International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.2 (2014), pp.281-290
http://dx.doi.org/10.14257/ijmue.2014.9.2.28

Petri Net Based Semantic Web Service Composition

Azizbek Marakhimov[1], Jaegeol Yim[2*] and Jaehun Joo[3]

[1]*Cooperative Department of Techno Management, Dongguk University*
[2]*Department of Computer Engineering, Dongguk University*
[3]*Department of Information Management, Dongguk University*

{azizbek, yim, givej}@dongguk.ac.kr

Abstract

Automatic composition of Web services is a challenging task as independently developed Web services are not always compatible with each other. In order to resolve the heterogeneity between Web services and improve the quality of Web service composition, we propose a new approach based on semantics for Web service composition, which widely deploys Petri Nets and ontology. This paper proposes a framework for semantic Web service composition that builds Petri net model of interactions between Web services and builds ontology representing domain knowledge, then combines the Petri net models with ontology-based mediation.

Keywords: *Web Service, Composition, Semantic Mediation, Petri Net, Ontology*

1. Introduction

A fundamental aspect of Service Oriented Architecture (SOA) is to develop platform-independent and autonomous software components, Web services that are freely available in the distributed form over the Internet. Web services are self-contained modular software applications that are developed to provide unique functionality or interoperability between different existing applications [1]. However, the functionality of a single Web service is limited to address the actual needs of users. Therefore, Web service composition is a promising solution to combine the functionality of two or more Web services as a single service and fulfill the contemporary user requirements. Seamless and automated composition of Web services has considerable contribution in streamlining these processes. However, the main challenge in web service composition is the autonomy and heterogeneity of individual Web services developed by various service providers. Independently developed Web services are not always exactly compatible with each other, thus cannot be straightly composed together. Taking into account the use of mediators in previous studies, we consider this method as a primary way for implementing web service composition. We propose a framework for Petri net based semantic Web service composition, which addresses semantic context of Web services by identifying semantically similar objects and solving their schematic differences based on domain ontology.

2. Related studies

Previous studies cover Web services from wide variety of perspectives such as SOA implementation and testing, Web service ubiquity and security issues. Web services are core

* Corresponding Author

ISSN: 1975-0080 IJMUE

of SOA implementation as they provide interoperability of machine-to-machine interaction over the network [2]. Such architecture provides standard protocols for communication and service composition purposes [3]. In most cases, individual Web services might not be able to encompass the whole business process. Thus, composing multiple independent Web services that can interoperate to provide a complete business capability is reasonable [2, 3].

Web service mediation enables a service requester to connect to a relevant service provider regardless of the heterogeneities between them and works in a transparent way – neither of them needs to be aware of its existence [4]. The incompatibility of early languages as Web Service Flow Language (WSFL) and Web Services Choreography Interface (WSCI) has emerged second-generation languages, such as the Business Process Execution Language for Web Services (BPEL4WS, or BPEL). BPEL composition interacts with a Web services' subset to achieve a given a task and returns a mediated processes as a result of the composition [5].

Among recent studies, Song [6] provided a general framework for Web service composition based on semantics. The proposed approach implements Web service composition in two phases by (a) generating composition module for involved Web services and (b) executing the composition based on semantic Web services repository. Tan *et al.* [7] approached the Web service composition through building a mediator process based on Petri nets. Dao *et al.* [9] discussed a similar framework for semantic composition of Web services. Accordingly, the composition architecture contains translator, process generator, process evaluator, execution engine and Web service repository. Cash. Dao *et al.* [9] constructed data mapping table as shown below:

Table 1. Data Mapping table (Dao et al., 2008)

No.	Rules	No. of Services
1	*GoodsName → GoodsHS*	1
2	*GoodsName → Info*	1
5	*CountryName → Zone*	2
6	*CountryName → Info*	2
….	…….	…..

