Posrednik sporočil v vlogi pregledovalnika sporočil Solr pri logističnem procesu

Daniel K. Rudolf, Alen Vincelj, Nina Novinec
ACTUAL Informacijske tehnologije, d. d., Ferrarska ulica 14, 6000 Koper
daniel.kovacevicrudolf@actual-it.si; alen.vincelj@actual-it.si; nina.novinec@actual-it.si

Izvleček

Pri razvoju informacijskih sistemov na področju logistike pristaniških dejavnosti se podjetje Actual I. T. srečuje s potrebo špediterjev, da imajo vpogled v stanje svojih posredovanih sporočil (npr. delovna naročila) v sistem Luke Koper. Zato želi podjetje ugotoviti, katere tehnologije so najbolj ustrezne za posredovanje in pregledovanje posredovanih sporočil. Bistveni namen raziskovalnega prispevka je pregledati in opisati obstoječe posrednike sporočil ter obravnavati sestavne dele posrednika sporočil in način, kako so aplikacije povezane med seboj. Cilj prispevka je ugotoviti, katera orodja za posredovanje in pregledovanje sporočil so najbolj primerna za prikaz stanja računalniških sporočil. Rezultati raziskave so pokazali, da je za izdelavo pregledovalnika računalniških sporočil (elektronska izmenjava podatkov, EDI) najprimernejša rešitev v obliki spletne strani, zgrajena z arhitekturnim vzorcem MVC, ki uporablja tehnologijo Solr za iskalnik.
Ključne besede: posrednik sporočil, pregledovalnik sporočil, Solr, EDI.

Abstract

Message Broker in Solr Message Viewer Role in the Logistics Process

In the IT systems development of the port area logistics operations, the Actual IT company is faced with the forwarding agents' need for insight into the status of their transmitted messages (e.g. working orders) in the Luka Koper system. Therefore, the company wants to determine which technologies are most appropriate for the transmission and viewing of transmitted messages. Therefore, the main purpose of this research paper is to review and describe the existing message brokers and address the components of a message broker and the way applications are related to each other. The aim of this paper is to determine which tools for message mediation and reviewing are best suited for displaying the status of computer messages. The results show that the most suitable solution for developing an electronic message viewer (Electronic Data Interchange - EDI) is in the form of a website built with the MVC architectural pattern and using Solr search technology.
Key words: Message broker, Message viewer, Solr, EDI.

1 UVOD

Logistični proces v svojem izvajanju zahteva med drugim tudi učinkovit pretok informacij od točke izvora pa vse do točke, ko pride neko blago do točke potrošnje. V ta namen se je uveljavila elektronska izmenjava podatkov (EDI), ki zagotavlja vsa sredstva za doseganje potrebnega usklajevanja pri podpori logističnega procesa. Informacijsko-komunikacijska tehnologija je pomembna pri elektronski izmenjavi podatkov, saj ponuja orodja, ki omogočajo uresničitev takšne vrste izmenjave podatkov. Posrednik sporočil je eno izmed orodij, ki v logističnem procesu omogoča izmenjavo elektronskih sporočil med različnimi aplikacijami. Elektronska izmenjava podatkov je bila definirana že v devetdesetih letih prejšnjega stoletja za izmenjavo podatkov med računalniškimi sistemi in poslovnimi partnerji (Corbitt, 1992). Na splošno je opredeljena kot »medorganizacijska *računalnik v računalnik* izmenjava poslovnih dokumentov v standardiziranih formatih« (Clifton, 1989). Razširitev te opredelitve vključuje brezpapirno ali elektronsko trgovanje (Kimberley, 1991).

Elektronska izmenjava poslovnih podatkov iz enega računalnika v drugega poenostavlja sodelovanje organizacije s strankami, distributerji, dobavitelji, prevozniki in ponudniki storitev (Smalheiser, 1990). Elektronska izmenjava podatkov s svojo

sposobnostjo posredovanja sporočil povečuje lastno učinkovitost, da deluje bolj natančno in z večjo hitrostjo. Elektronski prenos poslovnih dokumentov, kot so naročilnice, ponudbe, računi in podatki o plačilu, nadomešča ustno in pisno komunikacijo (Raney in Walter, 1992).

Elektronska izmenjava podatkov za svoje delovanje potrebuje štiri elemente (Corbitt, 1992):

- elektronsko pošto za hitro medsebojno komuniciranje,
- »online« omrežje za zagotavljanje hitrega sporočanja,
- elektronsko poslovno dokumentacijo,
- standardne protokole za prenos datotek.

Elektronsko izmenjava podatkov lahko razumemo kot nadomestitev naročilnic v papirni obliki z elektronskimi ekvivalenti. Elektronska izmenjava podatkov podpira vnaprej natisnjene obrazce ali poslovne dokumente in jih po elektronski poti posreduje med računalniškimi aplikacijami (Clarke, 2001). Implementacija elektronske izmenjave podatkov poleg izboljšave izmenjave informacij lahko vodi tudi k povečanju količine podatkov, prenesenih med družbami (Iskandar, 2000, str. 25).

Ena od tehnologij, ki omogoča hitro izmenjavo računalniških sporočil med aplikacijami, je ServiceMix. Podjetje Actual I. T. se namreč pri razvoju informacijskih sistemov na področju logistike pristaniških dejavnosti srečuje s potrebo špediterjev, da imajo vpogled v stanje posredovanih sporočil. Ta povratna informacija oziroma pregled bi jim omogočil pregled nad stanjem sporočil. To pomeni, da v primeru, ko špediter pošlje sporočilo (npr. dispozicijo na kontejnerski terminal, manifest v carinsko upravo ipd.) in zaradi kakršnega koli razloga ne dobi odgovora, nima pregleda, kakšno je stanje poslanega sporočila. V ta namen bomo izdelali pregledovalnik sporočil, ki je spletna stran, namenjena špediterjem, s katero lahko pregledajo, kakšno je stanje njihovih poslanih sporočil.

V prispevku bomo zato analizirali sodobno programsko opremo in računalniške sisteme v logistiki z namenom izgradnje pregledovalnika sporočil. Na podlagi teorije in prakse bomo v prispevku prikazali arhitekturo programa oziroma spletne aplikacije, ki se povezuje na posrednika sporočil (ServiceMix), z namenom, da lahko pregledovalnik sporočil špediterju prikaže stanje njegovih poslanih sporočil v Luko.

2 METODOLOGIJA

Raziskavo smo izvedli z analizo literature in razvojem programske opreme. V prvem delu prispevka smo s pomočjo analize obstoječe literature opisali delovanje posrednika sporočil ter pregledali prakso in uporabnost tehnologij glede posrednikov sporočil. Pregledovalnik sporočil smo zgradili sami s pomočjo tehnologij ASP.NET, programsko opremo MVC 4 in pogonom Solr. Testirali smo delovanje pregledovalnika sporočil in na podlagi rezultatov testa bomo sestavili okvir, v katerem smo predstavili vlogo pregledovalnika sporočil v posredniku sporočil.

