválení, nebo editaci provedených změn.

4.2 Metodika použití nástroje CKAN

Pro práci s nástrojem CKAN bylo nutno vytvořit metodiku, která standardizuje použití nástroje pro mapování veřejných dat v ČR. V této kapitole uvádíme stručný výtah z vytvořené metodiky, která bude publikována v [1].

4.2.1 Balíčky dat

Základní jednotkou pro popis dat v software CKAN je tzv. balíček (anglicky package). Balíček obsahuje odkaz na stažitelné soubory s daty nebo na API aplikace, prostřednictvím kterého lze data získat, a popis těchto dat.

Každý balíček CKAN má několik atributů, jejichž hodnoty je možné nastavit během zakládání balíčku nebo upravit u již existujících balíčků. Některé atributy balíčku jsou pevně dané, např. Name (identifikátor balíčku), Title (název balíčku), Licence (licence datových zdrojů) nebo Notes (popis balíčku). K balíčku lze ale přidávat i další vlastní vytvořené dvojice atribut-hodnota (pole označená Extras), kde autor balíčku definuje jak název atributu, tak určuje i jeho hodnotu. Díky tomu je možné přizpůsobit údaje sledované u balíčku vlastním potřebám. V CKAN jsou názvy standardních atributů vytvořeny v angličtině.

Balíčky se vytvářejí pro datové zdroje, které jsou dobře strojově zpracovatelné (např. formáty xls, xlsx, xml, csv). Balíček je možné založit i pro službu (aplikaci), prostřednictvím které lze data získat. Jako zdroj je pak uvedeno API služby.

Do balíčku jsou sdružovány datové zdroje, které spolu souvisí. Pro datové zdroje musí platit všechny atributy balíčku, tj. všechny datové zdroje v balíčku musí být zpřístupněny pod stejnou licencí nebo právními podmínkami, musí jim odpovídat text popisu balíčku, musejí mít stejného autora/původce atd. Pokud spolu datové soubory věcně souvisejí, ale v některém atributu balíčku se liší, pak je třeba pro ně vytvořit samostatné balíčky. V takovémto případě je ale vhodné vytvořené samostatné balíčky opatřit stejnými štítky pro zásadní kategorie, aby uživatel mohl balíčky lépe vyhledat.

Do CKAN se nevkládají samotné datové soubory, ale pouze odkazy na ně. Odkaz na stažitelný soubor nebo na API je v terminologii CKAN označován jako “zdroj” (anglicky Resource). Balíček může mít žádný zdroj nebo několik zdrojů.

Veškeré texty (s výjimkou atributů identifikátoru balíčku „Name“, formátu zdroje „Format“ a polí „Extras“), jsou vytvářeny v jazyce anglickém i jazyce českém. Anglické označení by mělo v pořadí předcházet označení českému.

4.2.2 Štítky (TAGs)

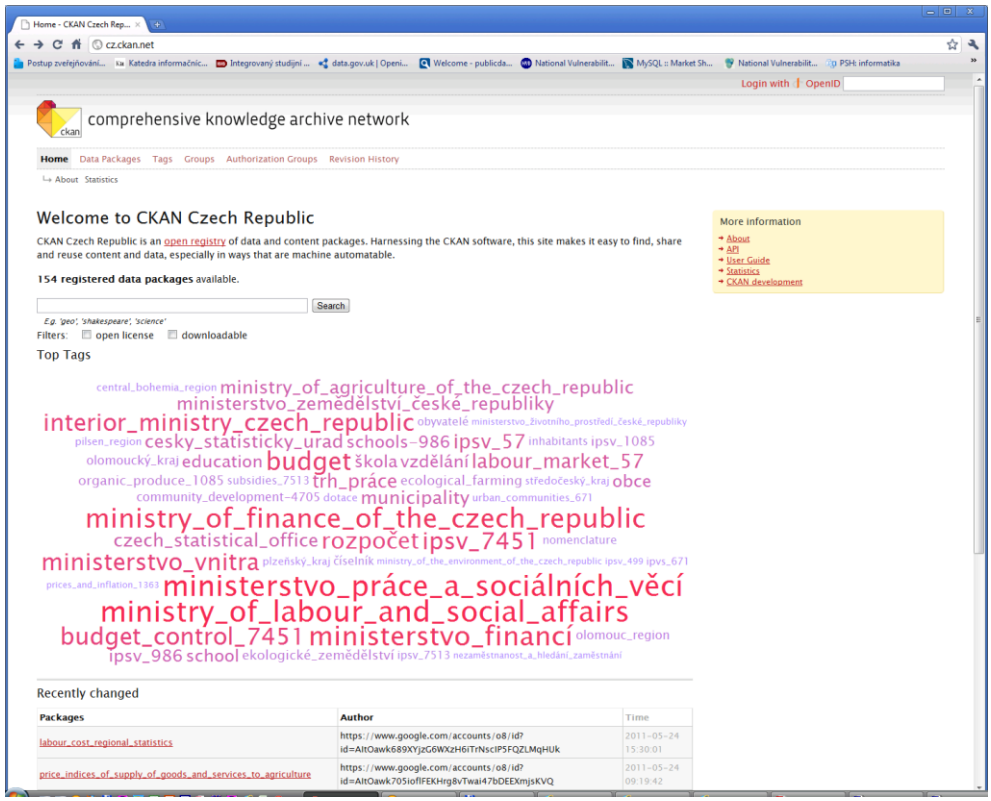
Štítky slouží k označení balíčku věcnými kategoriemi, pomocí kterých jsou popsána hlavní témata obsahu balíčku. Pomocí štítku tak je možno například vyjádřit, že data obsažená v balíčku pocházejí z Ministerstva zemědělství České republiky (přidáním tagu “ministerstvo_zemědělství_české_republiky”, respektive anglickým ekvivalentem “ministry_of_agriculture_of_the_czech_republic”) nebo že se vztahují k ekologickému zemědělství (tagy “ekologické_zemědělství”, “organic_produce”).

Pro štítky v anglickém jazyce jsou primárně voleny pojmy z Integrated Public Sector Vocabulary [4]. Pro označení tagu dle IPSV je voleno preferované znění konceptu IPSV slovníku (to je obsaženo v atributu Name, respektive skos:prefLabel). Kromě štítku, který obsahuje samotný anglický slovní název, je třeba přidat ještě štítek, který obsahuje číslo Id,

pod kterým je pojem evidován v IPSV. Takovýto štítek bude mít prefix "IPSV_" následovaný Id číslem pojmu z IPSV.

Pro české tagy je využíván CZ-NACE tam, kde se podaří vhodný termín v tomto slovníku najít. Pro české i anglické tagy platí, že pokud vhodný pojem v Integrated Public Sector Vocabulary, respektive v CZ-NACE, není, pak jsou tagy vytvářeny jako volně tvořená slovní spojení.

Protože v CKAN je seznam tagů tvořen jako seznam textových řetězců oddělených mezerou, tak v případě, že je tag víceslovný, jsou mezery nahrazeny podtržítkem (např.: "ministerstvo_zemědělství_české_republiky").



Obrázek 1: Ukázka rozhraní nástroje CKAN

4.2.3 Skupiny (Groups)

Kromě označování balíčků štítky (tagy) je možné přiřazovat balíčky do skupin (groups). Uživatelům mohou být udělena práva k úpravě skupin a skupiny lze využít ke klasifikaci balíčku.

Skupiny jsou využity k vyznačení, že se balíčky vztahují k České republice (zařazením balíčku do skupiny "czech_republic").

Skupiny mohou být dále vytvářeny a používány pro seskupování balíčků i z jiných nežli geografických hledisek. Příkladem může být skupina "uep-survey". Zařazením balíčku do

skupiny “uep-survey” je vyjádřeno, že vznikl v rámci projektu IGA “Sémantické propojování dat ve veřejné správě”.

Skupiny je možné vytvářet dle potřeb, nicméně je třeba mít na paměti, že skupiny nenahrazují tagy. Skupiny slouží k seskupení většího množství balíčků, kde skupina nemusí být vždy věcná, ale může mít i administrativní povahu, jako je tomu v případě skupiny “uep-survey”, která vyjadřuje, že balíček vznikl v rámci určitého grantového projektu.

4.2.4 Oprávnění

Nástroj CKAN rozlišuje několik typů oprávnění pro jednotlivé balíčky:

- žádná role v seznamu oprávnění – balíček není viditelný/přístupný
- reader – jen čtení balíčku
- editor – jako reader, plus úpravy balíčku
- admin – jako editor, plus úpravy oprávnění

Oprávnění lze nastavit pro jednotlivé role, jimiž jsou:

- konkrétní uživatel
- autorizační skupina
- Logged-in – role představující všechny přihlášené uživatele
- Visitor – role představující jakéhokoli (i nepřihlášeného) uživatele

Ve výchozím nastavení může balíček upravovat každý přihlášený uživatel. Pokud je třeba zamezit situaci, kdy každý přihlášený uživatel může balíček upravovat, je třeba práva přístupu pro roli “logged_in” nastavit na hodnotu “reader”.

V současné době jsou všechny zmapované datové zdroje přístupné na čtení všem uživatelům, tj. i bez nutnosti přihlášení.

4.2.5 CKAN API

Kromě webového rozhraní lze s CKAN pracovat pomocí API. CKAN API je postaveno na architektonickém stylu REST (REpresentational State Transfer). Každá metoda/funkce je tedy dostupná pomocí svého URI. K jednotlivým funkcím CKAN API je možné přistupovat přímo pomocí protokolu HTTP. Pro získání dat (např. seznamu všech balíčků) je využit HTTP GET požadavek. Pro vkládání dat do CKAN prostřednictvím API je pak použit HTTP požadavek typu POST.

Pro přístup k CKAN prostřednictvím API je k dispozici i několik nástrojů a knihoven pro různé programovací jazyky, které usnadňují tvorbu dalších aplikací využívajících funkčnosti nástroje CKAN. Bližší informace a návody na práci s CKAN API lze nalézt v dokumentaci [12].

4.2.6 Lokalizace a úpravy CKAN

Nástroj CKAN je v současné době provozován OKFN, který připravuje upgrade na vyšší verzi, která poskytne přívětivější uživatelské rozhraní pro vkladatele datových balíčků i pro čtenáře. Současně autoři případové studie připravují lokalizaci (uživatelské rozhraní) do českého jazyka.

4.3 Výsledky mapování - analýzy mapovaných dat

Mapování otevřených dat proběhlo na webových stránkách a portálech úřadů veřejné správy České republiky. Do mapování byla zařazena všechna ministerstva, Český statistický úřad, krajské úřady a 4 okresní města Libereckého kraje. Během mapování jsme se zaměřili na stažitelné soubory ve formátech, které lze dobře strojově zpracovat, zejména na soubory tabulkového procesoru (MS Excel - xls, xlsx apod.), XML soubory a soubor

hodnot oddělených čárkou (csv). Z hlediska věcného jsme se zaměřili zejména na soubory, které obsahují statistiky, výsledky šetření, číselníky nebo finanční a jiné výkazy.

Během mapování dostupných datových zdrojů v rámci vybraných úřadů veřejné správy bylo identifikováno 1470 vhodných souborů, které byly zařazeny do 145 balíčků v nástroji CKAN. Celková velikost zařazených souborů činí po zaokrouhlení na celé megabajty 1079 MB. Pěticí úřadů s největším počtem založených balíčků je po řadě Ministerstvo práce a sociálních věcí České republiky, Ministerstvo financí České republiky, Ministerstvo zemědělství České republiky, Ministerstvo vnitra České republiky a Český Statistický Úřad.

Nejvíce identifikovaných souborů je ve formátu MS Excel verze 2003 (soubor s příponou .xls). Soubory tohoto typu tvoří 93 % celkového počtu všech zařazených souborů. U souborů jiných typů byly identifikovány nejvýše desítky výskytů.

Nejvíce zařazených souborů pochází z Ministerstva práce a sociálních věcí, celkem 814. Většinu z tohoto počtu tvoří soubory, které obsahují údaje o regionální statistice ceny práce (celkem 780 souborů).

Český statistický úřad je orgánem s druhým největším počtem zařazených stažitelných souborů, počet je 218. Na třetím místě z hlediska počtu zařazených souborů se umístilo Ministerstvo vnitra České republiky se 124 stažitelnými soubory.

Ministerstvo práce a sociálních věcí, Ministerstvo vnitra České republiky a Český statistický úřad představují tři orgány veřejné správy, jejichž soubory tvoří největší objem dat v CKAN. Soubory Ministerstva práce a sociálních věcí mají celkovou velikost téměř 570 MB, soubory Ministerstva vnitra mají celkovou velikost téměř 381 MB. Soubory Českého statistického úřadu mají celkovou velikost po zaokrouhlení 41 MB. Za zmínku stojí ještě Ministerstvo zemědělství české republiky, jehož soubory mají celkovou velikost přes 38 MB. Objemy dat na dalších úřadech čítají od desítek kilobajtů až po necelých deset megabajtů.

V souladu s metodikou použití software CKAN bylo pro tematické označení balíčků využito štítků (tagů). V době vzniku tohoto článku bylo v CKAN registrováno 358 aktivně využívaných štítků. Z četnosti využití jednotlivých štítků lze vysledovat, jakých témat se balíčky v CKAN nejčastěji dotýkají. Témata odpovídající štítkům, které byly použity desetkrát a více, jsou uvedeny v tabulce 2.

Téma	**Štítek**	**Počet výskytů**
Rozpočet	budget	21
	budget_control_7451	20
	ipsv_7451	20
	rozpočet	19
Trh práce	ipsv_57	16
	labour_market_57	16
	trh_práce	16
Školství a vzdělávání	education	13
	ipsv_986	13
	school	13
	schools-986	13
	škola	13
	vzdělání	13

Tabulka 2: Témata odpovídající štítkům použitým desetkrát a více

V tabulce 3 jsou uvedena témata odpovídající štítkům použitým pětkrát až desetkrát.

Téma	Štítek	Počet výskytů
Ekologické zemědělství	ecological_farming	9
	ekologické_zemědělství	9
	ipsv_1085	9
	organic_produce_1085	9
Municipalita, obec	municipality	8
	obce	7
Olomoucký kraj	olomouc_region	8
	olomoucký_kraj	8
Dotace	ipsv_7513	7
	subsidies_7513	7
	dotace	6
Číselník	nomenclature	7
	číselník	7
Středočeský kraj	central_bohemia_region	6
	středočeský_kraj	6
Plzeňský kraj	pilsen_region	6
	plzeňský_kraj	6
Městská společenství	urban_communities_671	6
	ipvs_671	6
Životní prostředí	environment_499	5
	ipsv_499	5
	životní_prostředí	5
Nezaměstnanost a hledání zaměstnání	ipsv_468	5
	nezaměstnanost_a_hledání_zaměstnání	5
	unemployment_and_jobseeking_468	5
Ceny a inflace	prices_and_inflation_1363	5

Tabulka 3: Témata odpovídající štítkům použitým pětkrát až devětkrát

Jak již bylo uvedeno výše, úřady české veřejné správy ne ve všech případech uvádějí, za jakých podmínek lze s uveřejněnými soubory nakládat. Během mapování datových zdrojů tak nebyl žádný z balíčků vytvořených v CKAN označen některou z licencí pro otevřená data. Většina balíčků byla v CKAN označena volbou "Licence není uvedena".

