ELEKTROTEHNIŠKI VESTNIK 81(3): 81-85, 2014
IZVIRNI ZNANSTVENI ČLANEK

Model za integracijo, migracijo in varnostno kopiranje aplikacij SaaS prek skupnega podatkovnega modela

Rok Povše, Matjaž B. Jurič

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Tržaška 25, 1000 Ljubljana, Slovenija
E-pošta: rok.povse@fri.uni-lj.si, matjaz.juric@fri.uni-lj.si

Povzetek. Prispevek naslavlja področje računalništva v oblaku oziroma podrobneje programsko opremo kot storitev (ang. Software as a Service - SaaS). Čeprav so tovrstne aplikacije v zadnjem času postale zelo razširjene, se pri njihovi uporabi srečamo z nekaterimi omejitvami. Uporabniki pogosto uporabljajo več kot eno aplikacijo SaaS in pri tem nimajo možnosti: integracije podatkov med njimi, zamenjave ponudnika zaradi t. i. problema Vendor lock-in ter izdelave in vzdrževanja varnostnih kopij zunaj podatkovnih centrov ponudnika aplikacije SaaS. Zato v prispevku predstavljamo model, s katerim želimo nasloviti omenjene probleme in zagotoviti enotno platformo za (i) pripravo varnostnih kopij, (ii) izvajanje migracije podatkov – zamenjava ponudnika ter (iii) izvajanje integracije – povezovanje podatkov med več aplikacijami SaaS. Izvedena je bila tudi implementacija modela, na podlagi katere smo predlagani model verificirali.

Ključne besede: računalništvo v oblaku, programska oprema kot storitev, SaaS, integracija

A model for the SaaS-application integration, migration and backup with a common data model approach

The paper addresses the field of cloud computing, more specifically the SaaS (Software-as-a-Service) layer of cloud computing. Although the usage of the SaaS applications has become very popular in recent years, their end-users are still facing certain limitations. In fact, several users leverage more than one SaaS application without any possibility of integrating heterogeneous data. Also, they are often bound to a specific application due to the "vendor lock-in" problem and they are not always given the option to maintain data backups outside the provider's data centers. Hence, we address these issues by introducing a model providing a unified platform for (i) preparing data backups, (ii) executing data migration and (iii) performing data integration among different SaaS applications. We evaluate the proposed model by performing a demonstrational implementation.

1 UVOD

Računalništvo v oblaku je čedalje bolj uveljavljena paradigma za postavitev in dostavo različnih računalniških virov. Tipično lahko ločimo naslednje ravni storitev: infrastrukturo kot storitev (ang. Infrastructure as a Service - IaaS), platformo kot storitev (ang. Platform as a Service - PaaS) in programsko opremo kot storitev (ang. Software as a Service - SaaS) [1]. Programska oprema postaja najbolj razširjen vir, ki je na voljo v obliki storitve. Razlog za to je veliko končnih uporabnikov in pa tudi storitev, ki so na voljo [2]. Prednosti pri uporabi aplikacij SaaS so številne, od fleksibilnosti in nižjih stroškov, do splošne dosegljivosti ne glede na lokacijo in napravo. Posledično vsakdo od nas že uporablja katero od storitev tipa SaaS, kot so na primer storitve za elektronsko pošto, shranjevanje podatkov, urejanje dokumentov ipd.

Kljub vsemu uporaba aplikacij SaaS še ni tako razširjena, kot bi bilo pričakovati. Razlogi za to so različni, pri čemer pa izstopa predvsem nezmožnost zamenjave ponudnika storitve (ang. Vendor lock-in) [3]. Zato v nadaljevanju predstavljamo model za integracijo, migracijo in varnostno kopiranje aplikacij SaaS, ki obravnava te probleme.