3 RAZISKAVA POSREDNIKA SPOROČIL

Posrednik sporočil je sporočilni sistem za aplikacije, ki vključuje prenos sporočil, pravila in oblikovanje (Computer Desktop Encyclopedia, 2014). Posrednik sporočil je kot osrednje vozlišče (central hub) namenjeno okolju strank in zagotavljanju zmogljivosti razširjenega ESB (Enterprise Service Bus)[1] ali arhitekturnega modela programske opreme za oblikovanje in izvajanje komunikacije med seboj povezanih aplikacij v storitveno usmerjeno arhitekturo (SOA – Service-Oriented Architecture),[2] zaradi česar je kontrola transakcij še bolj pomembna v sestavljenih aplikacijah. Takšna arhitektura posrednika sporočil namreč omogoča večjo odzivnost na zahteve strank in povezovanje podjetij s poslovnimi strankami (Business Wire, 2006; PR Newswire, 2003).

Posrednik sporočil deluje kot strežnik, ki služi kot vmesnik med aplikacijami, ki omogoča pošiljanje podatkov na drugo aplikacijo, nazaj na aplikacijo ali asinhrono med aplikacijami. Posrednik sporočil izvaja operacije na sporočila, ki jih prejme. Te operacije vključujejo (Reinshagen, 2001; Huang, 2007; Msdn. Microsoft.com, 2014; Hohpe, 2003):

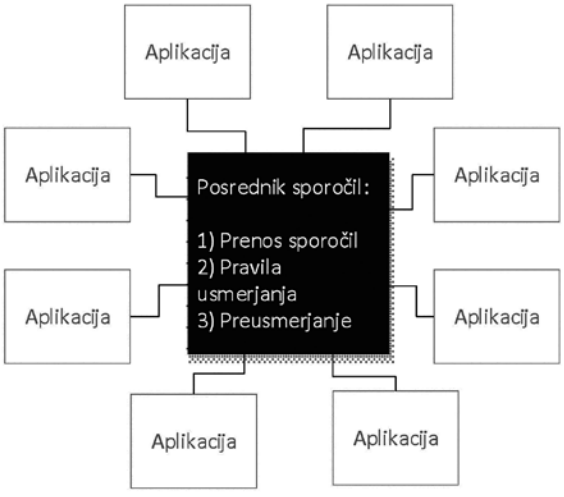
- obdelavo glave,
- varnostno preverjanje in šifriranje/dešifriranje,
- napake in obravnavanje izjem, odzivanje na dogodke in napake,
- usmerjanje od ene ali več destinacij,

[1] ESB je dodatek za podjetniško arhitekturno povezavo. To zagotavlja hitrejše, enostavnejše in bolj prožno integracijo, ki omogoča, da se integracija, ki bi jo izvedli, odzove na hitrost poslovanja (Credle idr., 2007).

[2] SOA (Service-oriented architecture): pri čemer »*Service-oriented*« definira način povezovanja poslovne aplikacije in procese povezanih storitev. Z arhitekturne perspektive je SOA arhitekturni slog, ki podpira usmerjanje storitev. Zaradi tega na izvedbeni ravni SOA izpolnjuje uporabo standardov, ki temeljijo na infrastrukturi, programskih modelih in tehnologiji, kot so spletne storitve (Credle idr., 2007).

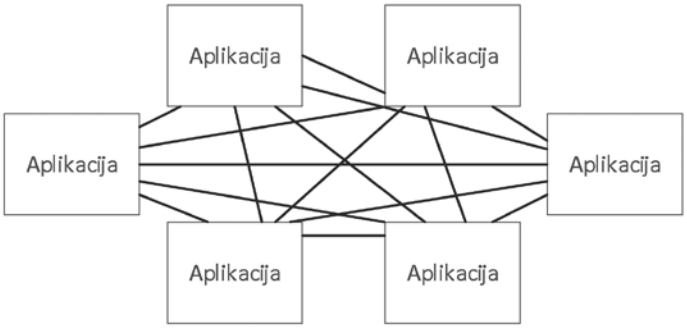
- invokacijo, sklicevanje spletnih storitev za pridobivanje podatkov,
- preoblikovanje sporočil na alternativno zastopanje.

Namesto da aplikacije komunicirajo druga z drugo, komunicirajo s posrednikom sporočil. Aplikacija mu pošlje sporočilo ter mu zagotovi logično ime prejemnikov. Posrednik sporočil pregleda aplikacije, ki so registrirane v skladu z logičnim imenom, nato jim pošlje sporočila. Komunikacija med aplikacijami vključuje le informacije o pošiljatelju, posredniku sporočil in predvidenih prejemnikih. Posrednik sporočil ne pošilja sporočila samodejno v katero koli drugo aplikacijo (Msdn.Microsoft.com, 2014). Na sliki 1 je prikazana konfiguracija posrednika sporočil. Uporaba posrednika sporočil omogoča prejemanje sporočila z različnih destinacij na podlagi določitve pravilne destinacije in usmerjanja sporočil na pravilne kanale (Hohpe in Woolf, 2004).



Slika 1: **Posrednik sporočil pri posredovanju sodelovanja med sodelujočimi aplikacijami (Msdn.Microsoft.com, 2014)**

Brez posrednika sporočil bi morala biti vsaka aplikacija posebej programirana, da kliče drugo aplikacijo in da dostavlja sporočila. Povezava aplikacij z namenom izmenjave sporočil je prikazana na sliki 2.

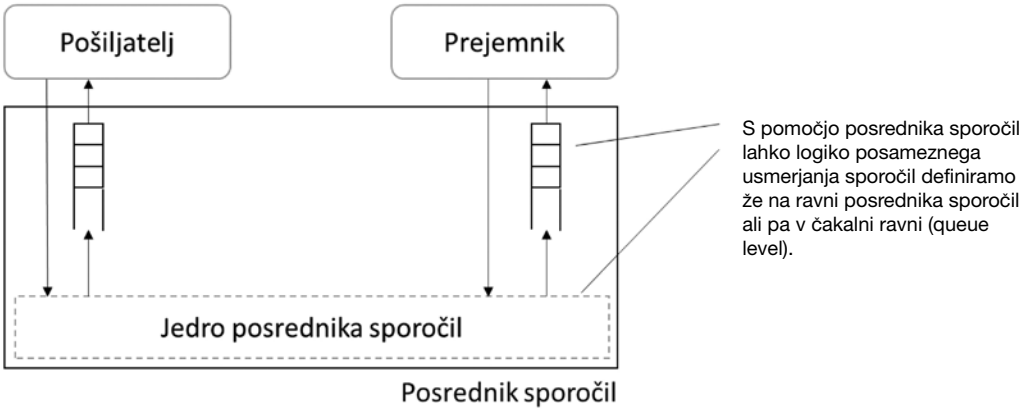


Slika 2: **Povezava aplikacij za izmenjavo sporočil brez posrednika sporočil (Computer Desktop Encyclopedia, 2014)**

3.1 Osnovna arhitektura posrednika sporočil

Za obdelavo sporočil posrednik sporočil omogoča razvijalcem, da si določijo logiko na vmesni[3] ravni. Prednost tega pristopa je, da je logika glede pošiljanja sporočil umeščena na vmesni ravni in ni porazdeljena med aplikacijami, tako da moramo v primeru, če je treba spremeniti logiko pošiljanja sporočil med aplikacijami, spremeniti le modul na vmesni ravni. Druga prednost tega pristopa je, da sta pošiljatelj in prejemnik nevezana (Anderson, 2006). Na sliki 3 je prikazana logika delovanja posrednika sporočil in njegova osnovna arhitektura.