Such semantic interoperability issues of heterogeneous Web services are broadly addressed by Stollberg *et al.* [10] and Cardoso and Sheth [11]. Figure 1 illustrates a simple mediation at processes level provided by Stollberg [10]. In this example interaction between the system user and the service provider is performed as message exchange, where User sends message of type *date* but Service provider accepts a message type of *route*. In this case, the mediator applies the order inversion pattern in order to hold the first message from the User, and redirect it to the Service provider in a proper order.

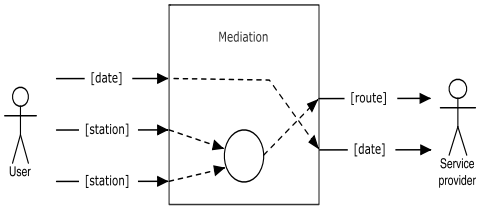


Figure 1. A process level mediation (Stollberg, 2006)

Although Web services and Web service composition are great solutions for solving business problems by providing high interoperability and reusability at lower costs, they have number of drawbacks that should be considered. One of the biggest issues is security. Bazarganigilani *et al.* [12] discussed security weaknesses of basic technologies of Web services such as XML, SOAP, UDDI, and WSDL. Specifically, WSDL documents that are available in open repositories can be utilized in planning and executing DOS attacks [12, 13]. Kalkhoran *et al.* [13] proposed a CPN based architecture of intrusion resistant Web services that combines a traditional security and fault tolerance techniques for providing viable security.

3. Web Service Mediation

Below, we provide a framework for Petri net based semantic Web service composition, which combines the Petri nets and ontology for solving semantic and functional mismatches during the composition process. For illustration, we use popular example two Web services to serve travelers' need; Web service A handles the air ticket reservation, while Web service B makes hotel room reservation. Both Web services are developed independently and thus have heterogeneous interfaces.

3.1. Proposed framework and algorithm

The proposed framework for semantic Web service composition is presented in Figure 2, which enables the composition of Web services taking into account semantic and functional heterogeneities by building a mediator. The framework uses BPEL descriptions of Web services, Semantic Annotations for WSDL and XML Schema for functional and semantic descriptions, Ontology Language for Services (OWL-S) and Petri Net formalism for building a mediator patterns. Referring to previous works [6, 7], we consider using BPEL descriptions of Web services as initial input to the composition framework with further transformation them into Colored Petri Nets (CPN). As Figure 2 illustrates, at the pre-composition phase BPL descriptions of Web services are transformed into CPNs. Business Process Execution Language (BPEL) is an XML language that supports process-oriented service composition [14]. When transformed from BPEL to CPN formalism, XML description of methods and states are changed into *transitions* and *places* respectively. CPN describes the functional behavior of Web services and indicates its input and output places. In our study, we consider the building the mediator based on the CPN models of involved Web services.

The proposed framework is functionally divided into four phases: initiation, pre-mediation, mediation execution and final composition. The framework externally communicates with service requester or *user*, *service provider*, and the *repository*. The mediation execution module performs actual composition using two key processes: semantic data mapping and mediator pattern selection. First, semantic data mapping is performed using domain ontology stored in *Ontology repository*. In this processes, the input and output interfaces of involved Web services are annotated with semantic terms and concepts provided in the ontology. This allows solving semantic mismatches and term duplications in the Web service interfaces. Second, the partial or full compatibility of Web services are checked and pre-defined mediator patterns are selected to mediate inputs and outputs of involved Web services. Mediator patterns are stored as Colored Petri Nets in the repository. Mediator pattern repository will store common types of mediators discussed in [9, 10]. Ontology repository will store domain ontology to specify the objects in data mapping. Consequently, this phase will consider both semantic data mapping and existing mediator patterns to analyze compatibility and select appropriate mediator types.