2 IZZIVI PRI INTEGRACIJI APLIKACIJ SAAS

Pri uporabi aplikacij SaaS se uporabniki srečajo s številnimi izzivi. Ti izzivi so tudi razlog, da uporaba aplikacij SaaS ni bolj razširjena. Trenutno še vedno ni na voljo enotnega in standardiziranega postopka, ki bi omogočal izvoz/uvoz podatkov iz različnih aplikacij SaaS, prav tako pa ni tehnologije, ki bi omogočala migracijo in sprotno integracijo podatkov med aplikacijami. Zaradi nezmožnosti integracije se zmanjšata fleksibilnost in cenovna ugodnost aplikacij SaaS, saj uporabniki praviloma potrebujejo dodatne rešitve, ki integrirajo podatke med različnimi aplikacijami SaaS. Te dodatne rešitve za integracijo mora danes namensko razviti vsak uporabnik aplikacij SaaS, kar je zahtevno tako s časovnega kot s finančnega vidika [4].

Poleg tega ponudniki aplikacij SaaS pogosto ne podpirajo zamenjave ponudnika storitve (ang. Vendor lock-in) in ni pričakovati, da bi se v prihodnosti to spremenilo. Posledično je prenehanje uporabe določene aplikacije SaaS oz. zamenjava ponudnika aplikacije oz. storitve težavna, draga in povezana s številnimi zapleti.

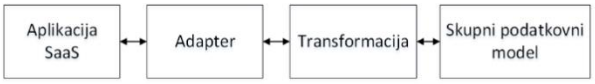
Prejet 16. december, 2013
Odobren 9. maj, 2014

Zaradi potrebnega dodatnega truda in sredstev se uporabniki teže in redkeje odločajo za prehod h konkurenčni aplikaciji, čeprav je ta ugodnejša in ponuja več funkcionalnosti.

Uporabniki izkazujejo tudi potrebo po vzdrževanju varnostnih kopij podatkov aplikacij SaaS na kakšni drugi lokaciji, nad katero imajo bolj neposreden nadzor (npr. lokalno na svojem strežniku ali pri drugem zaupanja vrednem ponudniku). Obstoječe aplikacije SaaS takšnega varnostnega shranjevanja praviloma ne omogočajo, ali pa je le-to zapleteno in drago za uporabo.

3 MODEL INTEGRACIJE NA PODLAGI SKUPNEGA PODATKOVNEGA MODELA

Model integracije aplikacij SaaS, ki vključuje izvajanje integracije in migracije podatkov med aplikacijami, temelji na modelu, ki ga v posplošeni obliki prikazuje slika 1.



Slika 1: Posplošen model integracije aplikacij

Komunikacija med aplikacijo SaaS in preostalimi komponentami modela poteka prek adapterja. Podatke, ki jih pridobi adapter, nato s pomočjo transformacij preslikamo v skupni podatkovni model, ki predstavlja podatkovno strukturo, skupno vsem aplikacijam in je tako podlaga za izvajanje migracije in integracije. Poglejmo še nekoliko podrobneje posamezne komponente modela.

3.1 Adapterji

Aplikacije SaaS izpostavljajo namensko razvite aplikacijske vmesnike. Določene aplikacije izpostavljajo spletne storitve SOAP, določene storitve REST, nekatere pa temeljijo na vmesnikih, za katere je treba uporabiti odjemalce v obliki posebnih programskih knjižnic. Heterogeni vmesniki za dostop so v tem primeru ovira, saj onemogočajo, da bi na enoten način naslovili pretvorbo iz podatkovnega modela, specifičnega za aplikacijo SaaS, v skupni podatkovni model. Pretvorba v skupni podatkovni model bi v tem primeru bila sklopljena z modulom za dostop do aplikacije SaaS, kar pa ni zaželeno. Komunikacija med aplikacijo SaaS in preostalimi komponentami tako vedno poteka prek komponente adapterjev, ki preostalim komponentam zagotavljajo enoten način predstavitve podatkovnega modela. To je tudi podlaga za implementacijo transformacij na enoten način. Implementacija adapterja je torej edina komponenta, ki pozna specifike vmesnikov uporabljene ciljne aplikacije SaaS.