5 Závěr

Případová studie shrnuje výsledky první etapy projektu „Sémantické propojování dat ve veřejné správě". Popisuje zejména způsob mapování otevřených dat veřejné správy, rekapituluje zmapované typy, objemy a formáty otevřených dat veřejné správy. Tato katalogizace dat zveřejněná prostřednictvím nástroje cz.ckan.net může být dále tuzemskou veřejností aktualizována a doplňována. Současně jsou v příspěvku představeny příklady aplikací nad otevřenými daty veřejné správy (viz přehled aktivit v ČR a ve světě), které mohou být pro čtenáře inspirací pro další využití strojově zpracovatelných dat veřejné správy.

Výzkum je podporován projektem IGA VŠE č. 10/2011.

Zdroje

1. Iniciativa za transparentní datovou infrastrukturu, 2011. http://www.opendata.cz
2. CKAN wiki documents, http://wiki.ckan.net/Main_Page
3. Comprehensive Knowledge Archive Network,. http://ckan.org
4. Integrated Public Sector Vocabulary, IPSV2_00, http://doc.esd.org.uk/IPSV/2.00.html
5. Katalog veřejných dat ČR, 2011. http://cz.ckan.net
6. Linked Data. Webový portál, online http://linkeddata.org/.
7. F. Maali, R. Cyganiak, V. Peristeras: Enabling Interoperability of Government Data Catalogues. IFIP 2010, Springer eBook, Springer, 2010 http://www.springerlink.com/content/61652w6463052010/
8. Open Data Commons: *Public Domain Dedication and License (PDDL)*, 200?. http://www.opendatacommons.org/licenses/pddl/1.0/.
9. Open Data Commons: *Open Database License (ODbL) v1.0*, 200?. http://www.opendatacommons.org/licenses/odbl/1.0/.
10. Open Knowledge Foundation, http://okfn.org
11. Open Knowledge Foundation: *Open Definition*, 200?. http://www.opendefinition.org/okd/.
12. Open Knowledge Foundation: *CKAN API*, 2009. http://knowledgeforge.net/ckan/doc/ckan/api.html
13. Samuels, E.: The Public Domain in Copyright Law. *Journal of the Copyright Society* 137 (1993).
14. SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. Online http://www.w3.org/TR/rdf-sparql-query/.
15. Thaney, K.: Sharing data on the Web. *Nodalities Magazine* 9 (2010) 7-8.
16. The National Archives: *Open Government Licence for public sector information*, 200?. http://www.nationalarchives.gov.uk/doc/open-government-licence/open-government-licence.htm

Annotation:
In this case study we describe our experience gained during the preliminary phase of the project which is aimed at applying the semantic technologies to publication of data in the Czech public administration. The case study is mainly aimed at data mapping methods, data catalogue software tool usage methodology and legal aspects of data licensing. We discuss also some results of the public administration data mapping conducted from March to May 2011. Mapped data packages are available through Comprehensive Knowledge Archive Network (CKAN) tool.

Opravy geokódů v databázi EPIDAT

Zdena DOBEŠOVÁ, Jan HARBULA, Michael HAVLÍK

Katedra geoinformatiky, PřF UP v Olomouci
Tř. Svobody 26, 771 46 Olomouc
zdena.dobesova@upol.cz
j.harbula@centrum.cz
michael.havlik@seznam.cz

Abstrakt. Tento článek popisuje postup opravy dat v epidemiologické databázi EPIDAT. Opravy se týkají hlavně údajů o místě bydliště pacienta, o místě nákazy a o místě onemocnění. Adresy a místa týkající se onemocnění byly opravovány podle Územně identifikačnímu registru adres UIR-ADR ČR. Část záznamů bylo opraveno automaticky podle chybovníkových tabulek pomocí SQL dotazů. Řada záznamů zůstala v databázi neopravených, neboť jsou neopravitelné. Oprava atributů týkajících se bydliště pacientů a míst nakažení je důležitá pro následné prostorové analýzy šíření infekčních nemocí a hledání prostorových a demografických korelací v Olomouckém kraji.

Klíčová slova: geokódování, infekční nemoci, prostorová databáze, UIR-ADR.

1 Úvod

V dnešní době databáze obsahují velké množství dat. Kvalita dat však limituje jejich následující využití pro analýzy. Výsledky analýz nemohou být spolehlivé, jestliže databáze obsahují chybné záznamy, nebo jsou některé chybné atributy v záznamech. Katedra geoinformatiky Univerzity Palackého v Olomouci spolupracuje s Krajskou hygienickou stanicí v Olomouci, která katedře poskytla část databáze EPIDAT. Databáze EPIDAT obsahuje údaje o infekčních onemocněních. S daty byly plánovány prostorové analýzy pro identifikaci šíření nemocí, sledování jejich časového vývoje a pro identifikaci prostorové závislosti s demografickými daty v rámci Olomouckého kraje.

Výzkumné studie v USA poukazují na důležitost prostorové informace v databázích sledující výskyt rakoviny. Chyby geokódování mají zpětný efekt na špatné statistické vyhodnocení databází o rakovině [10]. Ověření a zhodnocení kvality přesnosti geokódovaných databází doporučují před provedením analýz týkající se hodnocení zdravotních ukazatelů v American Journal of Public Health [7]. Detailnímu geokódování budov je věnována pozornost ve studiích týkajících se nemocí spojených s vysokým stupněm dopravy na ulicích [13]. Spojení adres a výskytu nemocí dovoluje provádět studie závislosti nemocí na životním prostředí a socioekonomických faktorech [1].

Před započetím analýz z dat EPIDATU se proto provedla kontrola dat. U informací o lokalizaci (prostorové informace) pacientů byly nalezeny evidentní chyby. Atributové údaje o diagnózách a časových údajích (datum onemocnění, doba léčby) byly považovány za správné, neboť je poskytli ošetřující lékaři. Před samotným zpracováním bylo tedy nutné provést geokódování, tedy opravu prostorových údajů. Následující části příspěvku popisují

obsah databáze EPIDAT, Územně identifikační registr UIR-ADR a hlavní kroky při opravě geokódovacích atributů pomocí SQL dotazů.

2 Geokódování a databáze EPIDAT

Databáze EPIDAT uchovává informace o pacientech a jejich infekčních nemocech. Kromě nemocí jsou zde uloženy také adresy pacientů. Bylo nezbytné zkontrolovat všechna data o lokalizaci související se zaznamenanými jevy pro následující správné hodnocení prostorové distribuce v Olomouckém kraji. Všechny tři lokalizační údaje, tedy trvalé bydliště pacienta, místo nákazy pacienta a místo onemocnění pacienta, jsou vyplňovány ručně pracovníky hygienické stanice a nejsou žádným způsobem verifikovány k platnému registru adres v České republice.

2.1 Teorie geokódování

Z pohledu teorie existují v geoinformatice dvě metody určení prostorové lokalizace objektů nebo jevů na zemi. První metodou je georeferencování. U této metody je bod určen přímo, definováním souřadnic X a Y v konkrétním souřadnicovém systému. Druhou metodou je geokódování, díky němuž jsou prvky lokalizované nepřímo pomocí geokódů [4, 7]. Gekódy velmi často tvoří adresy, názvy okresů, poštovní směrovací čísla, parcelní čísla, číselné označení vlakové trati apod. Geokód může propojit prvky s pojmenovanými plochami (okres, katastrální území), liniemi (ulice, autobusové linky) a body (číslo domu, název autobusové zastávky). Pro lokalizaci míst událostí spojených s infekčním onemocněním uložených v databázi EPIDAT byla použita tedy druhá metoda – geokódování.

2.2 Epidemiologická databáze EPIDAT

K zajištění povinného hlášení, evidence a analýzy výskytu infekčních nemocí v České republice slouží databáze EPIDAT, která je celostátně používána Hygienickou službou ČR od 1. 1. 1993. Program EPIDAT navazuje na ISPO (Informační systém přenosných onemocnění) z let 1982-1992. Hlášení infekčních nemocí je základem pro místní, regionální, národní a nadnárodní kontrolu šíření infekčních nemocí. Jeho zákonným podkladem jsou závazné předpisy: Zákon č. 258/2000 Sb., o ochraně veřejného zdraví, ve znění pozdějších předpisů a vyhláška MZ ČR č. 195/2005 Sb., kterou se upravují podmínky předcházení vzniku a šíření infekčních onemocnění.

Základní výstupy z databáze EPIDAT jsou zveřejňovány v časopise Zprávy epidemiologie a mikrobiologie, který je možný stáhnout z internetu. Na stránkách Státního zdravotního úřadu je možné sledovat s měsíční periodicitou průběžný stav vybraných hlášených infekcí v ČR.

Do databáze EPIDAT je vyplňováno celkem 53 diagnóz, které patří mezi infekční onemocnění. Databáze je průběžně plněna daty pracovníky krajských hygienických stanic, kteří přepisují lékařské záznamy infekčních hlášení od jednotlivých lékařských zařízení. Každý záznam o onemocnění jednoho pacienta obsahuje 50 atributů. Z pohledu geokódování jsou nejdůležitějšími atributy ULICE, OBEC a OKRES definující trvalé bydliště pacienta, dále MISTONAKAZ (místo nákazy – místo, kde se pacient nakazil) a MISTOONEM (místo onemocnění – místo, kde pacient onemocněl, nejčastěji místo ordinace nebo pracoviště jeho ošetřujícího lékaře).

Program pro zadávání dat nebyl od roku 1993 aktualizován a vývoj nové verze byl zastaven. Rozhraní je zastaralé a chybí v něm řada opatření proti zadávání chybných údajů, zejména z pohledu geoprostorové lokalizace záznamů.

Pro práci byly použity vybrané záznamy z databáze EPIDAT. Data poskytla Krajská hygienická stanice Olomouckého kraje. Poskytnutá datová sada z databáze EPIDAT obsahovala pouze 10 diagnóz infekčních onemocnění (v závorkách je uvedena používaná zkratka): salmonelóza (A02), virová střevní infekce (A08), lymská borelióza (A69.2), klíšťová encefalitida (A84.1), neurčená virová encefalitida (A86), virová meningitida (A87.9), varicella (B01), hepatitida A (B15), akutní hepatitida B (B16) a parotitida (B26). Pro všechny diagnózy byla dostupná data za roky 2004–2008.

Data byla poskytnuta na Katedru geoinformatiky od Krajské hygienické stanice Olomouckého kraje v územní identifikaci za obce Olomouckého kraje. Databáze byla před předáním upravena tak, aby nebyly poskytnuty osobní údaje pacientů. V původní databázi EPIDAT je tedy k dispozici navíc jméno, příjmení, rodné číslo a plná adresa pacienta, včetně čísla popisného bydliště. Z lokalizačních údajů nebylo předáno číslo popisné domu, což ztížilo opravy. Datová sada obsahovala celkem 23 999 záznamů pro čtyři výše uvedené roky a deset vybraných nemocí.

2.3 Územně identifikační registr adres UIR-ADR

Ministerstvo práce a sociálních věcí České republiky garantuje jeden ze speciálních registrů adres – databázi UIR-ADR (Územní identifikační registr adres). Tato databáze obsahuje registry krajů, okresů, obcí, měst, městských částí, pošt, ulic, veřejných prostranství a čísel popisných v České republice [8]. Registr UIR-ADR je uživatelům poskytován zdarma a je průběžně aktualizován. Poslední verze databáze je 4.2. Tento registr byl pro opravu geokódování databáze EPIDAT přijat jako národní standard.

Pro korekci geokódování databáze EPIDAT byly vytvořeny tři pomocné "Korekční tabulky UIR-ADR". Struktura všech tří tabulek je na obrázku 1. První dvě korekční tabulky byly založené na databázi UIR-ADR. Byla to korekční tabulka "Obec" a "Ulice". Hodnoty atributu "NazevObce" v tabulce "Obec" byly konvertovány do velkých písmen, jelikož databáze EPIDAT obsahuje názvy obcí jen v kapitálách a bez diakritiky. Do obou tabulek "Obec" a "Ulice" byl pomocí SQL dotazu z tabulky UIR-ADR přidán dvoupísmenný kód zkratky okresu "ZkratkaOkres" a identifikátor okresu "ID_Okres". Celkový počet záznamů pro ČR je v jednotlivých tabulkách následující. Tabulka Obec obsahuje 62 262 záznamů, tabulka Ulice obsahuje 72 532 záznamů.

Třetí korekční tabulka, "KatastralniUzemi", byla za účelem korekce geokódů převzata z ISKN (Informační systém katastru nemovitostí) Českého úřadu zeměměřického a katastrálního [3]. Tabulka "KatastralniUzemi" obsahuje 13 082 záznamů pro Olomoucký kraj. Důležitou informací je, že k jedné obci může patřit jedno nebo více katastrálních území. Informace o názvu katastru se občas objevuje v poštovní adrese místo pošty. Tato situace je původcem některých chyb a nejasností v databázi EPIDAT.

Obr. 1: Korekční tabulky UIR-ADR

3 Postup opravy dat v EPIDAT

První možnost opravy dat, která připadala v úvahu, byla postupná ruční oprava jednotlivých záznamů. Jelikož databáze EPIDAT obsahovala skoro 24 000 záznamů, byla manuální korekce hodnot zamítnuta jako časově nereálná. Byly proto hledány typy chyb a možnosti jejich automatických oprav. První prohlídka odhalila chybové hodnoty v názvu obcí, ve kterých se objevovali zkratky a překlepy. Název obce byl použit ve třech atributech databáze EPIDAT: místo trvalého bydliště (OBEC), místo nákazy (MISTONAKAZ) a místo onemocnění (MISTOONEM). Kromě zkratek a překlepů se chyby také objevovaly ve špatně zadané obci z důvodu záměny za jinou obec. Často názvy některých suburbanizovaných okrajových městských částí, které patří pod velké město, byly uváděny jako název obce. Do databáze poté byly nesprávně vyplněny názvy městských částí místo správného názvu města

Druhý typ chyb byly špatně vyplněný údaj ULICE u trvalého bydliště nemocného. Chyby vyplývají zejména ze situace, kdy např. ulice v malých vesnicích nejsou pojmenované. Jiným problémem je případ, kdy malá vesnice patří k větší vesnici a tvoří tak jednu obec. Potom v údaji ulice se objevoval název katastrálního území nebo název malé obce. Takové adresy pak obsahují nesprávné informace.

Třetí typ chyb byly nekorespondující informace v kombinaci města, okresu a ulice, např. město patří do jiného okresu nebo ulice neexistuje v daném městě, ale existuje v jiném městě. Řešení tohoto typu chyb bylo nejsložitější.