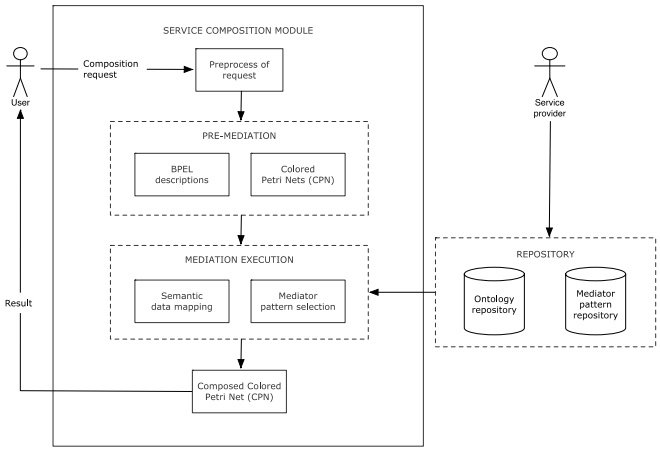


Figure 2. Proposed framework for semantic Web service composition

Further, we have developed an initial algorithm for inspecting the compatibility and selecting appropriate mediator type. This is an algorithm for executing the mediation within the provided framework, and consequently it will be refined and modified further. This algorithm checks data mapping rules and selects correct mediator types. The given algorithm checks each element in mapping rules, i.e. corresponding inputs and outputs of two involved Web services. As it can be seen, first `Source` (O_i) and `Target` (I_i) are checked. `Source` (O_i) is output of source Web service (Travel Agent) that goes as input to `Target` (I_i), an input to a target Web service (Hotel reservation). Then, the cardinality of `Target` (I_i) checked. $|\text{Target}(I_i)| = 1$ means that output of source Web service (Travel Agent) is required by only one input place in target Web service (Hotel reservation). Consequently, this mapping rule can be fully satisfied by the *store and forward mediator* (Pattern 1), which directly connects input and output places. In the case if `Source` (O_i) and `Target` (I_i) are not semantically same, *transformation mediator* (Pattern 2) is applied.

However, when cardinality of `Source` (O_i) or `Target` (I_i) is bigger than 1, *store and forward mediator* and *transformation mediator* cannot be applied. If $|\text{Source}(O_i)|$ is bigger than 1 then it means that output of source Web service (Travel Agent) sends two or more messages to only one input place in target Web service (Hotel reservation). Therefore, *merge mediator* is applied to satisfy this mapping rule. *Merge mediator* aggregates two output messages from source Web service (Travel Agent) and directs as one input message to target Web service (Hotel reservation) as shown in exhibit below.

Similarly, if $|\text{Target}(I_i)|$ is bigger than 1 then it means that output of source Web service (Travel Agent) is required by two or more input places in target Web service (Hotel reservation). In this case, *split mediator* is applied to divide a single output message from of source Web service (Travel Agent) and direct as two input messages to target Web service (Hotel reservation).

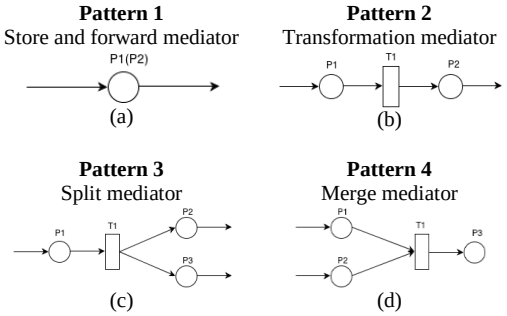

```
Input: Data mapping rules(mr_i)of Web service A and Web
       service B
Output: Composed Web service interfaces

If Source (O_i)=Target (I_i) AND |Target (I_i)|=1
Then call pattern_1 from repository;
Else if Source (O_i)! = Target (I_i) AND |Target (I_i)| = 1
Then call pattern_2 from repository;
Else if Source (O_i) = Target (I_i) AND |Source (O_i)| > 1
     Then call pattern_4 from repository;
Else if Source (O_i) = Target (I_i) AND |Target (I_i)| > 1
     Then call pattern_3 from repository;
Else if Source (O_i)! = Target (I_i) AND |Source (O_i)| > 1
     Then call pattern_4 from repository;
Else if Source (O_i)! = Target (I_i) AND |Target (I_i)| > 1
     Then call pattern_3 from repository;
```