3.1.1 Metapodatki adapterjev

Vsak adapter poleg implementacije pridobivanja in nalaganja podatkov vključuje tudi metapodatke, prek katerih definira konfiguracijo, potrebno za njegovo izvajanje. Prvi del konfiguracije se nanaša na definicijo vhodnih in izhodnih podatkov, na podlagi katerih je mogoče implementirati eno ali več transformacij. Definicija vhodnega in izhodnega dela podatkov je realizirana z dokumentom tipa XML Shema [5]. Drugi del konfiguracije se nanaša na konkretno izvajanje adapterja. Vsaka aplikacija SaaS ali druga ciljna storitev, s katero komunicira adapter, zahteva določene uporabniške podatke pri dostopu do njihovega aplikacijskega vmesnika. Konkretni podatki v tem primeru so na primer uporabniško ime, ključ za dostop do računa, tipično pa je zahtevan še začasni žeton. Če gre za ciljno storitev, kot je izdelava varnostne kopije, izvajanje adapterja zahteva na primer URL-naslov strežnika FTPS ter podatke za avtentikacijo.

3.1.2 Tipi adapterjev

Model predvideva več tipov adapterjev glede na njihov namen uporabe: adapterji za migracijo, adapterji za integracijo in adapterji za varnostno kopiranje. Število adapterjev za aplikacijo ni omejeno. Posledično lahko obstaja tudi več adapterjev istega tipa za isto aplikacijo, izbira konkretnega adapterja za ciljno aplikacijo pa je prepuščena končnemu odjemalcu. Vsi adapterji predvidevajo obojestransko komunikacijo, torej ne zgolj branja, temveč tudi nalaganje podatkov. Zmožnost realizacije obojestranske komunikacije pa je seveda odvisna od funkcionalnosti API-jev ciljne aplikacije SaaS. V okviru modela je predviden še adapter za spremljanje storitev, ki pa ne predvideva preslikave v skupni podatkovni model.

3.2 Skupni podatkovni model in transformacijski mehanizmi

Skupni podatkovni model je ključna komponenta predlaganega modela, ki definira strukturo skupnih podatkov, od katere so nato odvisne posamezne transformacije. Slika 2 prikazuje podrobnejšo zasnovo skupnega podatkovnega modela in odvisnosti med posameznimi komponentami. Skupni podatkovni model tako sestavlja nabor posameznih podatkovnih modelov. Posamezne podatkovne modele lahko nato realiziramo s shemami XML [5], ki predstavljajo tehnološko nespecifičen način za opis podatkovne strukture dokumenta, zasnovanega na XML.

3.2.1 Odvisnost med podatkovnimi modeli

Modeli so lahko medsebojno odvisni, lahko pa tudi dedujejo definicije. Na sliki 2 je primer dedovanja prikazan med modelom 1 in modelom 5 ter med modelom 1 in modelom 2. Modela 2 in 5 tako razširjata model 1 z dodatnimi definicijami podatkovne sheme. Podatkovni modeli so hkrati lahko odvisni tudi od preostalih modelov. Ponovno uporabne podatkovne

strukture lahko tako uporabi več modelov. Kot primer navedimo podatkovni model za »naslov«, ki se lahko pojavlja tako v podatkovnem modelu za zapis podrobnosti kupca, kot tudi pri podatkovnem modelu za zapis o naslovu za vročanje mesečnih računov. Na sliki 2 je primer odvisnosti prikazan na primer med modelom 2 ter modeloma 3 in 4.

3.3 Definicija transformacij

Adapter lahko iz posamezne storitve pridobi podatke, ki temeljijo na več različnih podatkovnih modelih, in jih nato preslika v skupni podatkovni model. Model transformacijskih mehanizmov predvideva dvonivojsko definicijo transformacij. Prvi nivo, kot je razvidno iz slike 2, predvideva eno ali več definiranih transformacij za vsak posamezni adapter. Tako se omogoči, da lahko uporabniki skupnosti prispevajo različne transformacije za isti adapter ter isto ciljno strukturo podatkov. Poleg tega si lahko uporabniki ustvarijo lastne zasebne transformacije, ki jih prilagodijo svojim potrebam. Na tej ravni je definirana tudi odvisnost med adapterjem in transformacijo, in sicer transformacija definira, s katerim ciljnim adapterjem (in posledično podatkovno shemo in ciljno aplikacijo) jo je mogoče uporabiti. Katera transformacija se uporabi med izvajanjem migracije, integracije ali varnostnega kopiranja, izbere končni uporabnik.