Nakonec bylo opraveno sedm atributů: obec (OBEC), ulice (ULICE), zkratka okresu (OKRES), místo a okres nákazy pacienta (MISTONAKAZ, OKRESNAK) a místo a okres onemocnění pacienta (MISTOONEM, OKRESONEM). Nejprve byl oddělen a samostatně opraven atribut OBEC (jako část trvalé adresy). Korekce okresu a ulice byla možná jen se současnou verifikací názvu obce. Dále byl přidán ke každému záznamů atribut katastr (KU), který je produktem současné opravy obcí, ulic a okresů.

Lze tedy hovořit o dvou stupních oprav:

- korekce překlepů a zkratek v každém z pěti atributů zvlášť,
- korekce nekorespondujících hodnot atributů v rámci jednoho záznamu.

Některé záznamy byly opravovány opakovaně v různých atributech. Pro účely porovnávání tedy bylo nezbytné zobrazit staré i nové hodnoty těchto opakujících se oprav. Bylo to vyřešeno ponecháním původních starých hodnot a přidáním nových atributů s opravenými hodnotami. Do struktury databáze tedy bylo přidáno sedm nových atributů s předponou OPRAVA (např. atribut pro opravený okres tedy nese název OPRAVA_OKRES). Do databáze EPIDAT bylo dále přidáno postupně sedm nových informačních atributů s předponou POZNAMKA, které nesou informace o jednotlivých provedených opravách a jejich důvodech. Například atribut POZNAMKA_OBEC_OKRES byl přidáván při každé opravě nalezení nesouladu obce a okresu. Poznámka buď obsahuje informaci, co bylo opraveno, nebo obsahuje informaci, že nelze opravit hodnotu z důvodu nejednoznačnosti (obce stejného jména se nachází ve dvou okresech). Navíc byl ještě přidán nový atribut pro katastrální území KU.

3.1 Oprava zkratek a překlepů

Proces oprav se skládal z několika kroků. Prvním krokem bylo vyhledání zřetelných chyb pro každý atribut a vytvoření speciálních dvousloupcových tabulek tzv. chybovníků. První sloupec byl určen pro chybnou podobu hodnoty a druhý sloupec byl určen pro správnou hodnotu atributu. Druhým krokem bylo manuální naplňování tohoto chybovníku opravenými hodnotami z “Korekční tabulky UIR-ADR”, která byla vytvořena na základě

registru UIR-ADR (zmíněného v kapitole 2.3). Třetím krokem byla automatická aktualizace všech chybných záznamů v databázi EPIDAT podle nových atributů z chybovníku vyplněním atributu OPRAVA.

Proces korekce byl kombinací automatického a manuálního postupu. Manuální úroveň byla nezbytná, protože nastavení správných hodnot bylo někdy obtížné a záleží na operátorově geografické znalosti, zejména v kombinaci více chyb v jednom záznamu (nesprávná ulice se kombinovala s chybným názvem místní části obce a se špatným okresem). Kombinace automatické identifikace chyb, manuální naplňování chybovníků a automatická aktualizace záznamů v EPIDATu přinesla dobré výsledky při opravování dat.

Názvy obcí byly v původní databázi EPIDAT vyplněny na Krajské hygienické stanici z jednoduchého interního číselníku nebo manuálně operátorem. Tento číselník obsahuje jen obce velkými písmeny a bez diakritiky. Víceslovné názvy v interním číselníku obsahují zkratky (např. BYSTRICE P.HOSTYN., správně Bystřice pod Hostýnem). Překlepy a zkratky byly tedy velice časté.

Stejné chyby se vyskytovaly víckrát. Do chybovníku byly zařazené pouze jedinečné výskyty chyb. Celkem chybovník pro obce obsahoval 21 těchto jedinečných chyb. Celkový počet opravených záznamů v atributu Obec v databázi EPIDAT podle chybovníku obcí byl 323.

	CHYBNE_OBCE	SPRAVNE_OBCE
	BRODEK U PROSTEJ.	BRODEK U PROSTEJOVA
	BYSTRICE P.HOSTYN	BYSTRICE POD HOSTYNEM
	CELECHOVICE NA H.	CELECHOVICE NA HANE
	DOMASOV N.BYSTRIC	DOMASOV NAD BYSTRICI
	DOMASOV U STERNB.	DOMASOV U STERNBERKA
	DVUR KRALOVE N.L.	DVUR KRALOVE NAD LABEM
	HRADEC N.MORAVICI	HRADEC NAD MORAVICI
▶	HUSTOPECE N.BECV.	HUSTOPECE NAD BECVOU
	KOVALOVICE-OSICA.	HUSTOPECE NAD BECVOU
	LOUCNA	HUTISKO-SOLANEC
	MEROVICE N.HANOU	HUZOVA
	MESTO LIBAVA	HVEZDLICE
	MILOTICE N.BECVOU	HVEZDONICE
	MIMO UZEMI CR	HVEZDONOVICE
	NOVE MESTO NA MOR	HVOZD
	OLSANY U PROSTEJ.	HVOZD

Obr. 2. Příklad chybovníku pro opravu chyb v názvu obce

Stejné korekce byly provedeny pro místo onemocnění (MISTOONEM) a místo nákazy (MISTONAKAZ). Variabilita těchto míst byla větší, protože místo nákazy mohlo být v jiném kraji nebo úplně mimo území České republiky (Hurgada-Egypt, Chorvatsko, …). Hodnota atributu SPRAVNY_MISTONAKAZY v těchto případech byla stanovena jednotně pro všechny záznamy na hodnotu "Mimo území ČR". Opraveny byly všechny záznamy pro atributy OBEC, MISTONAKAZY a MISTOONEM.

3.2 Oprava okresů, ulic a kombinací chyb

Po opravě obcí následovala oprava okresu. Kontrolovalo opět pomocí chybovníků a korekčních tabulek, zda obec (místo nákazy a místo onemocnění) má správně uveden okres. Kontrolovaly se tedy tři atributy s informací o okrese.

V případě chybovníku okresů obsahoval tento chybovník 191 záznamů, což odpovídalo 2 757 opraveným záznamům v databázi EPIDAT u trvalého bydliště pacienta. Na úrovni kombinace chyby obce a okresu u trvalého bydliště zůstalo chybných přibližně 60 záznamů.

Navíc zůstalo v atributech místa a okresu nákazy a místa a okresu onemocnění 115 záznamů neopravených [5].

Velmi problematická byla korekce atributu ulice (ULICE). Názvy ulic totiž v malých vesnicích velmi často chybí. V poštovní adrese byl často použit název malé vesnice jako místní části obce (tedy často název katastru) a vedle toho navíc název obce (s poštou). Operátor vyplnil tedy do atributu ULICE název vesnice (katastru) nebo jiný nesprávný lokální název. Problém byl vyřešen přidání dalšího atributu navíc s názvem KU (katastrální území).

CHYBNA_OBEC	CHYBNY_OKRES	OPRAVA_OBEC	OPRAVA_KU	OPRAVA_OKRES	POZNAMKA_obec_OKRES	ZPRAVA O CHYBACH_obec_OK
LETOHRAD	PR	LETOHRAD		UO	oprava okres	Oprava OKRES nebo OBEC
LHOTA	OC	LHOTA		XX	NELZE SPRAVNE OPRAVIT	Oprava OKRES nebo OBEC
LHOTA	PV	BRODEK U KONICE	LHOTA U KONICE	PV	oprava obec, ku	Oprava OKRES nebo OBEC
LIBINA	OC	LIBINA		SU	oprava okres	Oprava OKRES nebo OBEC
LIPINKA	OC	LIPINKA		SU	oprava okres	Oprava OKRES nebo OBEC
LIPNIK	PR	LIPNIK NAD BECVOU		PR	oprava obec	Oprava OKRES nebo OBEC

Obr. 3. Příklad opravených záznamů

Jednotlivé kroky opravy ulic byly stejné jako v případě oprav obcí. Automaticky byl vyplněn další chybovník - ulic, podle jedinečných chyb v databázi EPIDAT. Správné informace byly přidávány ručně podle korekční tabulky UIR-ADR Ulice. Nakonec byla podle tohoto chybovníku automaticky aktualizována databáze EPIDAT. Zajímavá situace je ukázána na obr. 4. Stejná ulice "Bezručova" existuje jak v obci HORKA NAD MORAVOU jako ulice s názvem "P. Bezruče", tak v obci UNIČOV jako "Bezručovo nám.". U každého záznamu byly osobou, která korekci provedla, zapsány do posledního sloupce poznámky (POZNAMKA) zprávy o provedené opravě.

CHYBNE ULICE	SPRAVNA OBEC	SPRA	SPRAVNA ULICE	POMOCNA OBEC	POMOCNY OK	POZNAMKA
Bezručova	HORKA NAD MORAVOU	OC	P. Bezruče			oprava ulice
Bezručova	LIPNIK NAD BECVOU	PR	P. Bezruče			oprava ulice
Bezručova	PROSTEJOV	PV	Bezručovo nám.			oprava ulice
Bezručova	UNICOV	OC	Bezručovo nám.			oprava ulice
Bílá Lhota	BILA LHOTA	OC				nazev obce
Bilsko	CHOLINA	OC		BILSKO		oprava obec
Bílsko	BILSKO	OC		BILSKO		oprava obec
Bílsko	CHOLINA	OC		BILSKO		oprava obec
Biskupice	CHOLINA	OC		BISKUPICE	PV	oprava obec, okres

Obr. 4. Příklad chybovníku pro opravu ulic s poznámkou o opravě

Chybovník ulic obsahoval 2 675 jedinečných záznamů. Celkově bylo na základě chybovníku ulic opraveno v databázi EPIDAT 8 403 záznamů. V atributu ULICE mělo 6 529 záznamů hodnotu NULL. Zůstalo neopraveno 412 záznamů.

3.3 SQL dotazy

Všechny korekce záznamů byly založeny na SQL příkazech: SELECT, INSERT and UPDATE [4, 9, 11]. Chyby v tabulce EPIDAT byly identifikovány pomocí skupiny šesti dotazů SELECT, které spojili tabulku EPIDAT pomocí levého vnějšího spojení (LEFT OUTER JOIN) s korekčními tabulkami UIR-ADR. Nesouhlasící záznamy, připojené levým vnějším spojením, byly vybrány pomocí podmínky „WHERE IS NOT NULL" a „NULL".

Příklad dotazu SELECT pro odhalení nesprávných jedinečných záznamů v atributu s názvem obce:

```
SELECT DISTINCT Epidat.Obec
FROM Epidat LEFT JOIN Obec ON Epidat.Obec = Obec.NazevObce
WHERE (((Epidat.Obec) Is Not Null) AND ((Obec.NazevObce) Is
Null));
```

Šest chybovníků bylo vyplněno pomocí dotazů INSERT založených na předchozích dotazech SELECT. Oba druhy těchto dotazů sloužily pro první automatickou identifikaci chybových záznamů. Ve druhém kroku oprav bylo manuální plnění chybovníků správnými atributy prováděno taktéž pomocí dotazu SELECT jako zdroj seznamu položek ve druhém sloupci (Obr.1).

Příklad dotazu INSERT pro přidání záznamu do chybovníku:

```
INSERT INTO Chybovnik (Chybna_Obec)SELECT DISTINCT …..;
```

Nakonec dotazy UPDATE podle chybovníků automaticky aktualizovaly v EPIDATU nové hodnoty sedmi nových geokódovacích atributů a také sedmi poznámek.

```
UPDATE Epidat SET OPRAVA_T =Chybovnik.SPRAVNA_T WHERE …. ;
```

Pro všechny stupně korekcí bylo použito více než 40 dotazů. Záznamy v chybovníku bylo možné smazat pomocí dotazu DELETE při opakovaném plnění. Pro použití dotazů a navigaci v jednotlivých krocích opravy EPIDATu byl navržen průvodce (Obr. 5). Tento průvodce byl navržen realizován v programu Microsoft Access 2003.

Obr. 5. Rozhraní průvodce opravou databáze EPIDAT

4 Propojení databáze EPIDAT s geografickou lokací

Finální proces geokódování je propojení geografických souřadnic k neprostorovým datům. Pro toto prostorové spojení je vždy zapotřebí mít dvě skupiny dat. První skupinou dat jsou vstupní data, která se zpracovávají. Druhou skupinu tvoří data, se kterými budou vstupní data konfrontována. Druhá sada je relační databáze, která obsahuje kromě atributů i grafickou složku. Tou mohou být například polygony měst a obcí, adresní body a podobně. V poslední řadě je zapotřebí programu, který nám umožní propojení provést.

Nejpřesnější úrovní geokódování v České republice je geokódování s přesností na adresní bod. Tato úroveň vyžaduje shodu v názvu ulice a čísle popisném, případně orientačním čísle domu. Pokud se záznam nepodaří dohledat s přesností čísla domu, je umístěn na střed ulice. Jedná se o další úroveň geokódování, která je však stále většinou zcela dostačující pro velkou většinu geografických analýz. Pokud i této úrovně přesnosti není možné u některých záznamů dosáhnout, je záznam umístěn na střed části obce nebo na střed obce. Databáze EPIDAT byla kódována nejdříve s přesností na polygon obce.

Geokódování bylo provedeno v prostředí software ArcGIS Destkop. Opravená databáze EPIDAT byla exportována do výstupních tabulek ve formátu XLS. Samotný proces geokódování začal tvorbou tzv. adresového lokátoru. Tím byl dataset v programu ArcGIS, který uchovává atributy adresy, přidružené indexy a pravidla, definující proces přeložení neprostorových popisů míst, jako jsou adresy ulic, do prostorových dat, které mohou být zobrazeny jako prvky v mapě [12]. Adresovému lokátoru je nutné nastavit jeho styl, vybrat referenční data a definovat pole mapy. Styl adresového lokátoru musí odpovídat způsobu kódování adres v dané zemi. Pro Českou republiku odpovídá v ArcGIS styl nazvaný World Cities with Country.

Referenční data představovala třídu prvků (vrstvu), pomocí které byla data z relační databáze geokódována. V případě dat databáze EPIDAT byla použita bodová vrstva obcí pro ČR. Stejně tak ale mohla být zvolena polygonová vrstva obcí nebo katastrálních území. Po dokončení geokódování byla vytvořena nová vrstva se všemi atributy z tabulky a adresového lokátoru. Nyní bylo možné provádět různé vizualizace a prostorové analýzy.

5 Závěr

Kvalita prostorových informací v databázi EPIDAT je omezená. Detailní přehled geokódovacích atributů z poskytnutých vybraných dat z databáze EPIDAT pro Olomoucký kraj přinesl výsledky, které jsou uvedeny v tabulce 1. Opravit záznamy pro obce, místo nákazy a místo onemocnění se podařilo u většiny dat. Nejhorší situace je v atributu ulice, kde bylo více než 35 % chybných záznamů [5] způsobeno množstvím typografických a geografických chyb. Ulice byla opravena u 33% záznamů. Zbylá dvě procenta záznamů zůstala neopravitelná.