O_i - *Web service output* $|Source\ (O_i)|$ - *number of output elements*

I_i - *Web service input* $|Target\ (I_i)|$ - *number of input elements*

The Petri net models of predefined mediator patterns are stored in mediator repository as shown in Figure 2. The patterns are provided below:



3.2. Travel Ontology

One of the noticeable limitations of existing approaches in Web service composition is that checking the compatibility of involved services requires a manual effort. To solve this problem and semantically enrich the mapping rules, we proposed ontology for a travel domain. We extended ontology provided by Park [15] to address the semantic issues in our study. We modified it by shortening its structure and adding several new concepts. An abstract view to ontology is provided in Figure 3. We developed our ontology using OntoGen software tool. We explicitly described meanings of concepts related to travel and tourism activities and properties (relationships) between those concepts by Web Ontology Language (OWL) and Resource Description Framework (RDF). The ontology describes 10 main concepts and 48 subsequent sub-concepts related with travel domain. During annotation process all input and output messages of Web service A and Web service B will be described using the concepts of ontology and new semantic data mapping rules will be constructed. This will enable us to automate the process of checking Web service compatibility for building mediators.

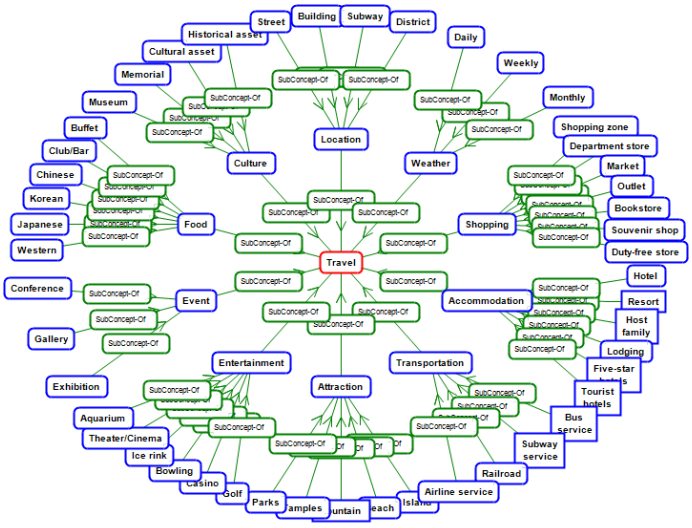


Figure 3. Proposed ontology for travel domain

3.3. Mediation

The objective of the mediation is to compose two Web services into a single process to serve travelers' need. The interaction of involved Web services consists of several steps of message exchange between them. When Web service A (TravelAgent) is initiated, it invokes Web Service B (HotelReservation) by sending message *ReservationReq*. Further, in *GetHotelReservation* process TravelAgent receives message with *AvailabilityConfirmation*, and instantly responds with message including *ArrivalDate*, *DepartureDate*, and *Roomtype*. Finally, TravelAgent receives message including *ReservationID*, and responds by sending *UserID*. On the other hand, HotelReservation is initiated with identical *ReservationReq* message, and invokes TravelAgent by sending a message with *AvailabilityConfirmation* and *ReservationID*. Then, TravelAgent accepts a message containing *CheckinDate*, *CheckoutDate*, *RoomType*, and *CustomerID*. Finally, it replies with message containing *Confirmation* and arranges the reservation (Figure 4).