Povezavo s skupnim podatkovnim modelom zagotavlja druga raven transformacij. Le-ta definira nabor podtransformacij, ki so na sliki za primer transformacije S1ANT2 označene s S1ANT2P1, S1ANT2P2 in S1ANT2P3. Vsaka podtransformacija definira, kako se posamezni tip podatkov, ki ga generira adapter, preslika v konkretni podatkovni model skupnega podatkovnega modela. Prednosti takšnega dvonivojskega pristopa so predvsem na strani končnega uporabnika, ki dopolnjuje in uporablja transformacije. Uporabnik lahko pri uporabi transformacije označi, naj se uporabi zgolj omejen nabor podtransformacij. Poleg tega spreminjanje konkretnih podtransformacij ne vpliva na preostale podtransformacije, ki so del celotne transformacije.

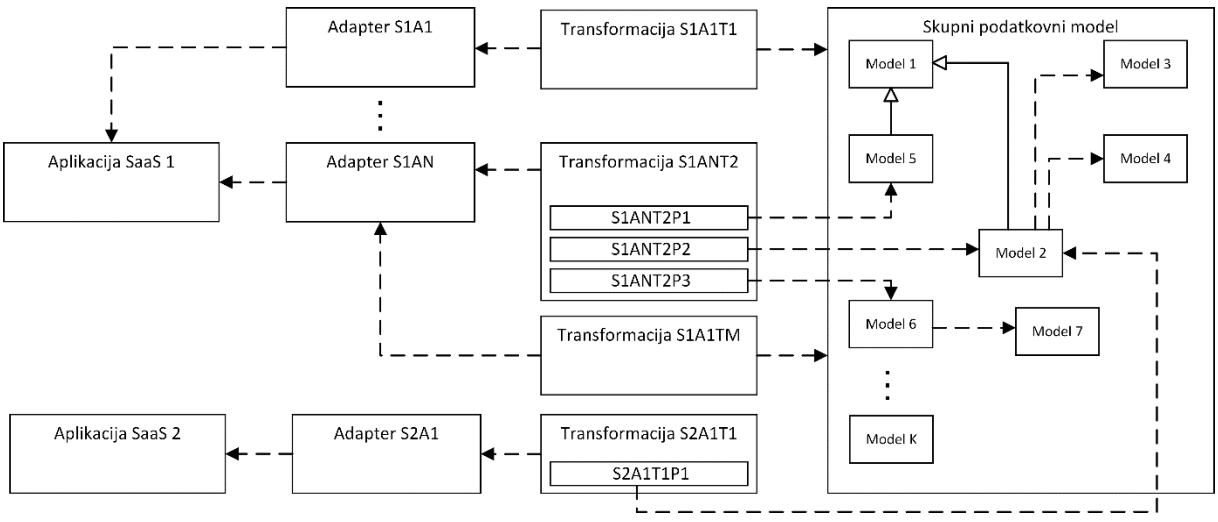
Transformacije oziroma bolje rečeno posamezne podtransformacije so zasnovane na tehnološko nespecifičnih jezikih EXtensible Stylesheet Language (XSLT) [6] in XQuery [7]. Obe tehniki sta način za implementacijo transformacijskih mehanizmov med podatki, skladnimi s shemo XML.

3.3.1 Avtomatsko generiranje transformacij

Samodejno generiranje preslikav med podatkovnimi modeli je že uveljavljeno raziskovalno področje [8], v okviru katerega so nastala številna orodja. Uporaba teh orodij je podprta tudi v modelu, ki ga predlagamo, in sicer v obliki podpore avtomatskega generiranja transformacij med podatkovno strukturo adapterja in podatkovno strukturo skupnega podatkovnega modela. Uporaba orodij za samodejno generiranje preslikav se uporabi predvsem pri ustvarjanju prve transformacije adapterja in nam s tem omogoči, da laže definiramo preslikavo med novim adapterjem in skupnim podatkovnim modelom.