Tabulka 1. Počet správných, prázdných, chybných a opravených záznamů v databázi EPIDAT

Atribut	Správné záznamy	Prázdné záznamy	Chybné záznamy	Opravené záznamy
Obec	92,73 %	0,00 %	7,27 %	7,27 %
Ulice	37,78 %	27,21 %	35,01 %	33 %
Místo nákazy, okres	91,27 %	0,49 %	8,24 %	7,9 %
Místo onemocnění, okres	91,85 %	0,03 %	8,12 %	7,6 %

Projekt realizovaný na Katedře geoinformatiky nalezl způsob, jak opravit chybné záznamy s pomocí národního registru UIR-ADR a katastrálního registru. Postup korekce byl poloautomatický proces založeným na SQL dotazech, pomocí kterých byl automaticky identifikovány chyby a nakonec dotazy automaticky databázi EPIDAT aktualizovaly. Ruční vkládání správných hodnot operátorem bylo ale nezbytné. Další nezbytností pro provedení oprav je dobrá znalosti regionální geografie, zejména v případě kombinace více chyb v rámci jednoho záznamu pro malé obce. Chybovníky lze využít opakovaně pro novou sadu dat, buď se všemi diagnózami, nebo s aktuálnějšími daty. Stejným způsobem lze provést opravu databáze EPIDAT pro jiný region než Olomoucký kraj. Celý projekt je přínosný v tom, že nalezl postup, jak opravit databázi EPIDAT. Tento postup je opakovaně aplikovatelný na databázi EPIDAT pro některý další ze čtrnácti krajů v České republice. Také lze nasadit na opravu dat z dalšího časového období, než byly uvedené čtyři roky 2004-2008. Testovací databáze také obsahovala pouze záznamy pro 10 nemocí a tak se předpokládá zopakování postupu pro všechny infekční nemoci. V tomto směru bude pokračovat spolupráce s Krajskou hygienickou stanicí v Olomouci

Užitečným výsledkem je také následující doporučení pro vylepšení nové verze programu EPIDAT: Je nezbytné převzít Územně identifikační registr UIR-ADR a z něj importovat číselníky adres do této aplikace, včetně automatické aktualizace podle průběžné aktualizace UIR-ADR. Dále by bylo užitečné doplnit do evidence údaj o astrálním území a to výběrem z číselníku, který by byl založen na datech z ISKN.

Úroveň kvality prostorové informace by také vzrostla, pokud by se kontrolovaly i domovní adresy. Tato data nebyla z důvodu ochrany dat z Krajské hygienické stanice předána. Z nabytých zkušeností lze předpokládat, že postup oprav by se prováděl analogicky jako opravy ostatních geokódů.

Literatura

1. Croner, C.M., Sperling, J., Broome, F.R.: Geographic information systems (GIS): new perspectives in understanding human health and environmental relationships, *Statistics in Medicine* 15 (1996), 1961–1977.
2. Cromley, E.K., McLafferty, S.L.: *GIS and public health*, The Guilford Press, New York, 2002.
3. Český úřad zeměměřický a katastrální. (2010). Dostupné: <http://www.cuzk.cz>
4. Dobešová, Z.: *Databázové systémy v GIS*. Vydavatelství University Palackého University, Olomouc, 2004.
5. Havlík, M.: *Průvodce geokódováním zdravotnických dat databáze EPIDAT*, bakalářská práce, Katedra geoinformatiky, Univerzita Palackého, Olomouc, 2010.
6. Geokódování a reverzní geokódování. *Geobusiness 1*. Springwinter, Praha (2009) 38-39.
7. Krieger, N., Waterman, P., Lemieux, K. Zierler, S., Hogan J.W.: On the wrong side of the tracts? Evaluating the accuracy of geocoding in public health research, *Am J Public Health* 91 (2001), 1114–1116.
8. Ministerstvo práce a sociálních věcí: UIR-ADR Územně identifikační registr adres Dostupné: <http://forms.mpsv.cz/uir/>
9. Pokorný, J.: Dotazovací jazyky, Karolinum, Karlova University, Praha, 2002.

10. Rushton, G., Armstrong, M.P., Gittler J., Greene B.R., Pavlik C.E., West M.M., Zimmerman, D.L.: Geocoding in Cancer Research: A Review. *American Journal of Preventive Medicine*. Volume 30, Issue 2, Supplement 1, (2006) 16-24
11. Šimůnek, M.: *SQL*, Grada, Praha. 1999.
12. Wong, A., Crosier, S.: Geocoding in ArcGIS Tutorial*,* ESRI, Redlans, USA, (2008) <Geocoding_in_ArcGIS_Tutorial.pdf>
13. Zandbergen P.A.: Influence of geocoding quality on environmental exposure assessment of children living near high traffic roads, *BMC Public Health* 7, 37. (2007).

Annotation:
The geocode correction in database EPIDAT.
This article describes data correction in epidemiological database EPIDAT. The correction is aimed to correct all information about address of patients, place of infection and the get ill place of patients. Address and a location were compared against the valid Czech Territorial Identification Address Register UIR-ADR. Some records have been automatically repaired by SQL queries. There are also some irreparable records in EPIDAT. Percentage of correct records is more than 90% for attribute town, place of infection and place of patient get ill. The worse situation is in street attribute - 35% wrong records caused by lot of typing errors.
Trial database were taken for Olomouc Region for 10 selected diagnoses for period 2004-2008. Number of records was 24 thousand in trial database. The suggested universal steps of geocode repairing can be repeated to records about other 43 diagnoses or to newly recorded data from 2009 up to the present time. Moreover, the steps of semiautomatic correction can be applied to other 13 separate EPIDAT databases from each Regional Hygiene Station in the Czech Republic. The attribute correction is crucial for further spatial analysis of epidemiological data from the point of spreading diseases and spatial correlations.

Postery

Hodnocení webu v prostředí e-commerce

Kateřina SLANINOVÁ, Petr SUCHÁNEK

Katedra informatiky, SU OPF Karviná
Univerzitní náměstí 1934/3, 733 40 Karviná
slaninova@opf.slu.cz, suchanek@opf.slu.cz

Abstrakt. Elektronická komerce představuje v dnešní době standardní a velmi silný nástroj pro podporu obchodních aktivit. Jednou z klíčových oblastí e-commerce systémů jsou jejich webová rozhraní, která jsou na úrovni Business to Customer představována webovými stránkami internetových obchodů. Klíčovou podmínkou pro získání maximálního efektu vyplývajícího z účelu, pro který byly vytvořeny, je zajištění jejich přístupnosti. Článek je zaměřený na problematiku hodnocení přístupnosti vybrané skupiny webových stránek internetových obchodů dle doporučení Web Content Accessibility Guidelines a podmínek pro optimalizaci pro vyhledávače.

Klíčová slova: e-commerce, webové prezentace, přístupnost webových prezentací

1 Úvod

Úspěšná webová prezentace, ať už je jakéhokoliv zaměření, by měla z jedné strany respektovat všechna pravidla marketingového mixu [1], z druhé strany by měla respektovat moderní požadavky z hlediska webdesignu, návrhu navigační struktury, optimalizace pro vyhledávače a dodržování doporučovaných standardů.

Autoři se v článku zabývají základním komunikačním rozhraním e-commerce systémů (zejména na úrovni B2C – Business to Customer), kterými jsou webové stránky internetových obchodů. V textu jsou uvedeny vybrané metody monitorování a hodnocení přístupnosti webových stránek vybraných internetových obchodů (e-shopů). Hodnocení bylo provedeno v souladu s doporučeními WCAG (Web Content Accessibility Guidelines), vydanými organizací W3C (World Wide Web Consortium), a podmínkami vyplývajícími z optimalizace pro vyhledávače (SEO – Search Engine Optimization). Článek vznikl na základě podkladů vyplývajících z řešení projektu OP VK č. CZ.1.07/2.3.00/09.0197 - Posílení konkurenceschopnosti výzkumu a vývoje informačních technologií v Moravskoslezském kraji.

2 Přístupnost webových prezentací

První verze dokumentu WCAG 1.0 z roku 1999 obsahuje 14 doporučení[1], očíslovaných tematicky dle obecných principů využívaných při designu přístupných stránek. WCAG 1.0 se stala podkladem pro další metodiky (Section 508, Blind Friendly Web apod.). Od roku 2008 je v platnosti metodika WCAG 2.0, která vznikla jako reakce na již nevyhovující WCAG 1.0. Tato metodika je rozdělena do 4 základních principů, pro každý z principů je

[1] http://www.w3.org/TR/WCAG10/

definováno několik pravidel, která jsou testována pomocí tzv. kontrolních kritérií vyjádřených třemi úrovněmi: A (nejnižší), AA a AAA (nejvyšší).

Vytvořený moderní web by měl odpovídat doporučeným standardům konsorcia W3C také z hlediska kódu jazyka, který byl při tvorbě použit. Provádí se tedy tzv. kontrola validity kódu.

Základní podmínky týkající se vlastností, funkcionalit a správy www stránek internetového obchodu v návaznosti na jejich přístupnost a maximalizaci efektivity jejich hlavního účelu můžeme shrnout do následujících bodů [2]: optimalizace pro vyhledávače, atraktivnost, snadná a intuitivní ovladatelnost, rychlost, bezpečnost, propojitelnost, návratnost investic a zisk, modulárnost, možnost analýzy dat z internetového obchodu, monitorování webu.

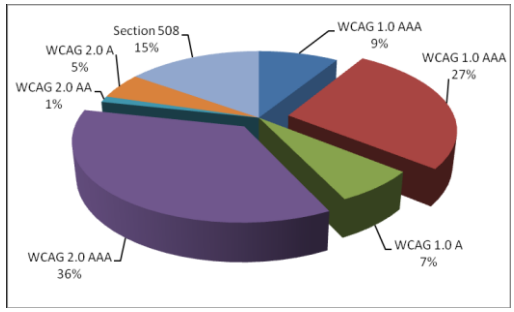
3 Analýza webových rozhraní vybraných internetových obchodů

Autoři provedli analýzu vybraných internetových obchodů na základě doporučení WCAG 1.0 a WCAG 2.0. Výběr testované množiny byl proveden na základě ratingu webových stránek provedeného vyhledávačem Google. Zvoleno bylo celkem 7 okruhů dle zaměření internetových obchodů (nejprodávanější produkty dle Českého statistického úřadu za rok 2010): vstupenky, elektronika, knihy, kosmetika, mobilní telefony, sportovní potřeby, počítače, plus dva zástupci internetových obchodů zaměřených všeobecně a dva zástupci srovnávacích agentů prohledávacích internetové obchody. Analýza byla provedena pro tyto oblasti:

1. Testování chyb v URL adresách, neplatných odkazů, chyb ve skriptech apod.
2. Analýza přístupnosti na základě WCAG 1.0, WCAG 2.0 a Section 508. Kontrolní kritéria byla rozdělena dle jednotlivých priorit (A – AAA).
3. Testování kompatibility s vybranými typy nejpoužívanějších prohlížečů (IE 6.0 – 9.0, Firefox 2.0 – 3.6, Safari 5.0, Opera 11.0, Chrome 10.0) včetně prohlížečů mobilních zařízení (iPhone 4.0, Android 3.0).
4. Shoda s vybranými zákony a doporučeními (US CAN SPAM Act 2003[2], EU Privacy and Electronic Communications Regulations 2003[3] a dodržování autorských práv.
5. Optimalizace pro vyhledávače – na základě doporučení pro vybrané vyhledávače (Google Search Guidelines, Bing Search Guidelines a Yahoo Search Guidelines).
6. Validita kódu (HTML/XHTML, CSS) a hodnocení na základě W3C Style Guide for Online Hypertext[4].
7. Testování použitelnosti dle Usability.gov a W3C Best Practices.

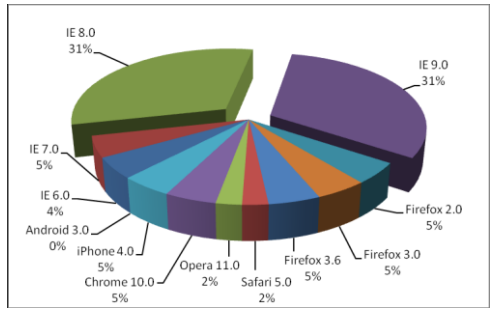
Testováno bylo celkem 32 www stránek. Pro testování stránek byl použit software Sort Site 4 Evaluation. Podíváme-li se na jednotlivé oblasti podrobněji, bylo zjištěno, že v oblasti přístupnosti vyhověl pouze jeden testovaný web – levneELECTRO.cz. Z grafu v Obr. 1 je patrné, že ostatní weby měly v průměru nejvíce problémy s doporučením WCAG 2.0 (priorita AAA tvořila až 36% ze všech zjištěných nedostatků) oproti WCAG 1.0 (priorita AAA pouze 9%). Doporučení WCAG 2.0 nejvyšší priority AAA zahrnují např. unikátní titulky pro každý web, unikátní ID u objektů, popisky u ovládacích prvků formulářů, chyby způsobující potíže čtečkám, zatímco doporučení WCAG 1.0 zahrnují především základní požadavky jako nechybějící titulky, alternativní texty obrázků, rámů apod.

[2] http://en.wikipedia.org/wiki/CAN-SPAM_Act_of_2003
[3] http://www.ico.gov.uk/for_organisations/privacy_and_electronic_communications.aspx
[4] http://www.w3.org/Provider/Style/



Obr. 1. Analýza přístupnosti na základě WCAG 1.0, WCAG 2.0 a Section 508.

Z analýzy kompatibility vyplynulo, že nejvíce měly internetové obchody problém s prohlížečem IE ve verzi 8.0 a 9.0, mobilní technologie byly vesměs všude dobře podporovány (viz Obr. 2).

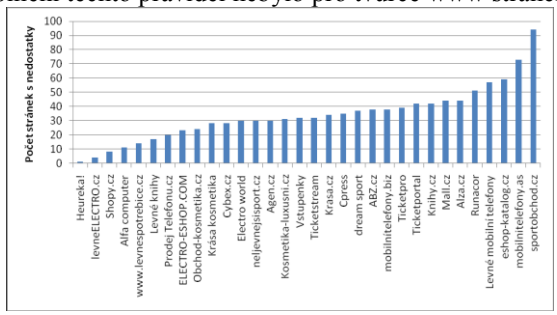


Obr. 2. Testování kompatibility s vybranými typy nejpoužívanějších prohlížečů.

Celkové srovnání testovaných internetových obchodů dle počtu stránek, u kterých byly zjištěny nedostatky dle metodiky uvedené výše, je zobrazeno na Obr. 3.

Je patrné, že kvalita webového rozhraní testovaných internetových obchodů nezávisí na jejich zaměření. Nejlépe se v hodnocení umístil srovnávací web Heureka!, drtivá většina internetových obchodů měla zjištěny nedostatky u 30 % stránek ze 100 testovaných.

Z výsledků testování dle pravidel WCAG je vidět, že starší pravidla nejsou již dnes problémem, zjištěné nedostatky nebyly příliš závažné. Ovšem ještě vysoké procento www stránek má problém s novými pravidly. Důvodem může být fakt, že buď www stránky vznikly dříve, nebo splnění těchto pravidel nebylo pro tvůrce www stránek prioritou.