[3] Vmesna raven (angl. middleware level) zagotavlja arhitekturo, ki služi kot osrednja točka za komunikacijo med vsemi aplikacijami (Computer Desktop Encyclopedia, 2014).



Slika 3: **Osnovna arhitektura posrednika sporočil (Anderson, 2006)**

Osnova pri usmerjevalni logiki posrednika sporočil so identiteta pošiljatelja (kdo je pošiljatelj), vrsta sporočila ter vsebina sporočila. Usmerjevalna logika je običajno napisana v jeziku, ki temelji na pravilih, kot je na primer pravilo o dostavi. (Npr. če je bilo neko sporočilo poslano iz enega sistema v drugi sistem, potuje najprej v posrednik sporočil. Nato se to sporočilo v posredniku obdela po določenih pravilih in se ovrednoti. Če se sporočilo ovrednoti za veljavno, se stanje sporočila postavi v stanje, da je treba sporočilo poslati v drugo aplikacijo.) (Anderson, 2006)

3.2 Povezava med aplikacijami v posredniku sporočil

Čeprav računalniki med seboj komunicirajo na različnih ravneh, obstaja veliko razlik med različnimi vrstami računalniških sistemov, ki sodelujejo med seboj, osnovni komunikacijski proces pa je relativno enoten in standardiziran (Weed, 2014). TCP/IP je niz komunikacijskih protokolov, ki omogočajo, da računalniki med seboj komunicirajo na internetu. Njegova dva glavna protokola sta TCP (protokol za krmiljenje prenosa) in IP (internetni protokol). Protokola TCP in IP določata, kako naj se naprave povežejo z internetom in kako naj se podatki prenašajo prek naprav (Warren, 2012).

Povezavo aplikacij v posredniku sporočil omogočajo te komponente (Coles, 2013):

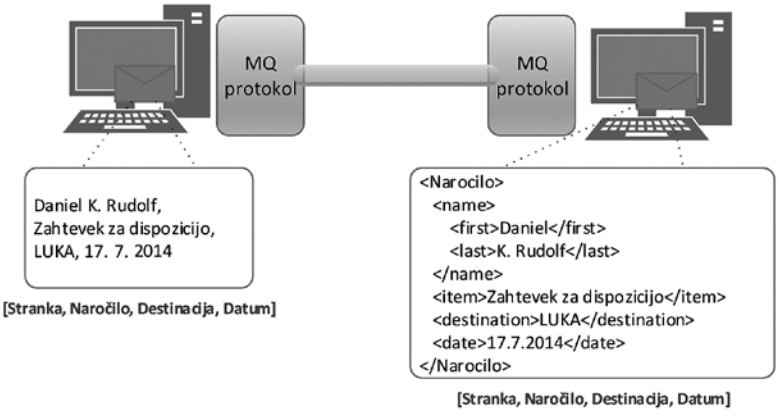
- komunikacijski protokoli, npr. MQ,[4] TCP/IP, HTTP; datotečni sistem, SMTP, POP3 itd.,
- strukture in oblike zapisov podatkov, npr. binarni (C/COBOL), XML, industrijski (SWIFT, EDI, HL7), uporabniško definirani itd.,
- pravila za obdelavo sporočil, npr. usmerjanje, spreminjanje, obogatitev, filtriranje, monitoriranje, distribuiranje, razstavljanje, korelacija, zahtevek/odgovor, objavi/naroči, seštevanje itd.

Na podlagi zapisanega lahko ugotovimo, da:

- aplikacije med seboj v posredniku sporočil komunicirajo prek komunikacijskih protokolov; tipični protokoli v uporabi vključujejo TCP/IP in protokole višje ravni, kot so FTP, SMTP in http;
- prek komunikacijskega protokola aplikacije izmenjujejo podatke običajno v strukturi in oblikah zapisov, znanih kot sporočila; oblike teh sporočil se lahko opredelijo iz struktur C ali COBOL copybox ali pa preprosto uporabimo standardizirani format, kot je XML;
- da povežemo aplikacije skupaj, tako da so njihovi protokoli in sporočila interoperabilni, je treba pravila za obdelavo sporočil uporabiti v enem ali obeh sistemih, ki jih poskušamo povezati. Ta pravila so lahko relativno preprosta, na primer usmerjanje sporočil z enega mesta na drugo ali transformacija ene oblike sporočila v drugo itd.

Primer pravila za obdelavo sporočil v posredniku sporočil je prikazan na sliki 4. Primer prikazuje pošiljanje sporočila iz enega sistema (aplikacije) v drug sistem prek protokola MQ, pri čemer se poslano sporočilo transformira v obliko XML.

[4] Protokol MQ »Message Queues« ali čakalna vrste so programskoinženirske komponente, ki se uporabljajo za medprocesno komuniciranje (IPC) ali za komunikacijo znotraj istega procesa.



Slika 4: **Vzorec posredovanja v posredniku sporočil – transformacija**

3.3 Seznam posrednikov sporočil

Namen posrednika sporočil je, da sprejme sporočilo od aplikacije in izvede določene akcije na teh sporočilih. Poznamo različne posrednike sporočil.

- Apache ServiceMix je fleksibilni, odprtokodni kontejner oziroma »vsebovalnik«, ki združuje funkcije in funkcionalnosti Apache ActiveMQ, Camel, CXF in Karaf v močno delujočo platformo, ki jo lahko uporabljamo za izgradnjo lastne integracijske rešitve (http://servicemix.apache.org/).
- JBoss Messaging + FUSE Message Broker: JBoss-MQ je zmogljiva, prilagodljiva platforma za pošiljanje sporočil, ki zagotavlja varno dostavo informacij ter omogoča internet stvari. JBoss lahko preprosto namestimo in upravljamo, pri čemer se lahko v kateri koli konfiguraciji razporedi mreža posrednikov prek infrastrukture ne glede na to, ali gre za oblačno ali hibridno konfiguracijo (http://www.jboss.org/products/amq/overview/).
- Microsoftov BizTalk strežnik združuje EAI (Enterprise Application Integration) in integracijo B2B (Business-to-Business). BizTalk zagotavlja močan spletni razvoj in izvedbo okolja, ki zagotavljata dolgotrajne poslovne procese tako znotraj podjetij kot med njimi. Posrednik sporočil omogoča standardni prehod za pošiljanje in prejemanje dokumentov prek interneta, prav tako ponuja širok nabor storitev, ki zagotavljajo celovitost podatkov, dostavo, varnost in podporo (http://www.microsoft.com/en-us/server-cloud/products/biztalk/default.aspx#fbid =6Gd_vzTYZ8Q).
- Oracle Message Broker ima pomembno vlogo pri Oraclovih sistemih EAI (Enterprise Application Integration). Gre za prilagodljivo in odprto platformo, idealno za vključevanje strateških aplikacij, e-poslovanja ali obstoječih sistemov. Oracle Message Broker omogoča odprt, asinhron, sistemsko neodvisen, sporočilen komunikacijski mehanizem (http://docs.oracle.com/cd/A87860_01/doc/ois.817/a65435.pdf).
- Posrednik sporočil JORAM vključuje Java implementacijo specifikacije JMS 1.1 (Java Message Service API). To omogoča dostop do zelo porazdeljenega MOM (Message Oriented Middleware tehnologija). Projekt JORAM se je začel že leta 1999 in je odprtokodna programska oprema, izdana pod licenco LGPL od leta 2000 (http://joram.ow2.org/).
- RabbitMQ (Mozilla Public License): Gre za aplikacijo za robustno pošiljanje sporočil, preprosto za uporabo (pokriva knjižnice programskih jezikov C#, Python, Java, Ruby in PHP). Deluje na vseh večjih operacijskih sistemih in podpira veliko število razvojnih platform. Na voljo sta odprtokodna in komercialno podprta različica (http://www.rabbitmq.com/).