3.4 Kompatibilnost aplikacij

Zasnova skupnega podatkovnega modela in transformacijskih mehanizmov omogoča, da lahko na podrobnejši model odvisnosti in zasnove skupnega podatkovnega modela (slika 2) gledamo tudi kot na graf. Za vsako aplikacijo SaaS je lahko definiran adapter. Za vsak adapter je lahko definiranih več transformacij, ki vsebujejo po vsaj eno pod-transformacijo, le-ta pa je odvisna od konkretnega modela skupnega podatkovnega modela. Omenjeno pot v grafu lahko opišemo tudi iz nasprotne smeri, torej konkretni model skupnega podatkovnega modela je lahko odvisen od vsaj ene podtransformacije, ki vedno pripada določeni transformaciji. Transformacija je vedno odvisna od določenega adapterja, le-ta pa od



Slika 2: Zasnova skupnega podatkovnega modela in odvisnosti komponent

konkretne aplikacije SaaS. Tako lahko hitro ugotovimo, da omenjeni model integracije omogoča tudi identifikacijo kompatibilnih aplikacij prek skupnega podatkovnega modela. Konkretni modeli skupnega podatkovnega modela tako predstavljajo presek, na podlagi katerega je mogoče identificirati in končnemu uporabniku predlagati tiste aplikacije SaaS, nad katerimi je mogoče izvajati integracijo ali migracijo. Tako ni potrebno, da vzdržujemo kakršne koli metapodatke o kompatibilnosti aplikacij SaaS. Na sliki 2 lahko identificiramo, da je kompatibilnost med aplikacijama SaaS 1 in SaaS 2 omejena na model 2 ter model 1. Obe aplikaciji namreč definirata adapterja (S1AN, S2A1), ki sta odvisna od dveh transformacij (S1ANT2, S2A1T1). Omenjeni transformaciji vključujeta podtransformaciji (S1ANT2P2, S2A1T1P1) s skupno odvisnostjo od modela 2. Odvisnost med aplikacijama je razvidna tudi prek podtransformacij S1ANT2P1 in S2A1T1P1, ki sta posredno odvisni od modela 1.

3.4.1 *Dopolnjevanje skupnega podatkovnega modela*

Končnim uporabnikom je omogočeno dopolnjevanje skupnega podatkovnega modela. Pri tem imajo možnost, da razvijejo nov podatkovni model, če jim noben od obstoječih podatkovnih modelov ne ustreza. Boljši pristop je, da uporabijo obstoječi podatkovni model in ga razširijo. Takšen primer je tudi prikazan na sliki 2, kjer modela 5 in 2 razširjata model 1. To je tudi priporočen način definiranja novih podatkovnih modelov, saj poleg tega, da poenostavi definicijo novega podatkovnega modela, ne omejuje kompatibilnosti aplikacij. Nov podatkovni model namreč definira, kateri obstoječi podatkovni model razširja, zato se tudi ohrani relacija med novo transformacijo in obstoječim podatkovnim modelom.