Obr. 3. Celkové srovnání na základě doporučení WCAG 1.0 a WCAG 2.0.

Hodnocení zaměřené na oblast SEO ukázalo, že pro firmy není prioritou sledovat nové trendy vyhledávacích algoritmů a pravidel využívaných internetovými vyhledávači. Dávají spíše přednost vytvoření takových stránek, které osloví předpokládaný počet uživatelů. Z tohoto hlediska je pro firmy jednodušší zaměřit se spíše na SEM - Search Engine Marketing a zaplatit si přímo reklamu ve vyhledávači (AdWords apod.) nebo se zaměřit na výměnné odkazy atd.

Zcela specifickou oblastí je oblast legislativy. Programy určené pro testování webových stránek se zaměřují především na oblast dodržování autorských práv. Zde testované weby obstály v 75%. Mnohem větší problémy měly weby s dodržováním, práv EU – až 97%. Vzhledem k tomu že se autoři zaměřili pouze na české e-commerce systémy, neměl by tento fakt příliš omezit jejich fungování. Tato informace by však měla být brána na zřetel v případě akceptace zahraničních zákazníků.

4 Závěr

Hlavním cílem článku bylo provedení analýzy přístupnosti vybraného vzorku komerčních www stránek internetových obchodů. Na základě statistiky Českého statistického úřadu za rok 2010 bylo vybráno 7 kategorií nejprodávanějšího zboží v ČR, dva zástupci internetových obchodů zaměřených všeobecně a dva zástupci srovnávacích agentů prohledávacích internetové obchody. Na základě hodnocení 7 kriteriálních oblastí se došlo k závěru, že firmy vytvářející a/nebo provozující internetové obchody mají z hlediska přístupnosti jejich webových stránek internetových obchodů ještě značné rezervy zejména v oblastech použitelnosti a dodržení pravidel WCAG 2.0. Je evidentní, že nedostatky zjištěné provedenou analýzou negativně ovlivňují efektivitu www stránek internetových obchodů a tudíž snižují cíle vyplývající z účelu jejich využití. Odstraněním těchto nedostatků by mnohé firmy mohly rozšířit oblast potenciálních uživatelů o ty, kteří dosud měli problémy s přístupem na web (ať už to byli uživatelé s technickými či zdravotními problémy) a zvýšit tak ROI (Return of Investment), což je ve světě e-business samozřejmě prioritou.

Literatura

1. Steinová, M a kol.: *E-marketing II*. Ostrava: VŠB TU, 2003.
2. Suchánek, P.: E-business Development Key Areas. *In 5-th International Symposium on Business Administration. Çanakale: Çanakale Onsekiz Mart University*, 2008.

Title: Website Assessment in E-commerce

Annotation:

The authors of the article deal with the basic communication interface of e-commerce systems which are e-shop web sites. In the text, inter alia, the need for presentation of information on Internet as part of marketing strategy is presented. The key area to support the success of all types of web sites is the issue of their accessibility. The paper presents selected methods for monitoring and evaluating the accessibility of selected group of e-shop web sites. The evaluation was conducted in accordance with the recommendations of Web Content Accessibility and conditions arising from the search engine optimization.

Open source nástroje pro podporu řízení projektů v multiprojektovém prostředí

Jan KUČERA[1]

[1]*Katedra informačních technologií, FIS, VŠE Praha*
nám. W. Churchilla 4, 130 67 Praha 3
xkucj30@vse.cz

Abstrakt. Kroky a akce, které organizace realizují za účelem dosažení svých cílů, mají často podobu projektů. V situaci, kdy v organizaci probíhá více projektů současně, a další jsou plánovány, je třeba řídit nejen projekty samotné, ale i celé portfolio projektů. Řízení portfolia projektů může vyžadovat vyhodnocení značného objemu dat týkajících se projektů, zdrojů a dalších podstatných skutečností, proto může být vhodné uvažovat o nasazení softwarových nástrojů pro podporu této oblasti. Tento článek stručně představuje open source nástroje]project-open[a Project.net, které lze pro podporu uvažované oblasti řízení využít.

Klíčová slova: Open source, Project Portfolio Management, PPM, řízení portfolia projektů,]project-open[, Project.net.

1 Úvod

Kroky a akce, které organizace realizují za účelem dosažení svých cílů, mají často podobu projektů. Svozilová definuje projekt podle metodiky PMBOK jako „dočasné úsilí vynaložené na vytvoření unikátního produktu, služby nebo určitého výsledku“ [6, str. 22]. Prostřední, ve kterém probíhá více projektů, a další jsou plánovány, případně probíhá proces vyhodnocení podnětů pro jejich realizaci, pak bývá označováno jako multiprojektové prostředí (viz. [6]).

V multiprojektovém prostřední je třeba věnovat pozornost nejen řízení jednotlivých projektů, ale i řízení portfolia projektů jako celku, aby bylo zajištěno maximálně efektivní využití zdrojů organizace, a aby do projektového portfolia byly zařazovány takové projekty, jejichž realizace povede k naplnění stanovených cílů organizace. Řízení projektů v multiprojektovém prostředí může vyžadovat vyhodnocení značného objemu dat týkajících se projektů, zdrojů a dalších podstatných skutečností, přičemž získání a zpracování těchto dat může být poměrně náročné. Z tohoto důvodu může být vhodné uvažovat o nasazení softwarových nástrojů pro podporu této oblasti.

Tento článek se věnuje open source nástrojům pro podporu řízení projektů v multiprojektovém prostředí. V první části článku jsou obecně diskutovány důvody, proč uvažovat o nasazení open source nástrojů. Ve druhé části jsou pak vybrané open source nástroje stručně představeny. V závěru článku jsou shrnuty získané poznatky.

2 Proč uvažovat o nasazení open source nástroje

Pro podporu řízení projektů v multiprojektovém prostředí je na trhu k dispozici celá řada nástrojů označovaných jako Project Portfolio Management software (PPM software). PPM nástroje disponují funkcemi pro podporu řízení jak individuálních projektů, tak i jejich portfolií. Řízení projektů v multiprojektovém prostředí PPM nástroje typicky podporují pomocí funkcí pro oblasti správy podnětů pro realizaci projektů, řízení a správy portfolia projektů, řízení projektů a programů, řízení zdrojů, finanční řízení, řízení rizik a reporting. Práce projektových týmů pak bývá podporována funkcemi pro podporu spolupráce, správy dokumentů, workflow a některé nástroje nabízejí také přímou podporu pro uznávané metodiky jako je například PMBOK nebo PRINCE2.

Nástroje pro podporu řízení projektů a jejich portfolií lze nalézt i v oblasti open source softwaru (OSS). Důvodem pro implementaci OSS řešení může být potenciál úspor díky nižším nákladům na licence a jejich správu a možná větší flexibilita v porovnání s proprietárními řešeními daná nezávislostí open source licencemi na hardwaru nebo počtu uživatelů. Tyto obecné přínosy využívání OSS uvádí studie [3].

Ne všechny organizace plně využijí celou šíři funkcí nabízených robustními PPM nástroji [1]. Implementace open source PPM nástroje tak může být vedle využití nástroje v modelu Software-as-a-Service další alternativou k využívání robustních PPM nástrojů.

Dalším důvodem pro využívání open source softwaru může být dostupnost jeho zdrojového kódu a legální možnost jeho úprav. Díky tomu je možné OSS nástroj využít jako základ celkového řešení v případě, že se organizace rozhodne vybudovat vlastní softwarové řešení pro podporu řízení projektů a jejich portfolií. Open source software tak může být využit pro pokrytí typizované funkčnosti nevyžadující specifickou implementaci.

3 Dostupné open source nástroje

Pro podporu řízení projektů existuje v oblasti open source softwaru řada nástrojů, které se liší svým zaměřením, rozsahem funkcí anebo způsobem vývoje. Lze tak nalézt jednak jednodušší nástroje vhodné spíše pro evidenci projektů a základních údajů o nich nebo pro přidělování úkolů jednotlivým členům týmu. Příkladem těchto jednodušších nástrojů jsou dle [2] nástroje Achievo, PHProjekt nebo dotProject.

Oproti těmto jednodušším nástrojům lze ale nalézt i OSS nástroje, které se snaží nabídnout komplexnější podporu pro řízení projektů v multiprojektovém prostředím. Do této kategorie patří nástroje]project-open[a Project.net představené v dalším textu.

Popis nástrojů vychází z práce [2], ve které jsou tyto nástroje podrobně hodnoceny. V této práci je také popsán postup výběru uvedených nástrojů a dále jsou diskutována specifika open source softwaru z hlediska výběru softwarového nástroje.

3.1]project-open[

]project-open[je nástroj pro podporu řízení projektů v organizaci, jehož vývoj je řízen stejnojmennou společností, která také nabízí pro nástroj technickou podporu a další služby (konzultace, školení). Společnosti]project-open[se také podařilo pro nástroj vybudovat širokou partnerskou síť s partnery v řadě zemí světa.

Nástroj]project-open[je webová aplikace vyvíjená v programovacím jazyce Tcl/Tk s využitím platformy OpenACS. Pro provoz aplikace je využíván AOLServer. Architektura nástroje je modulární a jednotlivé moduly jsou distribuovány pod třemi typy licencí. Moduly tvořící jádro nástroje, jsou distribuovány pod open source licencí GNU GPL. Zbylé

moduly jsou distribuovány pod proprietárními licencemi, kde jedna licence je bezúplatná (moduly lze získat zdarma; umožněno je i provádění úprav, nebudou-li upravené moduly dále šířeny) a druhá licence je úplatná, tj. moduly pod touto licencí je třeba zakoupit.

Nástroj disponuje podporou pro tvorbu harmonogramů projektů, jednotlivé úkoly lze přiřazovat pracovníkům a k dispozici je také výkaz práce. Nástroj disponuje jednoduchým úložištěm dokumentů, k dispozici je projektové diskusní fórum a projektová wiki. Funkčnost nástroje je orientována na podporu řízení projektových financí. Umožněna je evidence výnosů a nákladů, pracovat lze také s objednávkami a fakturami. Nástroj disponuje také podporou pro workflow s možností definice vlastních pracovních toků.

V porovnání s robustními nástroji pro řízení portfolia projektů chybí nástroji]project-open[rozsáhlejší podpora pro oblasti správy podnětů pro realizaci projektů, pro řízení a správu portfolia projektů (např. funkce pro modelování portfolia) a pro řízení rizik.

]project-open[umožňuje přímou integraci s několika účetními a ERP systémy. Kromě toho disponuje nástroj i obecným rozhraním založeným na technologii XML-RPC. Pro další verze je plánována podpora pro SOAP webové služby.

Nástroj]project-open[by mohl nalézt uplatnění v organizacích, jež preferují evidenci nákladů a výnosů spojených s projekty před propracovanými funkcemi pro podporu plánování projektů. Využití nástroje jako základu vlastního řešení může ztěžovat zvolená platforma a model licencování, kdy je pod open source licencí šířeno pouze jádro nástroje.

3.2 Project.net

OSS nástroj Project.net vyvíjí společnost Integrated Computer Solutions, Inc. ve spolupráci s komunitou. Tato společnost pro nástroj nabízí také technickou podporu, školení nebo konzultace. V rámci analýzy trhu PPM nástrojů společnosti Gartner [5], která hodnotí stav trhu v polovině roku 2010, byl Project.net zařazen do kategorie tzv. skrytých hráčů (niche players). Nástroj tak sice nepatří k vedoucím nástrojům na trhu, splňuje ale minimální požadavky, aby byl v rámci analýzy trhu hodnocen, což znamená, že výrobce je pro nástroj schopen poskytnout profesionální podporu na adekvátní úrovni kvality a komunita formovaná okolo tohoto open source nástroje je natolik početná, že nehrozí její rozpad.

Project.net je webová aplikace vyvíjená na platformě Java. Distribuována je pod open source licencí GNU GPL, jež pokrývá veškerou funkčnost nástroje. Platící zákazníci získávají mimořádná opravná vydání, která nejsou uvolňována pro potřeby komunity.

Nástroj se zaměřuje na podporu řízení projektů a podporu spolupráce. K dispozici jsou funkce pro tvorbu harmonogramů projektů. Úkoly lze přidělovat jednotlivým pracovníkům, nástroj označuje činnosti na kritické cestě a pro projekt je možné uložit více směrných plánů. Procento dokončení činnosti či projektu může být automaticky kalkulováno na základě údajů z výkazu práce. Dále je možné využít projektového diskusního fóra a wiki, nástroj disponuje také základními funkcemi pro správu verzí dokumentů.

V porovnání s robustními PPM nástroji postrádá tento nástroj rozsáhlejší podporu pro oblasti řízení projektových financí, řízení rizik a pro správu podnětů pro realizaci projektů. Zdroj [2] hodnotí podporu nástroje pro řízení a správu portfolia projektů ve verzi 9.1.3 jako omezenou. Verze 9.2 podporu této oblasti vylepšuje a přináší možnost definice vlastních pohledů na portfolio projektů [4]. Verze 9.2 také přináší integraci s open source business intelligence (BI) platformou Pentaho, což rozšiřuje možnosti reportingu a provádění analýz.

Požadavkům organizace lze nástroj přizpůsobit díky možnosti definovat vlastní workflow. Vytvářet lze i vlastní formuláře pro sběr dat a umožněn je také import a export dat z/do nástroje MS Project.

Nástroj Project.net by mohl nalézt uplatnění v organizacích vyžadujících nástroj, který jim zejména umožní vyvářet harmonogramy projektů a sledovat jejich průběh. Veškerá

funkčnost nástroje je distribuovaná pod open source licencí a nástroj je vyvíjen na standardní platformě Java EE, což umožňuje rozšíření nástroje o další požadované funkce.

4 Závěr

Řízení projektů v multiprojektovém prostředí může vyžadovat vyhodnocení značného objemu dat týkajících se projektů a dalších skutečností, a proto může být vhodné uvažovat o podpoře této oblasti pomocí softwarového nástroje. Pro podporu této oblasti existuje na trhu řada nástrojů, mezi kterými lze nalézt i nástroje vyvíjené jako open source software.

Pokud se organizace rozhodne vybudovat vlastní řešení pro řízení portfolia projektů, je možné zvažovat využití open source nástroje jako jeho základu. Splňuje-li open source nástroj všechny požadavky organizace, může při výběru softwarového nástroje představovat alternativu k proprietárním nástrojům a řešením.

V tomto článku byly ve stručnosti představeny open source nástroje]project-open[a Project.net, které se snaží nabídnout komplexnější podporu pro řízení projektů v multiprojektovém prostředí. Nabídka funkcí není tak bohatá jako u nejrobustnějších PPM nástrojů, v řadě oblastí je ale funkčnost těchto nástrojů dostatečně propracovaná, aby bylo možné uvažovat o jejich nasazení.

5 Poděkování

Tento článek je podporován grantem GAČR P403-10-0092.