Drugi uporabni posredniki sporočil so tudi:

- Niklas message broker (http://www.copernicus.nl/),
- Python Message Service: (http://pubsub.sourceforge.net/),
- WebSphere Message Broker: (http://www-03.ibm.com/software/products/en/ibm-integration-bus),
- SAP PI: (http://scn.sap.com/docs/DOC-41766),

- Sapo Broker: (http://oss.sapo.pt/),
- Spread Toolkit: (http://www.spread.org/),
- Axway integration Broker: (http://www.axway.com/products-solutions/integration/integrator).

Če skrbno pregledamo, kaj ponujajo odprtokodni in licenčni posredniki sporočil, hitro ugotovimo, da ServiceMix zagotovo ne bi bil prva izbira za razvoj aplikacij v smislu namizne programske opreme. Vendar je v primeru, ko se spopadamo z bolj kompleksnim okoljem, v katerem morajo sodelovati različne aplikacije, ServiceMix primernejši, saj lahko učinkovito »ublaži« breme.

Uporaba standardiziranih sporočil in natančno opredeljenih pravil za obdelavo sporočil nam poleg drugih koristi, ki jih prinaša uporaba komponente ServiceMix, pomaga razviti tudi rešitev (programsko opremo), ki bo v kompleksnem okolju delovala brezhibno in bo povezovala različne aplikacije.

3.4 Posrednik sporočil ServiceMix

Apache ServiceMix je fleksibilni odprtokodni kontejner oziroma »vsebovalnik«, ki združuje funkcionalnosti Apache ActiveMQ,[5] Camel,[6] CXF[7] in Karaf[8] v močno delujočo platformo, namenjeno izgradnji lastne povezane rešitve (ServiceMix, 2014). Gre za odprtokodni ESB (Enterprise Service Bus), ki omogoča hitrejšo, enostavnejšo in bolj prožno integracijo ter temelji na specifikaciji Java Business Integration (JBI). ServiceMix zagotavlja prilagodljivo razporeditev ter zanesljivost prek sporočil, usmerjenih na »middleware« tehnologijo, kot so JMS (Java Messaging Service) ali standardi spletnih storitev. ServiceMix je zgrajen okoli specifikacije Java Business Integration (JSR 208), ki določa, da vmesnik programske aplikacije (API) sporoča storitve in usmerjevalnik, ki uporablja Java Management Extension, ravnata z vprašanji, kot sta na primer uvajanje in nadziranje. Prednost ServiceMixa je, da omogoča poslovno integracijo z znatno nižjimi stroški poslovanja (Davies in Strachan, 2005; Rick, 2005).

Glavne značilnosti ServiceMixa so (ServiceMix, 2014):

- zanesljivo sporočanje z Apache ActiveMQ,
- sporočanje, usmerjanje in modeli podjetniške integracije (Enterprise Patterns Integration) z Apache Camel,
- WS-* in RESTful spletne storitve z Apache CXF,
- strežnik OSGi, ki ga poganja Apache Karaf.

ServiceMix vsebuje različne sestavne dele in storitve (Hanson, 2005).

- Komponente storitev:
  - usmerjanje, ki temelji na pravilih prek pogona Drools,
  - implementira Web Services Notification,
  - vključuje podporo Business Process Execution Language (BPEL) za Web Services BPEL prek PXE (Preboot Execution Environment – predzagonska izvedba okolja),
  - podpora za Java Connector Architecture,
  - skriptna podpora, ki omogoča katero koli specifikacijo Java.
- Povezave SOAP:
  - integracija z Apache Web Service Invocation Framework (WSIF),
  - podpora za Java API, ki temelji na spletnih storitvah XML,
  - API za XML (StAX) prek ActiveSOAP.
- Transportne povezave:
  - podpora e-pošte prek JavaMail,
  - podpora HTTP na strežniku ali pri odjemalcu,
  - podpora JMS prek ActiveMQ,
  - podpora FTP prek knjižnice Jankarta Commons Net.

Da bi standardizirali koncept ESB, je Java Community Process (JCP) nastopil s standardom JBI. Ta opisuje vse sestavne dele, ki jih najdemo v konceptu ESB. Apache ServiceMix je med drugim ena izmed najbolj uporabnih in izvajalnih ESB, ki so zavezani k standardu JBI (Irriger, 2010). Slika 5 prikazuje razmerje med JBI in ServiceMix, pri čemer ServiceMix vključuje popolno posodo JBI (kontejner), ki podpira vse dele specifikacije JBI vključno (Hanson, 2005):

- s storitvijo normaliziranega sporočila[9] in usmerjevalnika,
- z JBI Management Beans (MBeans),[10]

[5] ActiveMQ: gre za posrednik sporočil, ki implementira JMS API (Java Message Service).

[6] Camel omogoča izboljšanje definiranja, usmerjanja in posredovanja pravil v različnih domensko specifičnih jezikih, vključno z Java Fluent API, Spring ali Blueprint XML Configuration datoteke ter Scala DSL. To pomeni, da dobimo pametno dokončanje pravil usmerjanja v IDE, ne glede na to, ali je v Javi, Scala ali XML Editor.

[7] CXF je okvir odprtokodne storitve. CXF pomaga pri izgradnji in razvoju storitev z uporabo »frontend« programiranja API, kot sta JAX-WS in JAX-RS. Te storitve lahko komunicirajo prek različnih protokolov, kot so SOAP, XML/http, RESTful HTTP ali CORBA, in deluje prek različnih transporterjev, kot so HTTP, JMS ali JBI.

[8] Ukazi Karaf omogočajo ogled, start, stop, pridobivanje informacij o kontekstu in usmerjevalnikih Camel, ki potekajo v instanci Karaf.

[9] Normalized Message je v tem delu mišljen kot usmerjevalnik normaliziranega sporočila, ki je del okolja JBI in je odgovoren za posredovanje sporočil med komponentami JBI.

[10] MBeans je nadziran objekt Java, podoben komponenti JavaBeans, ki sledi vzorcu zasnove specifikacije JMX. MBean so lahko naprave, aplikacije ali kateri koli drugi viri, ki jih je treba upravljati.

- z nalogami za upravljanje in namestitev komponent,
- s polno podporo za uvajanje enot JBI z uvajanjem komponent JBI.