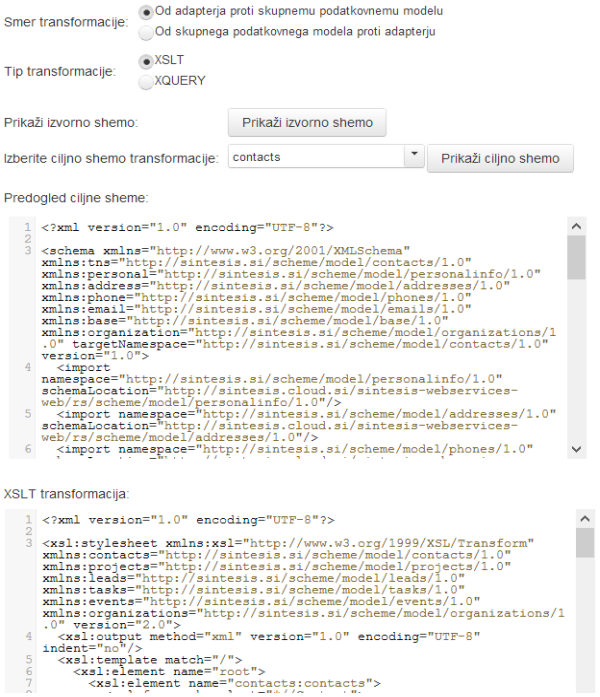
3.5 *Podatkovna integracija*

Izvedba migracije in varnostnega kopiranja je glede na predstavljeni model za končnega uporabnika relativno preprosta. Uporabnik najprej izbere izvorno aplikacijo SaaS, nato pa izbere še želen adapter in transformacijo. Če gre za varnostno kopiranje, mora izbrati še ciljno storitev, kamor bo podatke izvozil. V primeru migracije se na enak način kot za izvorno aplikacijo SaaS določi še ciljno aplikacijo SaaS.

Podatkovna integracija je nekoliko kompleksnejša, saj model vključuje še mehanizme za upravljanje matičnih podatkov (ang. Master Data Management - MDM). MDM je področje, ki celovito obravnava podatkovno integracijo in izolira matične podatke v centralni repozitorij [9]. Pri migraciji in varnostnem kopiranju zgolj enkratno dodajamo podatke in ni potrebe po vzdrževanju relacij med heterogenimi podatkovnimi viri, medtem ko moramo pri integraciji ohranjati relacije med povezanimi podatki različnih podatkovnih virov. Zato skupni podatkovni model vključuje tudi komponento navideznega registra, ki je namenjena vzdrževanju minimalnega nabora enoličnih identifikatorjev.

4 IMPLEMENTACIJA MODELA

Implementacijo omenjene platforme smo izvedli v okviru projekta Sintesis, ki je projekt strukturnih skladov in ga je financiralo Ministrstvo za izobraževanje, znanost in šport. Omenjena rešitev je dostopna na URL-naslovu http://sintesis.cloud.si in je na voljo vsem uporabnikom. Uporabnikom so na voljo tudi mobilni odjemalci za platforme Android, Windows Phone, iOS in BlackBerry 10.



Slika 3: Uporabnikovo dopolnjevanje transformacije

Implementacijo sistema smo izvedli s tehnologijami Jave EE. Adapterji so bili implementirani v obliki javanskih knjižnic, ki jih je mogoče dinamično nalagati iz podatkovne baze. Okolje, ki izvaja implementirane adapterje in transformacije, je implementirano z javanskimi strežniškimi zrni (ang. Enterprise JavaBeans - EJB), s čimer smo zagotovili zanesljivost in skalabilnost dane rešitve. Izvedba posameznih zahtevkov migracije, integracije ali varnostnega kopiranja lahko traja dlje časa, zato se ti izvajajo asinhrono. Takšen način izvajanja smo zagotovili z uporabo ločenega sporočilnega sistema (ang. Java Message Service - JMS). Posamezni zahtevki se najprej dodajo v izvajalno vrsto, ki je implementirana s tehnologijo JMS. Za prevzemanje zahtevkov iz vrste so nato odgovorna sporočilna zrna (ang. Message-Driven Bean - MDB). Le-ta predajo zahtevek strežniškim javanskim zrnom EJB, ki zahtevek izvedejo, rezultat izvedbe pa posredujejo klicatelju prek dodatnih vrst JMS.

Platforma je zasnovana kot razširljiva platforma, kjer lahko uporabniki dodajajo ne zgolj storitve SaaS, temveč naložijo tudi lastne adapterje, transformacije ali pa prilagodijo skupni podatkovni model lastnemu pogledu na podatke. Razvoj lastnih adapterjev poteka tako, da si uporabniki s portala prenesejo vzorčne projekte za okolje Eclipse, kjer adapter razvijejo in ga naložijo na portal v obliki knjižnice .jar. Dopolnjevanje skupnega podatkovnega modela in definicija transformacij potekata neposredno na portalu. Slika 3 prikazuje konkreten primer urejanja obstoječe transformacije. Uporabnik lahko določi smer izvajanja konkretne podtransformacije – v smeri skupnega podatkovnega modela ali v smeri proti adapterju. V nadaljevanju uporabnik lahko določi še način, kako se transformacija izvede (XSLT ali XQuery), ter prikaže podrobnosti izvorne ali ciljne podatkovne sheme, za katero definira transformacijo.