Literatura

1. Cardin, L., Cullen, A., DeGennaro, T.: *SaaS-Based Tools Lower Barriers To PPM Success*, 2008. http://www.powersteeringsoftware.com/pdfs/Forrester_20080410.pdf.
2. Kučera, J.: *Výběr a implementace open source nástroje pro řízení portfolia projektů*. VŠE, Praha, 2010.
3. Lyman, J., Asslet, M.: *Climate Change: User Perspectives on the Impact of Economic Conditions on Open Source Software Adoption*. The 451 Group, 2009.
4. Project.net Inc.: *Project.net Releases Version 9.2*, 2010. http://www.project.net/v9_2.
5. Stang, B. D.: *Magic Quadrant for IT Project and Portfolio Management*, 2010. http://www.gartner.com/technology/media-products/reprints/microsoft/vol10/article12/article12.html.
6. Svozilová, A.: *Projektový management*. Grada, Praha, 2006.

Annotation:
Managing of projects in multi-project environment can require an analysis of significant amount of data about project and other important facts. There are software tools which can help with gathering all the necessary data and their subsequent analysis. Some of these tools are developed as open source software. Open source tools]project-open[and Project.net are briefly described in this article. These tools aim at providing complex support to the project management performed in the multi-project environment. However, functionality provided by these tools is not as rich as the functionality provided by the most robust PPM solutions.

Penetrační testování metodikou OSSTMM

Jiri BARTOS[1]

[1]*KIP Ostravská Univerzita v Ostravě*
30. dubna 22, 701 03 Ostrava
jiri.bartos@proit.cz

Abstrakt. Cílem příspěvku je popis a seznámení s metodikou OSSTMM používanou pro testování a prověřování informační bezpečnosti a to jak pro technické penetrační testy, sociální testy, tak obecné hodnocení rizik informační bezpečnosti. Zároveň hodláme představit jednu z možností napojení výstupů aplikace metodiky do prostředí normativních standardů pokrývající implementace systémů řízení informační bezpečnosti.

Klíčová slova: OSSTMM, audit, penetrační testy, informační bezpečnost, ISO.

1 Úvod

Open Source Security Testing Methodology Manual je veřejně dostupná metodika pro testování bezpečnosti (čili nejen provádění penetračních testů). Z dnešního pohledu je metodika „de-facto“ standard pro testování bezpečnosti. Metodika je šířena pod speciální licencí (pod svou vlastní) Open Methodology Licence, pod kterou je i publikována a je tedy veřejně přístupná.

V současné době je platná verze OSSTMM v.3, která přináší změny hlavně v návrhových opatřeních, které jsou více generické a tím je zjednodušen modulový průchod (více dále).

2 Metodika

Hlavním smyslem tohoto dokumentu je tedy poskytnout metodiku pro přesné a důkladné testování bezpečnosti za přesně a jasně definovaných podmínek. Metodika poskytuje natolik rozsáhlý pohled na bezpečnost, že se dá velmi jednoduše použít pro naprostou většinu auditů informačních systémů, penetračních testů, apod. Následování metodiky navíc zaručí testu tyto vlastnosti:

- Test bude důkladný,
- bude obsahovat všechny důležité aspekty,
- postup bude vyhovovat všem zákonům a občanským právům,
- výsledky budou měřitelné a porovnatelné s ostatními
- výsledky budou opakovatelné,
- výsledky budou obsahovat fakta získaná během samotných testů.

Jak bylo řečeno, manuál je veřejně přístupný a jako takový, nevyžaduje žádná speciální školení ani certifikáty, jako podmínku k tomu, aby byl použit.

Je navíc možné, certifikovat systém o splnění podmínek OSSTMM, který mohou získat ty organizace (resp. jejich části), které udržují čtvrtletně úroveň RAV (viz. dále) alespoň

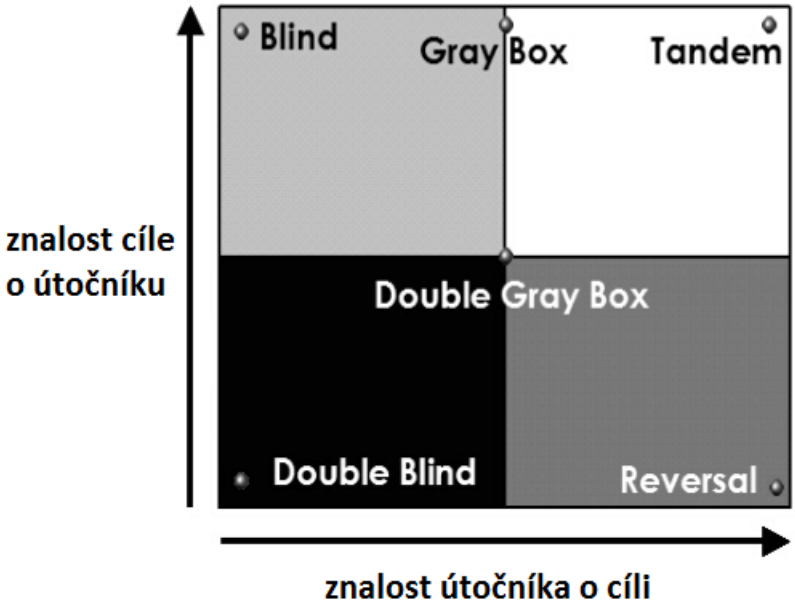
pod úrovní 95 % a nechají si provést, alespoň jednou ročně, ověření výsledků nezávislým subjektem.

2.1 Rozdělení bezpečnostních testů

OSSTMM definuje velké množství testů a přístupů k nim, z čehož vybraný typ je nutné uvést v závěrečné zprávě (jestliže se tak netane, je test považován za tzv. blind test).

Typy testů podle OSSTMM:

- **Blind**
  Testuje se cíl, o kterém nemá tester žádné informace. Tj. informace o postupech, samotných aktivech, implementovaných opatřeních, apod. Cíl testování o testování ví v předstihu a zná všechny jeho detaily. Jedná se primárně o test schopností testera, který jediný může svými schopnostmi a vědomostmi ovlivnit hloubku a rozsah testu.
- **Double blind**
  Někdy také black box. Je velmi podobný testu typu blind s tím rozdílem, že cíl testu neví o testu v předstihu a nezná žádné jeho detaily. Tento test je zaměřen opět na schopnosti testera, ale zároveň i na připravenost cíle vyrovnat se s útokem.
- **Gray box**
  Tester má limitované vědomosti o ochraně aktiv a aktivech cíle jako takových. Cíl testu je připravován v předstihu na tento test a zná všechny jeho detaily. Hloubka testu je závislá na schopnostech auditora a ve velké míře na kvalitě poskytnutých vědomostí.
- **Double gray box**
  Někdy též white box. Velmi podobný s předchozím, jen s tím rozdílem, že cíl nezná všechny detaily testu (typicky rozsah a čas).
- **Tandem**
  Někdy také crystal box. Tester i cíl s e připravují v předstihu a oba znají detaily testu, z čehož plyne, že jde o zaměření se na ochranu a monitoring cíle, s tím, že není možné testovat cíl na neznámé zranitelnosti.
- **Reversal**
  Tester má veškeré znalosti cíle. Cíl nemá žádné informace a neví ani, že test bude probíhat.



Obr. 1. Běžné podmínky testů (tzv. Common test methods) [1]

2.2 Použití metodiky pro penetrační test

Metodika jednoznačně definuje osnovu pro praktické použití. Osnova obsahuje pravidla, která je nutné dodržet pro správné použití metodiky. My nebudeme procházet oblasti zahrnující marketing, prodej, a kontaktní údaje, ale podíváme se čistě na povinnosti a kroky, které musí dodržet tester. Díky tomu je zaručena jak kvalita výstupů, tak jejich opakovatelnost a měřitelná kvalita samotných testů, což je v zásadě největší výhoda metodiky – zajištění opakovatelnosti a změřitelnosti výsledků, kde nezáleží na subjektu, který test provedl.

- **Definice hranice a rozsahu**
  - Rozsah testu musí být jasně definován před ověřováním zranitelností.
  - Musí být jasně vysvětleny limitace testu vzhledem k určenému rozsahu.
- **Plán testu**
  - Musí být vytvořen plán testu, který musí obsahovat začátek testu a předpokládané trvání testu.
- **Proces testování**
  - Tester musí respektovat a dodržovat bezpečnost, zdraví a soukromí veřejnosti a to jak v rozsahu testu, tak mimo něj.
  - Tester musí dodržovat zákony, které jsou platné v místě testu (myšleno ve fyzickém místě testu).
  - Cíl nesmí provádět žádné důležité nebo neobvyklé změny během testu.
  - Testeři by měli znát všechny použité nástroje, odkud tyto nástroje pocházejí, jak pracují, a měli by je mít otestované před samotným použitím v prostředí cíle.

- V případě objevení slabého místa během testu s velmi vysokou hrozbou, je nutné tento problém neprodleně nahlásit cíli spolu s praktickým doporučením pro řešení tohoto problému.
- Tester nesmí zanechat cíl s nižší aktuální úrovní zabezpečení, než jaká byla na začátku testu.
- Test na první pohled vysoce nestabilního a nezabezpečeného systému je zakázán, dokud nebude na místě implementována odpovídající bezpečnostní infrastruktura.

- **Zpráva**
  - Tester musí respektovat soukromí všech jedinců a udržovat jejich soukromí ve výsledcích.
  - Výsledky, které zahrnují osoby, v bezpečnosti neznalé, by měli být v anonymizované, popř. pouze statistické podobě.
  - Tester, ani analytik, jestliže taková role v řešitelském týmu existuje, by neměl podepisovat výsledky, kterých se přímo nezúčastnil.
  - Zprávy by měli zůstat objektivní, bez lží a osobního zabarvení.
  - Cíl musí být informován, jestliže tester změní testovací plán, místo, odkud je test prováděn, objeví vysoké riziko, před každým vysoce citlivým testem nebo testem s vysokým datovým tokem, popř. při jakémkoli problému vzniklém při testu.
  - Pokud jsou součástí zprávy doporučení, musí být tyto praktické a odůvodnitelné.
  - Zpráva musí jasně označit veškeré neznámé anomálie, uvést úspěšná i neúspěšná bezpečnostní měření.
  - Zpráva může obsahovat pouze kvantitativní metriky měření, které musí být založeny pouze na faktech a musí se vyvarovat jakékoli subjektivní interpretaci.

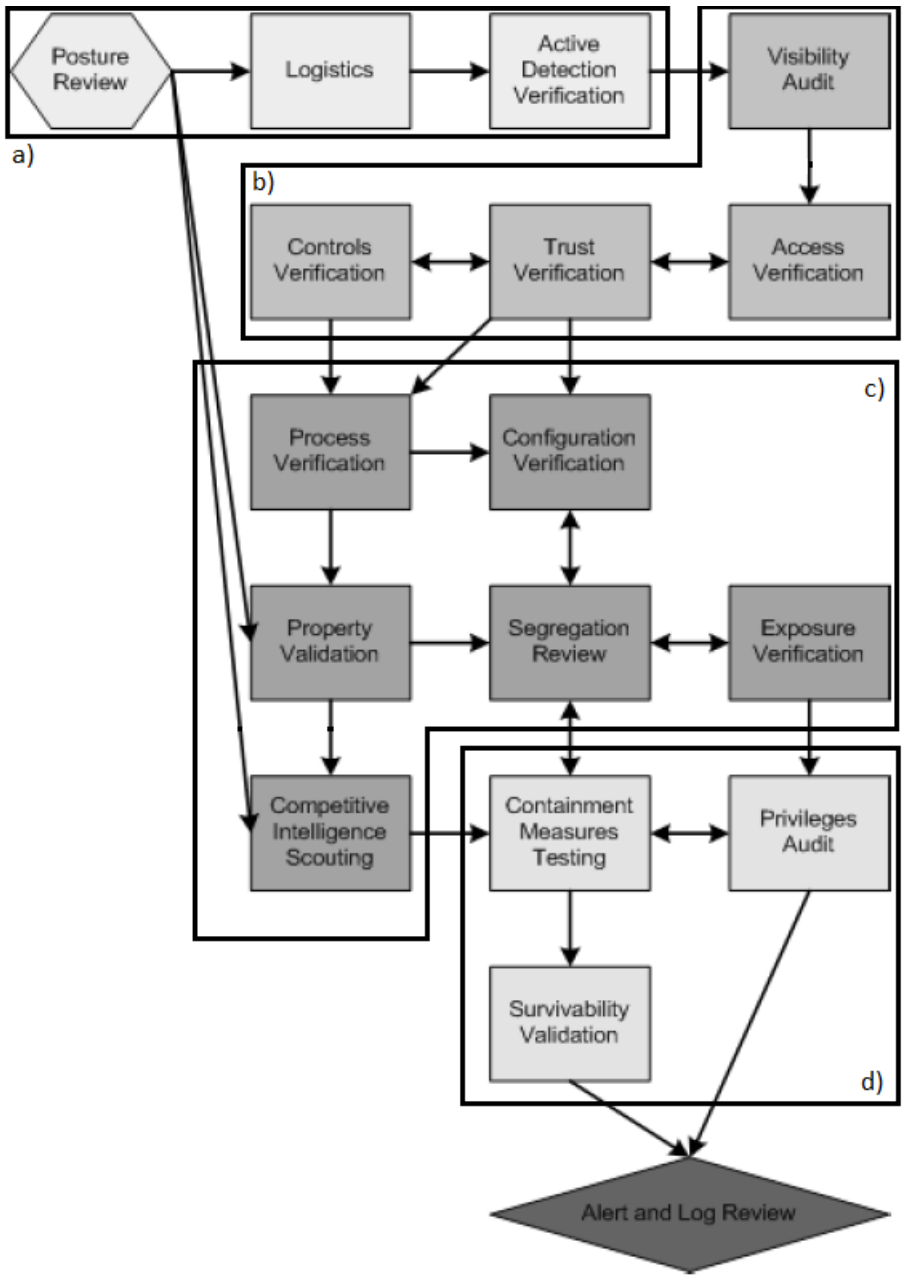
3 Aplikace metodiky

Metodika rozděluje potřebné úkoly, které musí být provedeny, do hierarchické struktury:

1. Kanál (CHANNEL)
2. Modul (MODULE)
3. Aktivita (TASK)

Spojením všech jednotlivých modulů pro všechny možné kanály získáme procesní mapy jednoduše zobrazující aplikaci samotné metodiky. Z pohledu procesní mapy je tato rozdělena na několik jednoduše odlišitelných fází (na obrázku č.2 odlišeny barevně) :

a) Úvodní fáze,
b) Fáze interakce,
c) Fáze vyšetřování,
d) Nápravná fáze.