Večina procesov inicializacije ServiceMix, procesov aktivacije in procesov izmenjave sporočil vključuje neko obliko komunikacije in/ali interakcijo s sestavnimi deli, ki temeljijo na JBI (Hanson, 2005). Storitve JBI (slika 5) vključujejo transformacijske, usmerjevalne in korelacijske storitve. Pri komponentah pa se standard JBI fokusira samo na osnovno infrastrukturo. Definira, katere komponente uporabiti, kakšna sporočila so v sistemu in v kakšni interakciji so. Oboje skupaj (storitve in komponente) pa podpirajo vzorce za izmenjavo sporočil (Irriger, 2010).

3.5 Pošiljanje sporočil v logističnem procesu

Logistika je proces načrtovanja, izvajanja in nadzorovanja učinkovitega in stroškovno učinkovitega pretoka in skladiščenja surovin in blaga ter s tem povezanih informacij od točke izvora do točke potrošnje. Pri tem elektronska izmenjava podatkov zagotavlja sredstva za doseganje potrebnega usklajevanja pri podpori logističnih operacij z namenom upravljanja naročil do dostave pošiljk (Daugherty, Germain in Droge, 1995). Konceptualni model elektronske izmenjave podatkov v logističnem procesu prikazuje slika 6.

Implementacija elektronske izmenjave podatkov ima pomembne koristi za vsa podjetja, ki sodelujejo v dobavni verigi. Prav tako je elektronska izmenjava podatkov bistvenega pomena za mednarodne špediterje. Konceptualni model elektronske izmenjave podatkov prikazuje faze, ki pomagajo pri opredelitvi strategije za njeno izvajanje v okviru podjetja. Model na sliki 6 prikazuje vpliv sprememb na različnih ravneh znotraj družbe. Gre za štiri osnovne ravni, ki so (od zgoraj navzdol): načrtovanje, oblikovanje, namestitev in podpora. Elektronska izmenjava podatkov na splošno vpliva na kulturo podjetja, prakso upravljanja in način, kako je podjetje strukturirano. Na ravni poslovnih procesov lahko elektronska izmenjava podatkov zahteva spremembe v operativnih postopkih, podatkovnih vnosih, strukturi poslovnih listin in načinu pretoka informacij v podjetju. Informacijski sistemi morajo biti prilagojeni, ker elektronska izmenjava podatkov pomeni uporabo posebnega prevajanja in komunikacijske programske opreme (Jackson in Sloane, 2003).

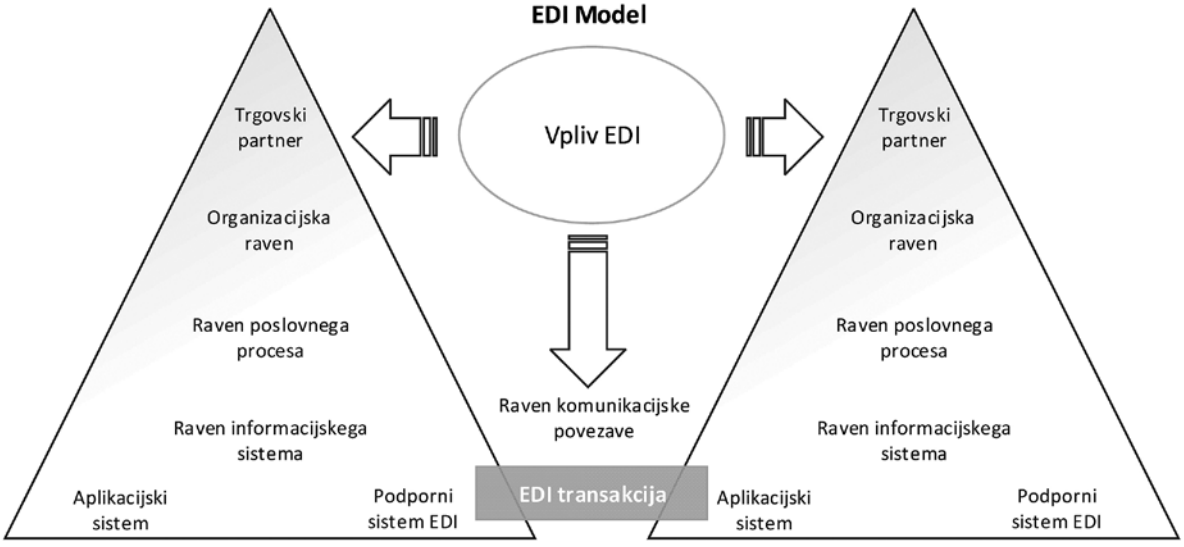
3.6 Pregledovalnik sporočil, njegov namen, struktura in zgradba

Namen pregledovalnika sporočil je prikaz stanja sporočil, ki jih špediterji pošiljajo med sistemi. Špediterji pri posredovanju sporočil prek posrednika sporočil želijo vedeti, kakšno je stanje njihovih sporočil, ali je poslano, sprejeto ali je prišlo do napake. Za izdelavo pregledovalnika sporočil smo uporabili napredne tehnologije:

- programski jezik ASP.NET,
- programski arhitekturni vzorec MVC (Model-View-Controller),
- pogon Solr.

Spletno aplikacijo oziroma pregledovalnik sporočil smo ustvarili s pomočjo programskega jezika ASP.NET s pristopom MVC 4, ki razdeli aplikacijo v

Slika 5: **JBI in ServiceMix (Hanson, 2005)**



Slika 6: **Konceptualni model elektronske izmenjave podatkov v logističnem procesu (Jackson in Sloane, 2005)**

tri med seboj povezane dele tako, da se ločijo interne predstavitve informacij od oblike informacij, ki so predložene ali sprejete od uporabnika (Reenskaug in Coplien, 2009; Burbeck, 1992).

Osrednji del – model – je sestavljen iz podatkov aplikacije, poslovnih pravil, logike in funkcij. Pogled je lahko kateri koli izhod za predstavitev informacij, npr. grafikon, diagram, tabela. Tretji del je krmilnik, ki sprejema vhod in ga pretvori v ukaze za model ali pogled (Code Project, 2008).

Uporabljeni pogon Solr nam omogoča hitro iskanje podatkov. Izbrali smo izredno znano, zmogljivo in hitro odprtokodno iskalno platformo. Solr je samostojni iskalni strežnik, ki z okolico (okoliškimi sistemi, aplikacijami) komunicira s pomočjo XML in HTTP. Komunikacija je omejena predvsem na podatke, povezane z indeksiranjem vsebin ali z izvedbo iskanja. Praviloma s pogonom Solr najprej izvedemo začetno indeksiranje vsebine in v nadaljevanju sproti indeksiramo vse dodatne podatke, ki se dodajajo k vsebini, ali pa izvajamo poizvedbe po vsebini. Kot smo že navedli, je iskanje s pomočjo pogona Solr izredno učinkovito in hitro.