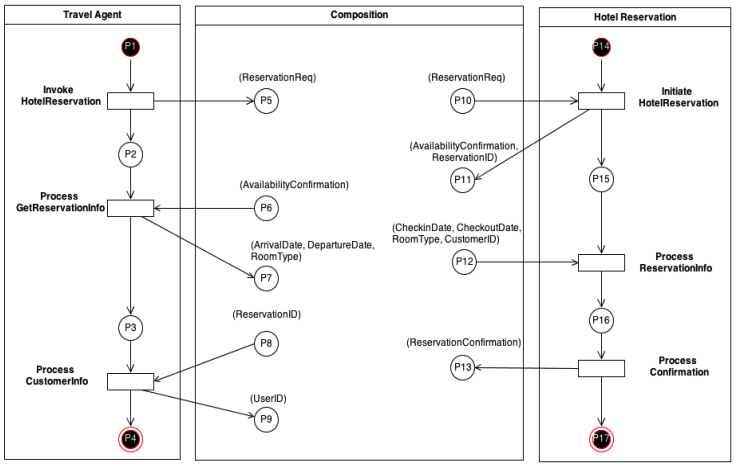


Figure 4. Petri Net model of involved Web service

There are number of mismatches in inputs and outputs of these Web services. They involve naming conflicts and data conflicts. Below, the mapping rules are identified and domain ontology is applied to solve these conflicts. These two Web services are from different providers that use different terminology and data formats. The need for mediation is to convert messages from Web service A to the format required by Web service B. Figure 5 shows the input and output messages of both Web services. The problem is that providers used different terminology to specify semantically similar objects. This makes interfaces of two services not compatible with each other.

Figure 5 shows that Web service A (TravelAgent) sends *ReservationReq*, *ArrivalDate*, *DepartureDate*, *RoomType*, and *UserID*, while Web service B (HotelReservation) accepts *ReservationReq*, *CheckInDate*, *CheckOutDate*, *Roomtype*, and *CustomerID*. In turn, Web service (HotelReservation) sends *AvailabilityConfirmation*, *ReservationID*, and *ReservationConfirmation*, while Web service A (TravelAgent) accepts *AvailabilityConfirmation* and *ReservationID.*

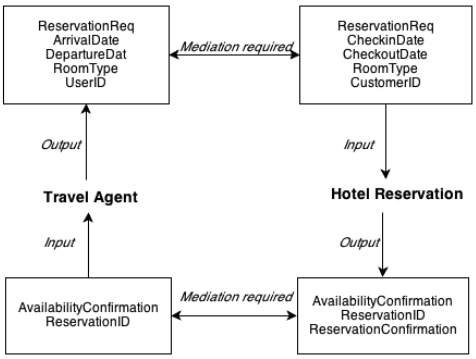


Figure 5. Interface mismatches of involved Web services

Table 2 shows the data level heterogeneities that exist between involved Web services. Unless these heterogeneities are solved, the Web services are not compatible and eligible for composition. Data level conflicts or mismatch in this scenario occurred due to use of different terminologies. The situation that should be solved is to match outputs of Web service A to the inputs of Web service B, and match outputs of Web service B to the inputs of Web service A.

Table 2. Semantic conflicts in mapping rules

Mapping rules	
Output of TravelAgent	Input of HotelReservation
ReservationReq	ReservationReq
ArrivalDate	CheckinDate
DepartureDate	CheckoutDate
RoomType	RoomType
UserID	Customer ID
Identified conflicts	
ArrivalDate vs. CheckinDate – naming conflict	
DepartureDate vs. CheckoutDate – naming conflict	
UserID vs. CustomerID – naming conflict	

Conventional approach of solving the situation involves manually mapping inputs and outputs of Web services. However, this approach will limit the solution only to two selected Web services. In contrast, we have proposed wider system framework above that includes repository of domain ontologies and mediator patterns. Such framework will allow using other Web services or other ontologies with less manual work. However, to execute the mediation the involved Web services should be modeled into mapping rules that will represent their corresponding inputs and outputs. Web service requires m inputs *{I1, I2, Im}*, and generates n outputs *{O1, O2, On}*. Thus the Web service can be represented with n rules of showing the correspondence of inputs and outputs as *I1 & I2 & … Im → O1*. In a similar way, we mapped inputs and outputs of involved Web services as shown in Table 2. Each mapping rule is associated with semantic concept of the Travel ontology. By this way, we annotate all inputs and outputs of Web services with semantic concepts of the ontology for solving the semantic mismatches provided in Figure 5.