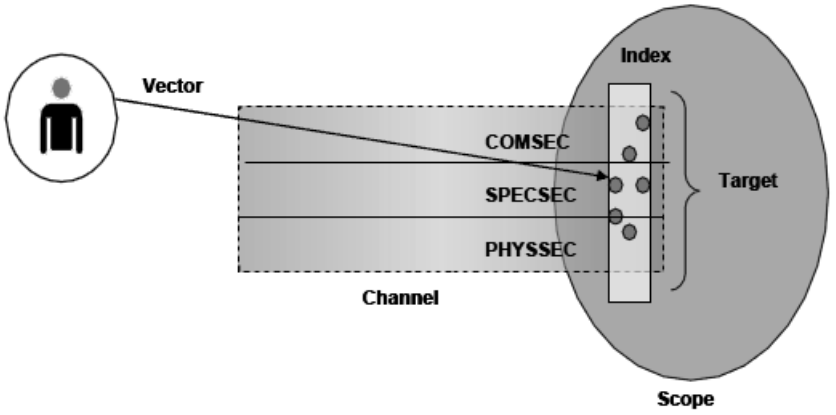


Obr. 2. Mapa bezpečnosti [1]

Pro aplikaci metodiky (a tedy metodiku samotnou) je definující, že metodika není lineární, jak je zřetelné z vazeb mezi jednotlivými moduly. Tato situace znamená, že samotný průchod (konkrétní cestu) mezi moduly si vybírá tester sám, a velmi často postupuje i zpětně, čímž aplikuje znalosti získané během aplikace „pozdějších“ modulů. Nejvýrazněji se tato vlastnost metodiky projevuje v kanálu PHYSSEC (viz dále). Podstatné při aplikaci metodiky je ovšem stále zaručená opakovatelnost testů a porovnání testů v čase.

Zásadní pro samotné testování a přístup k němu je tedy ustanovení a naplnění čtyř termínů, konkrétně:

- Scope – rozsah testů
- Channel – kanál, podle kterého bude testování probíhat (viz dále), který definuje, jaké moduly budou použity
- Index – počet cílů v rozsahu
- Vector – perspektiva, ze které tester přistupuje k testování Indexů



Obr. 3. Mapa bezpečnosti [3]

Jak je vidět na obrázku č. 3, obrovskou výhodou aplikace OSSTMM je faktická nemožnost uměle manipulovat s výsledky testů pouhým zvětšováním testovaného rozsahu (velikosti systému) – Index se i při zvětšování Scope nemění.

Podstatný je poté samotný kanál testování, který definuje moduly metodiky, které je možné a nutné aplikovat. Metodika určuje následující kanály, kde každý pokrývá odlišitelnou oblast informační bezpečnosti:

- **PHYSSEC** – personální bezpečnost a fyzická bezpečnost
- **SPECSEC** – bezpečnost elektronické komunikace
- **COMSEC** – bezpečnost datových sítí (bezpečnost provozu IT/IS) a bezpečnost telekomunikačních (analogových i digitálních) sítí

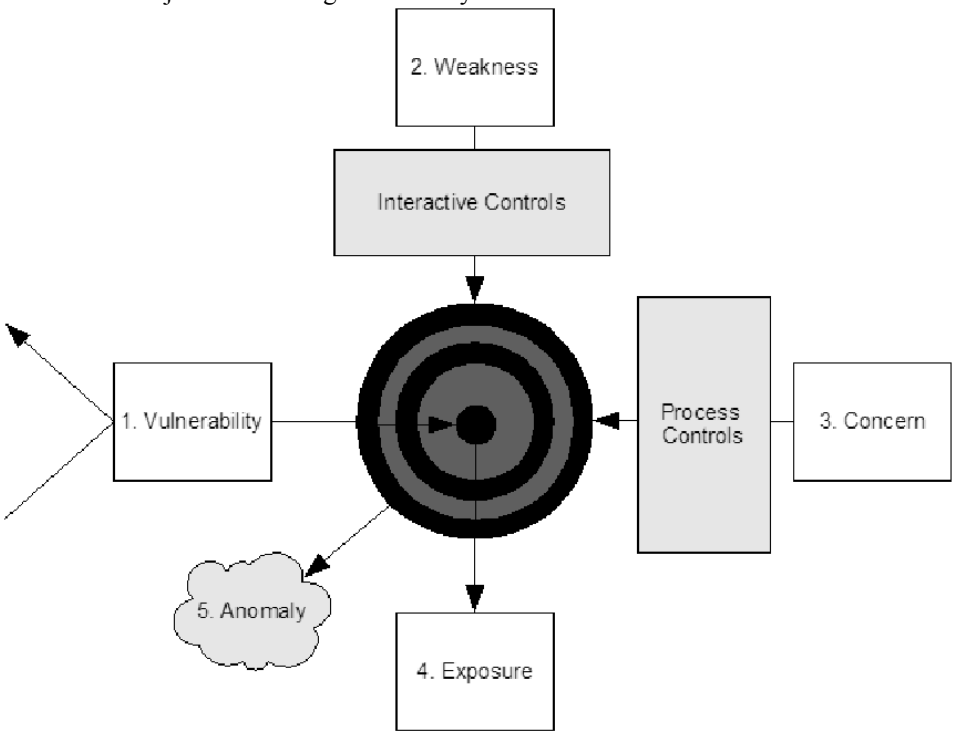
Tyto kanály jsou poté aplikovány na takzvané limity bezpečnosti, což je stav bezpečnosti vzhledem k známým, resp. zjištěným, nedostatkům uvnitř testovaného rozsahu – čili stav zjištěný testováním pokrývající testem zjištěné skutečnosti. Tyto nedostatky poté metodika rozděluje podle jejich dopadu na systém. Jedná se o následující kategorie nedostatků, resp. chyb:

- **Vulnerability** (zranitelnost) - znepřístupní aktiva autorizovaným subjektům, dovolí přístup neautorizovaným subjektům, dovolí neautorizovaným subjektům „skrýt“ aktiva, nebo sebe sama
- **Weakness** (slabina) – naruší/sníží efektivitu oblasti bezpečnosti kontrol
- **Concern** (účel nebo spíče zájem) – naruší/sníží/zneužije nebo zruší efektivitu oblasti bezpečnosti kontrol
- **Exposure** (vystavení) – poskytuje přímou/nepřímou viditelnost aktiv

- **Anomaly** (anomálie) – neidentifikovatelný/neznámý element

Základní vztahy je pak možné srovnat s normativním ISO přístupem.

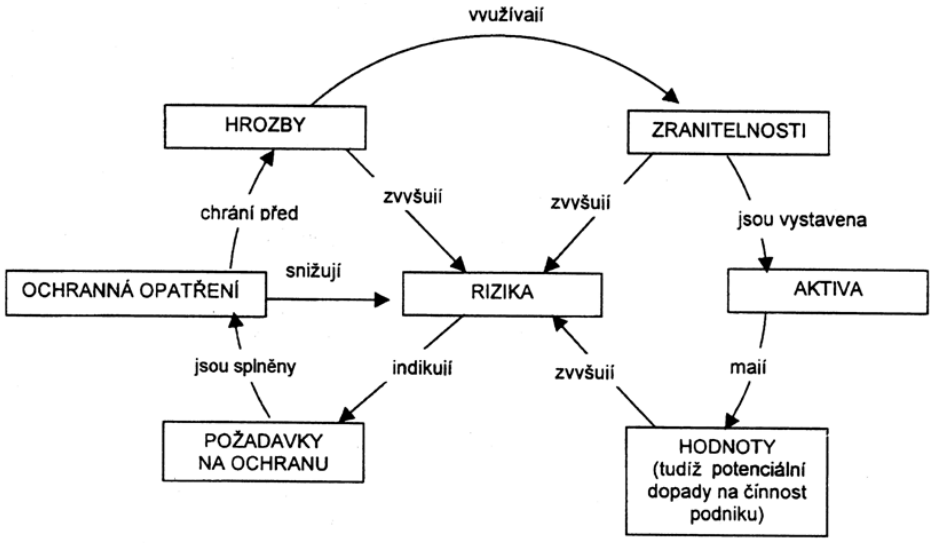
Na obrázku č. 4 je zobrazen přístup k identifikaci, resp. minimalizaci rizika z pohledu OSSTMM. Jak bylo výše definováno, klasifikujeme nedostatky (popř. chyby) do kategorií 1 až 5, kde tedy máme zranitelnost (vulnerability), která je kontrolována obecným přístupem k aktivu (resp. nedostatkem přístupu) – vzniká tam, kde dochází k interakci mezi nějakou hrozbou a aktivem; slabinu (weakness), která je minimalizována pouze interaktivními opatřeními; zájem (concern), jenž je definován množstvím a efektivitou toků nebo realizací kontrolních procesů a mechanismů; vystavení (exposure), jakožto samotnou viditelnost a dostupnost aktiva; a samozřejmě anomálii (anomaly) tedy chybu, nebo limitaci, která je fakticky neidentifikovatelná – jedná se tedy o cokoli, o čem nevíme, že existuje, neboli nad čím nemáme kontrolu. Je tedy jasně patrné, že identifikace jednotlivých hrozeb (nebo spíše typů hrozeb) probíhá separátně a i opatření pro minimalizaci rizika jsou vybírány a uplatňovány z jasně definované množiny příslušné k dané kategorii hrozby. Výsledek výpočtu RAV (Risk Assessment Value) je poté úhrnná hodnota působení jednotlivých kategorií nedostatků na aktivum, samozřejmě s uplatněním příslušných vybraných opatření. Je tedy možné, ale i vhodné, získat nejen celkovou hodnotu rizika, ale porovnávat a měřit i jednotlivé kategorie a složky tohoto rizika.



Obr. 4. OSSTMM [1]

ISO/IEC normy zabývající se informační bezpečností (systém managementu bezpečnosti informací tzv. ISMS - Information Security Management System) nabízejí de facto jednodušší model identifikace rizika a působení na aktivum, kdy jediný prvek, snižující riziko je jakékoli ochranné opatření (typicky z [4]) a hrozby jsou taktéž

klasifikovány a kvantifikovány stejném způsobem. Neexistuje tedy odlišný postup při nakládání se specifickými hrozbami nebo opatřeními. Proces výběru hrozby/opatření a vyhodnocování účinnosti opatření je vždy stejný. Na rozdíl od výpočtu RAV v OSSTMM tedy existuje pouze úhrnné (celkové) riziko vypočítávané některou z kvantitativních nebo kvalitativních metod [5] a míra specializace hrozeb a jejich rozložení (nebo alespoň představy o jejich rozložení) se ztrácí.



Obr. 5. ISO/IEC [2]

Právě soulad s jinými standardy (typicky ISO/IEC) není automatický – typicky z důvodu, v současné době, neexistence relevantní normy pro provádění technických auditů v oblasti informační bezpečnosti (to bude vycházet až z nově připravovaného standardu rodiny 27000, a to ISO/IEC TR 27008 Information technology - Security techniques - Guidance for auditors on ISMS controls, která by měla obsahovat doporučení pro auditory, kteří kontrolují implementovaná ISMS opatření vycházející z ISO/IEC 27002).

V současném stavu je tedy nutné vytvářet soulad expost. Nejvhodnějším přístupem se pak jeví právě porovnáním přístupu k hodnocení rizik, jak ukazují obrázky a 4 a 5.

4 Závěr

Příspěvek popisoval metodiku OSSTMM pro provádění testů a prověření informační bezpečnosti. Popsal způsoby použití a aplikace metodiky a rozdělení samotných bezpečnostních testů spolu s vysvětlením konkrétních rozdílností a cílového určení. Nakonec byl uveden jeden z možných aspektů napojení OSSTMM na systémy řízení informační bezpečnosti (ISO/IEC).

Literatura

1. [1] HERZOG P., Open Source Security Testing Methodology Manual 3.2, ISECOM, 2010

2. [2] ČSN ISO/IEC TR 13335-3. Informační technologie – Směrnice pro řízení informační bezpečnosti IT – Část 3: Techniky pro řízení bezpečnosti IT. Praha : Český normalizační institut, 1999.
3. [3] HERZOG P., Open Source Security Testing Methodology Manual 2.2, ISECOM, 2006
4. [4] ČSN ISO/IEC 27001:2006. Informační technologie – Bezpečnostní techniky – Systémy managementu bezpečnosti informací – Požadavky. Praha : Český normalizační institut, 2006.
5. [5] BARTOŠ, J. Analýza rizik. Sborník příspěvků Objekty 2010. Ostrava, 2010. s. 120-128. [2010-11-18]. ISBN 978-80-7368-899-8
6. ČSN EN ISO 19011:2002. Směrnice pro auditování systému managementu jakosti a/nebo systému environmentálního managementu. Praha : Český normalizační institut, 2002.
7. NIST Special Publication 800-115, Technical Guide to Information Security Testing and Assessment, National Institute of Standars, 2008

Annotation:

The goal of this paper is a description and introduction to OSSTMM methodics used for testing and auditing of information security, both for technical penetration tests and audits based on social engineering techniques and general information risk assessment. Furthermore, one of the possibilities of junction between results of methodics application and information security management system derived from an ISO/IEC standards will be introduced.

Návrh zabezpečeného monitorovacího systému v oblasti e-Health

Jan NAGY, Jiří SCHÄFER, Martin ZADINA, Petr HANÁČEK

Ústav inteligentních systémů, FIT VUT v Brně
Božetěchova 2, 61266 Brno
{inagy, schafer, izadina, hanacek}@fit.vutbr.cz

Abstrakt. Rostoucí demografická křivka a další faktory týkající se zdravotního stavu obyvatelstva předznamenávají rapidní nárůst služeb v oblasti zdravotnictví podporovaného elektronickými procesy a komunikačními technologiemi (e-Health). V tomto příspěvku se zamýšlíme nad konceptem architektury monitorovacího systému v prostředí e-Health. Zaměřujeme se zejména na komunikační a bezpečnostní stránku popisované architektury.

Klíčová slova: e-Health, monitorování pacientů, informační systém, bezdrátové technologie, zdravotnická informatika

1 Úvod

Monitorovací systémy jsou jedním z hlavních zdrojů dat v lékařském prostředí. Tyto systémy se typicky skládají ze senzorů, které jsou připojeny k pacientovi a komunikují se zařízením monitorujícím pacienta. Data z těchto senzorů mohou být uchovávána v databázích, které se připojí na informační systém zdravotnického zařízení, kde je s těmito daty manipulováno. Tyto systémy mohou být součástí infrastruktury zdravotnického zařízení. To umožňuje, aby data byla dostupná různým systémům uvnitř i vně nemocnice. Informační systémy zdravotnických zařízení pak mohou být propojeny za účelem výměny dat. Proto je nutné vytvořit mechanismus end-to-end zabezpečení k ochraně medicínských dat.

V poslední době jsou ve zdravotnictví stále více využívány bezdrátové technologie. V takovém případě komunikují senzory bezdrátově s monitorovacím zařízením umístěným u pacienta. Nebezpečí spočívá zejména v možnosti odposlechnutí nebo pozměnění (manipulace) komunikace. Proto musíme brát v úvahu bezpečnostní cíle jako důvěrnost, integrita, dostupnost dat. Je nutné se zaměřit na slabiny systému zpracovávajícího medicínská data, které by mohl využít útočník.