Pri uporabi pogona Solr se praviloma sklicujemo na bogato specifikacijo sheme, ki nam omogoča dobro prilagodljivost – predvsem pri obvladovanju in opisovanju različnih področij (npr. glava sporočila, telo sporočila) dokumenta. Iskalni pogon v svojem jedru uporablja za iskanje iskalno knjižnico Lucene Java. Pogon trenutno uporabljajo aplikacije z »močnimi« iskalnimi funkcijami; naj naštejemo nekaj podjetij, ki uporabljajo pogon Solr v svojih aplikacijah: eBay, CISCO, NASA, CNET Chanell, Apple Inc. idr. Sam pogon Solr je aplikacija Java, vendar se vse interakcije s Solr izvajajo z objavo sporočil prek protokola HTTP (v JSON, XML, CSV ali v binarni obliki); ta interakcija je namenjena predvsem za indeksiranje dokumentov. Tudi pri iskanju pogon z okolico komunicira s pomočjo protokola HTTP, z ukazom GET okoliški sistemi lahko dobijo rezultate v oblikah JSON, XML ali v drugih formatih (Python, Ruby, PHP, CSV, C#, itd.). Za delovanje strežnika Solr je treba namestiti Java 1.5 in primeren aplikacijski strežnik (kot je npr. Tomcat), ki podpira standard Servlet 2.4 (https://wiki.apache.org/solr/FAQ#General).

V programskem jeziku ASP.NET Solr uporabljamo s pomočjo uvožene knjižnice SolrNet (zadnja dostopna različica 0.4.0-beta2). Za izvedbo poizvedbe v Solr je treba ustvariti glavni krmilnik. Gre za kontrolo, ki požene poizvedbo Solr. Za krmilnik uporabimo ukaz, prikazan na sliki 7.

Struktura glavnega krmilnika je sestavljena iz delov, ki pošlje podatke v arhitekturo View (MVC) za prikaz. Posamezna vrstica kode v glavnem krmilniku (slika 6) izvede te operacije:

- var start: funkcija izračuna (deklarira) na podlagi iskanega parametra (gre za parameter, na podlagi katerega želimo izvesti poizvedbo) in PageSize (PageSize določimo samostojno, določa pa, koliko strani naj se prikaže na eno stran), koliko strani

```
public ActionResult Index(SearchParameters parameters)
{
  var start = (IskaniParameter.PageIndex – 1)*IskaniPrameter.PageSize;
  var prikaz = solr.Query(BuildQuery(IskaniParameter), new QueryOptions
  {
    FilterQueries = BuildFilterQueries(IskaniParameter),
    Rows = IskaniParameter.PageSize,
    Start = start,
    OrderBy = GetSelectedSort(IskaniParameter),
    // Določimo lahko še »Preverjanje besed«, »Grupiranje« itd.
  }
//Nato v isti metodi določimo še element, ki pošlje podatke v Pregledovalnik (»View«)
var pregled = new SpremenljivkeZaPregled
{
  Spremenljivke = prikaz,
  Iskanje = IskaniParameter,
  SkupajStevilo = prikaz.NumFound,//Ta del prešteje koliko je vse skupaj strani.
  //Ter ostale elemente, ki jih želimo prikazati: Grupiranje, Preverjanje besed (SpellCheck), itd.
}
return View(pregled);
}
```

Slika 7: **Solr struktura glavnega krmilnika pregledovalnika sporočil (Lastni vir, 2014)**

se bo prikazalo. Funkcijo uporabljamo zato, da lahko prikazno stran razdelimo na več podstrani. Funkcija je koristna zaradi boljše preglednosti, ker je namen Solr prikaz večjega števila podatkov v zelo kratkem času.

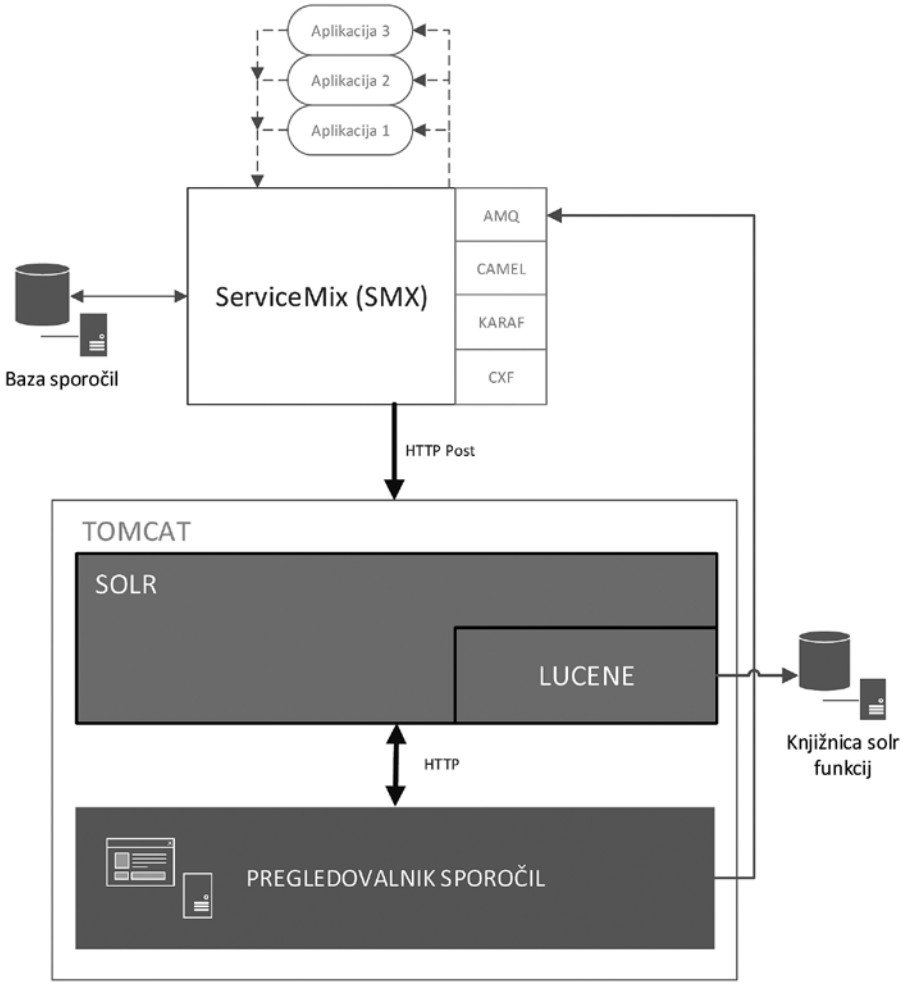
- var prikaz: sestavlja več delov, ki se združijo in izvedejo poizvedbo (vrstni red sestavnih delov ni pomemben). FilterQueries vrne parametre, ki jih je vnesel uporabnik, da na podlagi teh izvede poizvedbo. Start vrne rezultat, koliko strani se bo prikazalo na eno stran. Rows vrne število vseh strani, ki jih bo aplikacija prikazala na strani. OrderBy pove aplikaciji, kako naj sortira podatke iz iskanega parametra. Sortiranje (OrderBy) deklariramo pred glavnim krmilnikom, v katerem določimo, kako naj se posamezni parameter sortira (naraščajoče ali padajoče).
- var pregled: prav tako sestavlja več elementov, ki se združijo v celoto, ki jih pošljemo v prikaz.