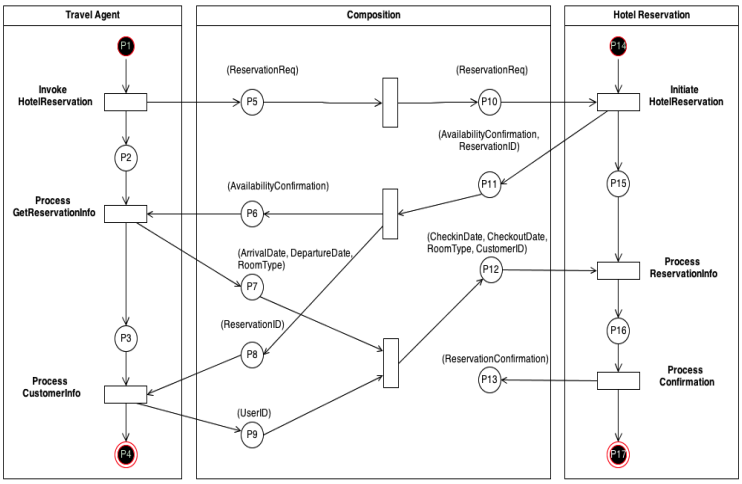


Figure 6. Mediated Petri net model of involved Web services

Figure 6 demonstrates the mediation of the TravelAgent and HotelReservation Web services. The mediator patterns that connect two Web services are applied based on the logic provided above. Accordingly, the case of place P1 and P10 satisfies the first rule (If Source (O_i) = Target (I_i) AND Target mr_i = 1 Then call $pattern_1$ from repository). The input and output of source and target Web services are same as they both exchange message with *ReservationReq* data. Thus, they are mediated using store/forward pattern. In the case of places P6, P8 and P11 the following rule is satisfied: (Else if Source (O_i) = Target (I_i) AND Source mr_i > 1 Then call $pattern_3$ from repository;). HotelReservation responds with *AvailabilityConfirmation* and *ReservationID*, while TravelAgent waits for only a message with *AvailabilityConfirmation.* Therefore, the split pattern is applied, which sends *AvailabilityConfirmation* to P6 and stores *ReservationID* until it is requested by P8. And finally, the case of places P7, P9 and P12 satisfies the following rule: (Else if Source (O_i)! = Target (I_i) AND Source mr_i > 1 Then call $pattern_3$ from repository). In this case, P12 accepts composed messages from two different places. In addition, input and output messages are not semantically equal to each other. Therefore, a merge pattern is applied to mediate the places. P12 accepts a message containing *CheckinDate, CheckoutDate, RoomType* and *CustomerID* in order to arrange and confirm the reservation. P7 replies with message containing *ArrivalDate, DepartureDate,* and *RoomType.* Further, P9 replies with message

containing *UserID.* In this case, the semantic conflicts shown in Table 3 are proposed to be resolved by the use of domain ontology. The merge pattern composes messages from P8 - P9 and redirects to P12, which in turn finalizes the reservation with message containing *ReservationConfirmation.*

4. Conclusion

In this paper, we described a semantic framework based on Petri Net and ontology for building mediator. The proposed framework involves designing Petri net models of interactions between Web services and building ontology for explicitly representing domain knowledge. Further, Web service message interfaces are semantically annotated using domain ontology for executing semantic mediation. Future work includes conducting a reliable Quality of Service (QoS) in key mediation aspects.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2011-0006942) and by 'Development of Global Culture and Tourism IPTV Broadcasting Station' Project through the Industrial Infrastructure Program for Fundamental Technologies funded by the Ministry of Knowledge Economy (10037393).