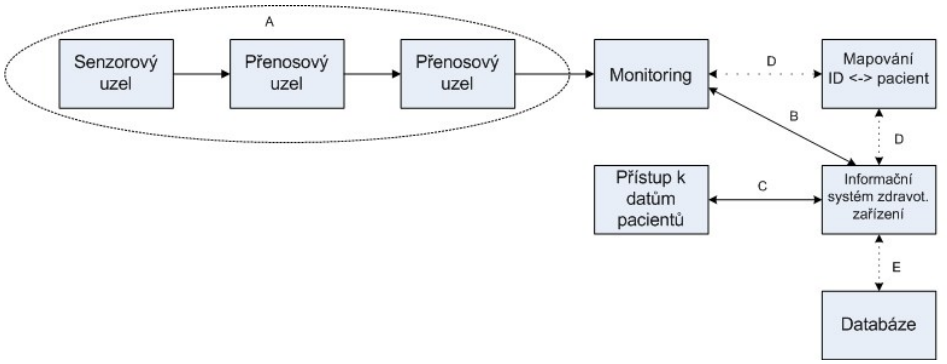
V minulosti byly představeny různé systémy pro monitorování zdravotního stavu pacientů, šlo zejména o monitorování dílčích životních funkcí bez napojení do stávající nemocniční infrastruktury. Komunikace v těchto systémech je založena na různých bezdrátových technologiích lišících se v operačním rozsahu nebo odolnosti proti rušení.

2 Návrh monitorovací platformy

Při návrhu platformy jsme vycházeli ze situace, kdy pacient leží na lůžku v nemocnici, jeho životní funkce jsou monitorovány pomocí senzorů, data o zdravotním stavu pacienta jsou

aktualizována s nemocniční databází a zdravotnický personál má možnost kdykoli k těmto datům přistoupit.

Navrhovaná platforma obsahuje několik logických částí (Obr. 1). Datové spoje mezi jednotlivými částmi systému (označeny písmeny A až E) mohou být realizovány různými komunikačními prostředky, od vnitřního/proprietárního rozhraní, přes komunikační standardy operující na malou (Bluetooth, NFC) či velkou vzdálenost (WiFi, GSM, Ethernet). Diagram jsme upravili podle [2].



Obr. 1. Návrh monitorovacího systému.

Na těle pacienta (nebo velmi blízko něj) jsou připevněny senzory, které posílají v pravidelných intervalech naměřené údaje o zdravotním stavu pacienta do jednotky „Monitoring". Informace z jednotlivých senzorů jsou doplněny identifikačním záznamem, který není nijak spojován s pacientem. Tento princip pseudonymity je nutný pro případ, že by došlo ke kompromitaci přenášených údajů.

„Monitoring“ je jednotka trvale umístěná u lůžka pacienta a je tedy neustále k dispozici pro příjem dat z různých senzorů. Z ní jsou data posílána do jednotky informačního systému (IS) zdravotního zařízení. Realizace konkrétního datového spoje mezi těmito bloky je závislá na umístění jednotky sbírající data od pacienta (uvažujeme mobilní i fixní jednotku a z toho vycházející bezdrátové/drátové přenosové technologie). Z informačního systému nemocnice jsou data ihned ukládána do databázového serveru. Odtud jsou data přístupná personálu nemocnice pomocí jednotky „Přístup k datům pacientů“. Jde o přenosné zařízení (například typu tablet), jehož prostřednictvím mají zdravotníci přístup k údajům o pacientech a jejich zdravotní dokumentaci. Taková komunikace musí být zabezpečená. Tablet musí být autentizovaný pro přístup k síťové infrastruktuře, stejně tak člověk, který jej využívá. K identifikaci jednotlivých záznamů, které jsou posílány do informačního systému zdravotnického zařízení, slouží část „mapování ID-pacient“.

2.1 Mobilní senzory

Ve výše navržené architektuře uvažujeme mobilní senzory (senzorové uzly). Tyto senzory jsou vybaveny bezdrátovým komunikačním rozhraním s asymetrickým šifrováním [1], aby byla zajištěna mj. jednoznačnost a nezaměnitelnost dat; jedná se o jeden z citlivých datových toků.

Při prvním použití je senzor spárován s monitorovacím zařízením. Synchronizace a spárování probíhá na bázi bezdrátové komunikace za pomocí asymetrického šifrování s použitím protokolu SSL nebo TLS s použitím certifikátů pro autentizaci, aby nedošlo

k záměrné, či nechtěné úpravě dat. V případě asymetrické šifry je použita komunikace pomocí certifikátů, a tedy i jednoznačný identifikátor jednotlivých senzorů.

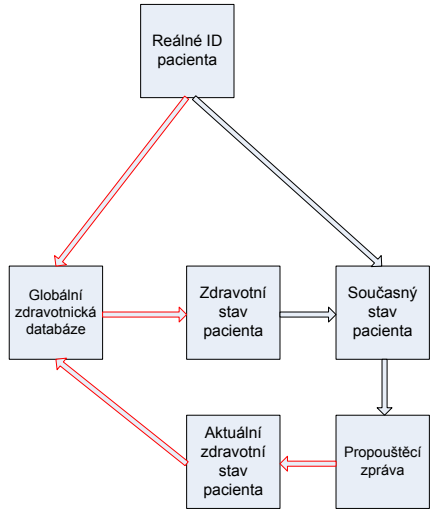
Po spárování začíná monitorovací zařízení od dotyčného senzoru přijímat data a případně je vyhodnocovat. Jakmile již senzor není potřeba, je spárování zrušeno a senzor je vrácen zpět do skupiny senzorů, ze které byl odebrán. Zde může být domácí stanice senzorů s možností dobíjet baterie těchto senzorů.

3 Bezpečnostní analýza systému

V celém systému je několik míst, která je nutno důsledně chránit. V první řadě se jedná o připojení modulů k monitorovacímu zařízení; bylo diskutováno v kap. 2.1.

Na následujícím obrázku (Obr. 2) je jedno z dalších citlivých míst – proces získávání nebo aktualizace záznamů v globálním zdravotním registru (nebo informačním systému nemocnice). K datům by měl mít přístup pouze ošetřující lékař na základě svolení pacienta anebo lékař záchranné služby. Každý lékař využívající tento systém musí vlastnit Osobní certifikát podepsaný Certifikační autoritou, bez tohoto certifikátu a dalšího faktoru autentizace (heslo nebo osobní OTP token s použitím osobního pinu) nebude moci získat z databáze jakékoliv informace, ani tyto informace aktualizovat.

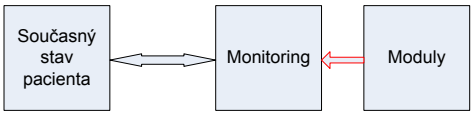
Podmínkou správné aktualizace nového zdravotního stavu je znalost správného identifikátoru nemocného. V případě, že se jedná o neznámého pacienta, či jedince bez záznamu v globálním zdravotním registru (příp. informačního systému nemocnice), není možné tento nový stav aktualizovat a vzniká tímto lokální anonymní pacient.



Obr. 2. Proces aktualizace záznamu v Globální zdravotnické databázi.

Při dlouhodobém pozorování není možné ukládat všechna nezpracovaná data ze senzorových modulů a z monitorovacího zařízení. Tato data musí být analyzována, redukována a relevantní údaje poslány v agregovaném stavu do zdravotnické databáze. V praxi to může vypadat tak, že při pravidelné návštěvě pacienta vyhodnotí ošetřující lékař

záznam z monitorovacího zařízení a zjistí relevanci dat a vytvoří z těchto dat zápis, který uloží do aktuálního zdravotního stavu pacienta (viz Obr. 3).



Obr. 3. Aktualizace stavu pacienta.

Protože údaje o zdravotním stavu jsou považovány za citlivé, musíme v našem systému definovat osoby odpovědné za nakládání s těmito údaji.

4 Závěr

Důležitou součástí nemocniční infrastruktury jsou monitorovací systémy. Ty mohou být využívány nejen při hospitalizaci, ale i před přijetím a vyšetřením pacienta mohou poskytnout cenné informace. V tomto článku jsme navrhli monitorovací architekturu, která může využít různé bezdrátové technologie přenosu. Popsali jsme možné datové toky v systému. Zaměřili jsme se zejména na bezpečnost, neboť s rostoucí penetrací těchto systémů bude význam bezpečnosti dat v kontextu zdravotnické informatiky stoupat.

Poděkování

Tato práce byla podporována FIT VUT v rámci grantu Pokročilé bezpečné, spolehlivé a adaptivní IT (FIT-S-11-1) a výzkumným záměrem Výzkum informačních technologií z hlediska bezpečnosti, MSM0021630528.

Literatura

1. Hanáček P., Nagy J., Pecho P.: Power Consumption of Hardware Cryptography Platform for Wireless Sensor. In: *Proc. of International Conference on Parallel and Distributed Computing*, Japan, IEEE CS (2009) 79-85 (anglicky)
2. Leister, W., Fretland, T., Balasingham, I.: Security and Authentication Architecture Using MPEG-21 for Wireless Patient Monitoring Systems. In: *Int.Journal on Advances in Security*, vol 2 no 1, 2009

Annotation:

Proposal of secured monitoring architecture in e-Health

Increasing demographic curve and other factors relating to the health of the population signals a rapid increase in health services supported by electronic processes and communication technologies, also known as e-Health. In this paper, we propose the architecture of the monitoring system which will use different wireless technologies. The result is a design and description of architecture, which will process the data on the health status of patients. We put emphasis on safety and anonymity of data during the proposal. Security of data transmission is achieved using asymmetric encryption.

Excalibur – nástroj pro data mining z výukových dat

Jaroslav BAYER[1], Hana BYDŽOVSKÁ[1], Jan GÉRYK[1], Lubomír POPELÍNSKÝ[2]

[1]*Centrum výpočetní techniky, FI MU*
Botanická 68a, 602 00 Brno
{bayer, bydzovska, geryk}@fi.muni.cz
[2] *Laboratoř dobývání znalostí, FI MU*
Botanická 68a, 602 00 Brno
popel@fi.muni.cz

Abstrakt. Popíšeme Excalibur, datový sklad a nástroj pro dobývání znalostí z výukových dat a projekty pomocí něj řešené.

Klíčová slova: data mining, datový sklad, Excalibur, výuková data.

1 Data mining z výukových dat

Informace obsažené v databázi Informačního systému Masarykovy univerzity (IS MU)[1] zahrnují kompletní data o studentech a jejich studiích, vyučovaných kurzech, vyučujících a managementu školy, např. známky studentů, zkoušky, hodnocení zkoušek a výuky, zápisy či registrace a informace o sociálních vztazích mezi uživateli, např. informace o odesílaných emailech nebo aktivitách v diskusních fórech. Návrh a struktura databáze IS MU odpovídá požadavkům transakčního zpracování dat. Pro účely analytického zpracování dat je tento model nevhodný. Proto jsme se rozhodli implementovat nový nástroj pro dobývání znalostí [2] spojený s datovým skladem – Excalibur, jehož databáze poskytuje efektivnější přístup k datům. Data jsou periodicky ukládána do systému jako snímky databáze IS MU v časových intervalech a doplněna o další agregované atributy, zejména percentily, sociální vazby uživatelů, počty přístupů k aplikacím apod. Co se týče nástrojů pro analýzu dat, postupně začleňujeme potřebné funkce z existujících nástrojů. Především využíváme metod statistického nástroje R[2] – pro předzpracování dat – a metod strojového učení implementovaných v systému Weka [3]. Pro výměnu vlastních modelů používáme jazyk PMML[3]. Excalibur je využíván v projektech Laboratoře dobývání znalostí Fakulty informatiky Masarykovy univerzity (FI MU), jejichž cílem je podpora rozhodování managementu FI MU. V následující části popíšeme jeden z projektů, zvyšování přesnosti klasifikace studijních dat analýzou sociální sítě. Z dalších zmiňme analýzu úspěšnosti studentů podle národnosti a pohlaví.

2 Zvyšování přesnosti klasifikace studijních dat analýzou sociální sítě

Cílem projektu bylo ověřit pozitivní vliv dat získaných ze sociální sítě na správnost detekce úspěšných studentů v průběhu studia. Ke klasifikaci jsme vybrali studia studentů FI MU,

[1] http://is.muni.cz
[2] http://www.r-project.org
[3] http://www.dmg.org

J. Zendulka, M. Rychlý (eds.), DATAKON 2011, Mikulov, 15.–18.10.2011, pp. 227–228.

kteří nastoupili v letech 2006-2008. Z databáze Excaliburu jsme extrahovali atributy charakterizující stav studia, např. známky, pohlaví, percentil, výsledky přijímacích Testů studijních předpokladů (TSP), počet souběžných studií, počet zapsaných kreditů v průběhu studia, počet opakovaných a uznaných předmětů apod. Nejlepší celková správnost byla dosažena pomocí rozhodovacích stromů, a to 82,5 % (při baseline, tj. klasifikaci do majoritní třídy 58,9 %). Poté jsme vytvořili sociální síť na základě vážených vztahů mezi jednotlivými uživateli. Z ní jsme za pomoci nástroje Pajek [1] spočetli její strukturální charakteristiky, zejména centralitu, stupně vrcholů, vážené součty vlastností sousedů a další. Velmi zajímavým zjištěním byl fakt, že nejlepších výsledků bylo dosaženo klasifikací pouze na atributech získaných analýzou sociálních dat, a to 97,7 % užitím metody líného učení IB1. Když jsme o tyto atributy obohatili původní sadu dat a aplikovali metody strojového učení, nejlepšího výsledku 93,6 % dosáhla metoda rozhodovacích pravidel PART.

3 Závěr

Naším konečným cílem je vytvoření komplexního škálovatelného nástroje pro dobývání znalostí z výukových dat s jednotným webovým rozhraním umožňujícím jednoduchý přístup k datovému skladu, k metodám dolování v datech, k jejich výsledkům a s možností vizualizace. Do budoucna uvažujeme o využití specializovaných nástrojů pro výběr rysů, jako je Feature Selection Toolbox[4], a především dalších vizualizačních nástrojů. Plánujeme také rozšířit funkcionalitu o analýzu textových dat, např. informace o jednotlivých předmětech a o závěrečných pracích.

Literatura

1. De Nooy, W., Mrvar, A., Batagelj, V.: *Exploratory social network analysis with Pajek*. Cambridge University Press, Cambridge, 2005.
2. Han, J., Kamber, M., Pei, J.: *Data Mining: Concepts and Techniques, 3rd Edition*. Morgan Kaufmann Publishers, USA, 2011.
3. Witten, H. I., Frank, E., Hall, M. A.: *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, Amsterdam, 2011.

Annotation:

Excalibur – a tool for data mining

The database of the Information System of Masaryk University contains all important data about students and their studies, courses, teachers or the management of the university. For analytical purposes, we have decided to implement a new web based tool for data mining (DM) and Machine learning (ML) connected to a data-warehouse fulfilled with educational data – Excalibur. This tool incorporates some standard DM and ML methods from the spread open-source tools. It is capable to visualise the data and the results. Excalibur has been employed in projects dealing with educational data several times. The most successful project considered with improving the classification of study-related data through social network analysis. The network was created from the weighted relations among students. The highest accuracy 97.7% was achieved using IB1 method.

[4] http://fst.utia.cz

Rejstřík autorů