4 ANALIZA VLOGE POSREDNIKA SPOROČIL V PREGLEDOVALNIKU SPOROČIL

Posrednik sporočil je kot arhitekturni vzorec namenjen validaciji, transformaciji in usmerjanju sporočil. Posrednik sporočil omogoča komunikacijo med aplikacijami, zato da si te lahko izmenjujejo sporočila. Za analizo smo uporabili posrednik sporočil ServiceMix. Kombinacija funkcij ServiceMix nam omogoča preprosto definiranje usmerjanja in posredovanja pravil v različnih programskih jezikih z namenom preprostega posredovanja sporočil. Slika 8 prikazuje vlogo posrednika sporočil, ki jo ima do pregledovalnika sporočil Solr.

Namen pregledovalnika sporočil je, da uporabniku pri pošiljanju sporočil iz ene aplikacije v drugo pri logističnem procesu omogoči pregled stanja njegovega sporočila. Sporočilo je bodisi sprejeto, zavrnjeno ali v pregledu. Pregledovalnik sporočil deluje po postopku (glej sliko 8):

1. Izvorni sistem pošlje sporočilo v posrednik sporočil (ServiceMix).
2. Posrednik sporočil sprejme, shrani in obdela poslano sporočilo ter pošlje sporočilo po določenih pravilih drugi aplikaciji.
3. Posrednik sporočil vsa prejeta sporočila »logira« oziroma zapiše v dnevnik zapisov.
4. Shranjena sporočila v posredniku sporočil pogon Solr po prej opredeljeni shemi indeksira vsa sporočila.
5. Špediter se prijavi v pregledovalnik sporočil (Message Viewer).
6. Prijavljeni špediter vpiše interno številko (interna številka je številka dokumenta, ki jo je ustvarila njegova aplikacija, ko je ustvaril sporočilo, ki ga je poslal v luški sistem) v pregledovalnik sporočil za pregled stanja poslanega sporočila v luški sistem.



Slika 8: **Posrednik sporočil SMX v vlogi pregledovalnika sporočil Solr**

7. Špediterjeva zahteva za poizvedbo se izvede prek protokola HTTP.
8. Pregledovalnik sporočil s pomočjo pogona Solr ustvari poizvedbo, s katero bo pridobil zahtevane podatke iz baze posrednika sporočil. V ozadju pogona Solr deluje pogon Lucene,[11] ki vsebuje knjižnico za izvajanje funkcij Solr (poizvedb).
9. V ozadju pregledovalnika sporočil in pogona Solr je postavljen spletni strežnik Tomcat, ki prek protokola HTTP izvede poizvedbo, ki na podlagi indeksiranih sporočil iz posrednika sporočil poišče iskano sporočilo.
10. Pregledovalnik sporočil iz posrednika sporočil pridobi podatke o stanju sporočil glede na poizvedbo, ki jo je špediter zahteval prek pregledovalnika sporočil.

Predlagani koncept pregledovanja sporočil prek posrednika sporočil omogoča uporabnikom v logističnem procesu, da se informirajo glede stanja svojih sporočil, ki jih pošiljajo v druge aplikacije. Funkcionalnost pogona Solr se je izkazala kot odličen pristop za iskanje podatkov. Solr namreč lahko zelo hitro najde iskane podatke, kar je priporočljivo za pregledovalnik sporočil. Za logistični proces je namreč pomembna informacija o stanju sporočila, s katerim se uporabniki (trgovski partnerji, ki si izmenjujejo sporočila, vzdrževalci sistemov in drugi, ki so vključeni v logistični proces) informirajo, da lahko pravočasno ukrepajo v primeru, če pride do napake.

[11] Lucene: Gre za brezplačno odprtokodno programsko knjižnico, namenjeno priklicu informacij, napisano v Javi. Jedro Lucene zagotavlja indeksiranje in iskalno tehnologijo, kot so spellchecking (ponudi podoben iskani parameter, če je iskani napačno vnesen), napredne analize/tokanizacija itd.

5 SKLEP

Posrednik sporočil je eno izmed orodij, ki v logističnem procesu omogoča izmenjavo elektronskih sporočil med različnimi aplikacijami, vendar ne omogoča pregledovanja sporočil glede na stanje sporočil oz. kaj se je zgodilo s sporočilom, ki je bilo poslano v neki drug sistem. Špediterja, ki pošilja sporočila med sistemi, pogosto zanima, kakšno je stanje njegovega sporočila, ki ga je poslal v drug sistem. Z raziskavo smo prišli do ugotovitve/rezultata, da je koncept pregledovalnika sporočil v obliki spletne strani, ki pridobiva podatke iz posrednika sporočil, najbolj ustrezen način posredovanja sporočil. Vloga pregledovalnika sporočila je v tem, da na posredniku sporočil izvede poizvedbo in prikaže rezultate. Gre za rezultate, ki špediterju kažejo, kakšno je stanje njihovega poslanega sporočila v drug sistem. Pregledovalnik sporočil je zgrajen iz različnih orodij, prednost izbranega pregledovalnika sporočil v obliki spletne strani pa je, da lahko deluje na svetovnem spletu, kar omogoča špediterjem, da do informacij glede stanja svojih sporočil dostopajo s katero koli napravo, ki ima dostop do interneta. Druga prednost takšnega pregledovalnika je uporaba tehnologije Solr, ki omogoča špediterju, da v obsežnem nizu sporočil zelo hitro najde iskano vrednost. Ugotovili smo, da moramo biti pri izbiri orodij pozorni, da izberemo tista orodja, ki bodo delovala na strojni opremi in na katerih lahko zgradimo pregledovalnik sporočil. Zato smo izbrali odprtokodni ServiceMix za posrednika sporočil, ker podpira uporabo tehnologije Solr, ki je jedro pregledovalnika sporočil. Predlagana programska orodja in tehnologije dajejo smernice za nadaljnji razvoj orodij na področju elektronske izmenjave podatkov.

6 VIRI IN LITERATURA

[1] Anderson, M. K. (2006). Enterprise Application Integration. ZDA: Univerity of Colorado. Pridobljeno 24. 6. 2014 s http://www.cs.colorado.edu/~kena/classes/7818/f06/lectures/05/.

[2] Burbeck, S. (1992). Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). Pridobljeno 30. 6. 2014 s http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.

[3] Business Wire (2006). MQSoftware Extends Message Broker Functionality with Transaction Monitoring; Q Nami!(R) Supports WebSphere(R) Message Broker V6.0.0.1. ZDA: Business Wire.

[4] Clarke, R. (2001). Electronic Data Interchange (EDI). ZDA: National Association of Credit Management, 9(103), str. 23–25.

[5] Clifton, R. G. (1989). EDI: A better Way of Doing Business. ZDA: Industrial Distribution: 78(4), str. 61.

[6] Code Project (2008). Simple Ecample of MVC (Model View Controller) Design Pattern for Abstraction. Pridobljeno 30. 6. 2014 s http://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design.

[7] Coles, D. (2013). First Steps With WebSphere Message Broker: Application Integration for the Messy. ZDA, San Francisco: IBM.

[8] Computer Desktop Encyclopedia (2014). Message Broker. Pridobljeno 9. 7. 2014 s http://www.computerlanguage.com/.

[9] Corbitt, T. (1992). Electronic Data Interchange (EDI). Velika Britanija: Institute of Management Services, 2(36), str. 1–20.