5 SKLEP

V prispevku smo predstavili pristop k migraciji, integraciji in varnostnemu kopiranju aplikacij SaaS, ki poteka na podlagi skupnega podatkovnega modela. Le-ta je predstavljen v obliki podatkovnih shem, ki predstavljajo enoten podatkovni model matičnih podatkov, kar tudi zagotovi prenosljivost podatkov med različnimi aplikacijami SaaS, tudi če so le-te različnega tipa.

Zasnova modela omogoča tudi, da lahko zgolj na podlagi relacij med posameznimi komponentami modela identificiramo kompatibilne aplikacije.

Implementacija modela ne omogoča končnim uporabnikom zgolj izvajanja varnostnih kopij, migracij in integracij, temveč tudi, da si prilagodijo način delovanja platforme. Uporabniki lahko dopolnijo skupni podatkovni model, razvijejo lastne transformacije in adapterje. Predlagani model bi lahko v prihodnosti razširili še z mehanizmi za samodejno preverjanje kakovosti razvitih komponent (adapterjev transformacij in podatkovnih modelov). Tako bi izboljšali njihovo kakovost ter poenostavili njihov razvoj.

Uporabnost implementacije modela je odvisna predvsem od dovršenosti programskih aplikacijskih vmesnikov končnih ponudnikov aplikacije SaaS. Ker so aplikacije SaaS čedalje bolj popularne in zaradi potreb po medsebojnem povezovanju aplikacij lahko upravičeno pričakujemo, da bo podpora za povezovanje aplikacij postala eden ključnih dejavnikov, ki jih bodo uporabniki upoštevali pri odločitvi o izbiri aplikacije SaaS. Tako lahko pričakujemo, da je cilj vseh ponudnikov, da v prihodnosti ponudijo čim bolj dovršene programske vmesnike za podatkovno integracijo z drugimi ponudniki aplikacij SaaS.

LITERATURA

[1] R. Dukaric, M. B. Juric, "Towards a unified taxonomy and architecture of cloud frameworks," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1196–1210, 2013.

[2] L. Columbus, "Cloud Computing and Enterprise Software Forecast Update, 2012," Forbes, 2012. [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2012/11/08/cloud-computing-and-enterprise-software-forecast-update-2012/.

[3] M. P. Papazoglou, W.-J. van den Heuvel, "Blueprinting the Cloud," *IEEE Internet Computing*, vol. 15, no. 6, pp. 74–79, 2011.

[4] H. Hai, S. Sakoda, "SaaS and integration best practices," *Fujitsu scientific & technical journal*, vol. 45, no. 3, pp. 257–264, 2009.

[5] W3C, "XML Schema," 2001. [Online]. Available: http://www.w3.org/XML/Schema.

[6] W3C, "XSL Transformations (XSLT) Version 3.0," 2012. [Online]. Available: http://www.w3.org/TR/xslt-30/.

[7] W3C, "XQuery 1.0: An XML Query Language," 2010. [Online]. Available: http://www.w3.org/TR/xquery/.

[8] E. Rahm, P. A. Bernstein, "A survey of approaches to automatic schema matching," *The VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001.

[9] M. Križevnik, M. B. Jurič, "Upravljanje matičnih podatkov kot osnova pri vpeljavi storitveno usmerjene arhitekture," *Uporabna informatika*, vol. 19, no. 1, pp. 5–14, 2011.

Rok Povše je doktorski študent in asistent v Laboratoriju za integracijo informacijskih sistemov na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Sodeluje pri raziskovalnih in aplikativnih projektih ter pri projektih v povezavi z gospodarstvom.

Matjaž B. Jurič je redni profesor na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer vodi Laboratorij za integracijo informacijskih sistemov. Je avtor oziroma soavtor številnih knjig in člankov s področja storitveno usmerjenih arhitektur, računalništva v oblaku in Jave. Poleg tega ima tudi nazive Java Champion, Oracle ACE Director in IBM Champion.