References

[1] G. Alonso, H. Casati, F. Kuno and V. Machiraju, "Web Services: Concepts, Architectures and Applications", Berlin: Springer-Verlag, **(2003)**.
[2] P. Kalamegan and Z. Godandapani, "A Survey on Testing SOA Built using Web Services", International Journal of Software Engineering and Its Applications, vol. 6, no. 4, **(2012)**, pp. 91-104.
[3] S. Sultana, R. Karim, R. Shahriyar, M. Akbar and S. I. Ahamed, "Ubiquitous Secretary: A Ubiquitous Computing Application Based on Web Services Architecture", International Journal of Multimedia and Ubiquitous Engineering, vol. 4, no. 4, **(2009)**, pp. 53-70.
[4] Q. Yu, X. Liu, A. Bouguettaya and B. Medjahed, "Deploying and Managing Web Services: Issues, Solutions, and Directions", International Journal on Very Large Data Bases (VLDB Journal), vol. 17, **(2008)**, pp. 537–572 .
[5] S. Clarke, "Current Solutions for Web Service Composition", IEEE Internet Computing, **(2004)**, pp. 51-59.
[6] G. Song, "A New Approach for Web Service Composition Based on Semantic", IEEE, **(2010)**.
[7] W. Tan, Y. Fan and M. Zhou, "A Petri Net-based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language", IEEE Transaction of Automation Science and Engineering, vol. 6, no. 1, **(2009)**, pp. 94-106.
[8] A. Iglesias, "Pure Petri Nets for Software Verification and Validation of Semantic Web Services in Graphical Worlds", International Journal of Future Generation Communication and Networking, vol. 3, no. 1, **(2010)**, pp. 33-46.
[9] C. Dao, C. Xu and C. Chunlai, "Semantic and Rules Based Upon Mediator Dynamic Web Service Composition in Logistics Information Application", IEEE International Conference on Management of Innovation and Technology, **(2008)**, pp. 532-536.
[10] M. Stollberg, E. Cimpian, A. Mocan and D. Fensel, "A Semantic Web Mediation Architecture", In Proceedings of the 1st Canadian Semantic Web Working Symposium, **(2006)**.
[11] J. Cardoso and A. Sheth, "Semantic E-Workflow Composition. LSDIS Lab", University of Georgia, **(2002)**.
[12] M. Bazarganigilani, B. Fridey and A. Syed, "Web Service Intrusion Using XML Similarity Classification and WSDL Description", International Journal of u- and e- Service, Science and Technology, vol. 4, no. 3, **(2011)**, pp. 61-72.
[13] Z. A. Kalkhoran and M. A. Azgomi, "A Multi-Layer Architecture for Intrusion Tolerant Web Services", International Journal of u- and e- Service, Science and Technology, **(2008)**, pp. 73-80.
[14] F. Curbera, "The Next Step in Web Services", Communications of ACM, vol. 46, no. 10, **(2003)**, pp. 29–34.
[15] H. Park, A. Yoon and H. Kwon, "Task Model and task Ontology for Intelligent Tourist Information Service", International Journal of u- and e-Service and Technology, vol. 5, no. 2, **(2012)**.

Authors

Jaegeol Yim

He received the M.S. and Ph.D. degrees in Computer Science from the University of Illinois at Chicago, in 1987 and 1990, respectively. He is a Professor in the Department of Computer Science at Dongguk University at Gyeongju Korea. His current research interests include Petri net theory and its applications, Location Based Service, AI systems, and multimedia systems. He has published more than 50 journal papers, 100 conference papers (mostly written in Korean Language), and several undergraduate textbooks.

Jaehun Joo

He is a professor of Department of Information Management at Dongguk University – Gyeongju Campus. He received his Ph.D. from Pusan National University, South Korea. His areas of research interest are e-commerce, collective intelligence, Semantic Web/ontology, business ecosystems, and knowledge management. He published many papers in International Journal of Human-Computer Studies, Journal of Sustainable Tourism, Decision Support Systems, Information Systems Management, International Journal of Industrial Engineering, Expert Systems with Applications, Journal of Computer Information Systems, and etc.

Azizbek Marakhimov

He is a Ph.D. candidate of Cooperative Department of Techno Management at Dongguk University – Gyeongju Campus. He received his Bachelor of Science degree in Business Computing from University of Westminster, UK, and Master of Business Administration degree from Dongguk University – Gyeongju Campus, South Korea. His areas of research interest are business ecosystems and strategic management, Web services, Semantic Web/ontology.