[10] Credle, R., Adams, J., Clark, K., Peng Ge, Y., Jeter, H., Lopes, J., Nasser, S. in Peri, K. (2007). Patterns: SOA Design Using WebSphere Message Broker and WebSphere ESB. ZDA: International Business Machines Corporation (IBM).

[11] Daugherty, P. J., Germain, R. in Droge, C. (1995). Predicting EDI technology adoption in logistics management: The influence of context and structure. Kanada: University of British Columbia, Faculty of Commerce and Business Administration, 31(4), str. 309–327.

[12] Davies, R. in Strachan, J. (2005). ServiceMix Open Source Enterprise Service Bus (ESB) Now Avaliable; Open Source ESB Provides Agile, JBI-based Integration Solution for SOA. ZDA: Business Wire.

[13] Hanson, J. J. (2005). ServiceMix as an enterprise service bus. Pridobljeno 30. 6. 2014 s http://www.javaworld.com/article/2072293/soa/servicemix-as-an-enterprise-service-bus.html.

[14] Hohpe, G. (2003). Hub and Spoke [or] Zen and the Art of Message Broker Maintensnvr. Pridobljeno s http://www.enterpriseintegrationpatterns.com/ramblings/03_hubandspoke.html.

[15] Hohpe, G. in Woolf, B. (2004). Enterprise integration patterns: designing, building and deploying messaging solutions. ZDA: Pearson Education, Inc.

[16] Huang, Y. (2007). Scalable Web service-based XML message brokering across organizations. Ph. D. Work. ZDA: Indiana University.

[17] Iskandar, Basuki Y. (2000). Electronic Data Interchange (EDI) adoption in automobile supplires. ZDA: Vanderbilt University.

[18] Irriger, A. (2010). Apache ServiceMix. ZDA: Methods & Tools, Software Development Magazine – Programming, Software Testing, Projet Management, Jobs.

[19] Jackson, M. in Sloane, A. (2003). Modelling information and communication technology in business: A case study in electronic data interchange (EDI). Velika Britanija: Emerald Group Publishing, Limited, 9(1), str. 81–113.

[20] Kimberley, P. (1991). Electronic Data Intrechange. ZDA: McGraw-Hill, str. 6–34.

[21] Msdn.Microsoft.com (2014). Message Broker. Pridobljeno 23. 6. 2014 s http://msdn.microsoft.com/en-us/library/ff648849.aspx .

[22] PR Newswire (2003). RentPort™ Releases Its New ‚Message Broker Technology'. ZDA: PR Newswire Association LLC.

[23] Raney, M. in Wlater, C. K. (1992). Electronic data interchange: The warehouse and supplier interface. Velika Britanija: Emerald Group Publishing, Limited.

[24] Reenskaug, T. in Coplien, J. (2009). The DCI Architecture: A New Vision of Object-Oriented Programming. Pridobljeno 30. 6. 2014 s http://www.artima.com/articles/dci_vision.html.

[25] Reinshagen, D. (2001). XML messaging, Part 1; Write a simple XML message broker for custom XML messages. ZDA: Network World Inc.

[26] Rick, W. (2005). Open Source Rules, Routing Drools. ZDA: United Business Media LLC. 13(11), str. 25.

[27] Rouse, M. (2005). Message Broker. Pridobljeno 24. 6. 2014 s http://whatis.techtarget.com/definition/message-broker.

[28] ServiceMix (2014). ServiceMix. Pridobljeno s: http://servicemix.apache.org/ (26. 6. 2014).

[29] ServiceMix (2014). What is ServiceMix 4? Pridobljeno 18. 6. 2014 s http://servicemix.apache.org/docs/5.0.x/user/what-is-smx4.html.

[30] Smalhaiser, K. A. (1990). Productivity through Messaging. ZDA: Fortune, str. 155–164.

[31] Solr Wiki (2014). FAQ. Pridobljeno 30. 6. 2014 s https://wiki.apache.org/solr/FAQ#General.

[32] Warren, C. (2012). How Do Computers Talk to Each Other on the Interner? Pridobljeno 24. 6. 2014 s http://mashable.com/2012/10/17/tcpip-faq/.

[33] Weed, G. (2014). How Do Computers Communicate? Pridobljeno 26. 6. 2014 s http://www.ehow.com/how-does_4564764_computers-communicate.html.

▪

Daniel Kovačevič Rudolf je magistriral na področju družboslovne informatike februarja 2014. Zaposlen je v podjetju Actual I. T., d. d., v Kopru kot mladi raziskovalec na projektu Razvoj naprednih informacijskih sistemov na področju logistike in pristaniških dejavnosti. Že več let se ukvarja z raziskovanjem sodobne informacijske tehnologije. Tema njegovega magisterija je bilo računalniško podatkovno rudarjenje oziroma odkrivanje znanja v podatkih. Ima tudi znanja s področja programiranja v programskih jezikih VisalBasic in C#. Na področju informatike je že objavil znanstveni članek v Uporabni informatiki, letnik XXI, št. 1, 2013.

▪

Alen Vincelj je leta 1997 diplomiral na Fakulteti za elektrotehniko Univerze v Ljubljani. Kot programski inženir je v letih 1998 do 2000 delal v podjetju Emona Obala, d. d. Svojo pot je nadaljeval v podjetju Redox, d. o. o., kot sistemski inženir, programski inženir, skrbnik podatkovne baze, specialist za mrežo. Od leta 2007 je zaposlen v podjetju Actual I. T., d. d., kjer deluje kot sistemski inženir, skrbnik podatkovne baze sistemski inženir in razvijalec oziroma inženir programske opreme. Sodeloval je v več projektih, trenutno sodeluje v projektu Razvoj naprednih informacijskih sistemov na področju logistike in pristaniških dejavnosti. Na področju informacijske tehnologije deluje kot strokovni skrbnik strežnikov za Windows in Linux, dober poznavalec SOA, strokovni skrbnik podatkovne baze za SQL Server, PostgreeSQL, Oracle MySQL DB in kot strokovni razvijalec v več programskih jezikih. Je dober poznavalec strežniških operacijskih sistemov in strokovnjak za programsko opremo posrednikov sporočil, z izkušnjami v Microsoft Biztalk, Apache ServiceMix, JBoss Fuse ESB. Njegovo delo je večinoma usmerjeno na sistemsko administracijo, pri čemer aktivno vzdržuje operacijski sistem Port in skupnostni sistem Port za Luko Koper (TinO), kontejnerski terminal TOS Tideworks in integracijo z operacijskim sistemom Port v Luki Koper ter skupnostni sistem Port v Luki Ploče.

▪

Nina Novinec ima mednarodni certifikat PMI za projektno vodenje. Magistrirala je na področju evropskih projektov informacijske in komunikacijske tehnologije. Ima desetletne izkušnje pri projektnem vodenju tehnoloških projektov, sofinanciranih na nacionalni in evropski ravni. V podjetju Actual I. T., d. d., opravlja raziskovalne in projektne naloge poslovnih ved na razvojnem projektu Razvoj naprednih informacijskih sistemov na področju logistike in pristaniških dejavnosti. Je avtorica prispevkov in predavateljica (PMI Slovenija, FOV Znanost v organizacijskih vedah, SAP SMB Forum, nacionalne predstavitve, Slovensko združenje za projektni menedžment).