World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:3, 2016

A State-Of-The-Art Review on Web Services Adaptation

M. Velasco, D. While, P. Raju, J. Krasniewicz, A. Amini, L. Hernandez-Munoz

***Abstract*—**Web service adaptation involves the creation of adapters that solve Web services incompatibilities known as mismatches. Since the importance of Web services adaptation is increasing because of the frequent implementation and use of online Web services, this paper presents a literature review of web services to investigate the main methods of adaptation, their theoretical underpinnings and the metrics used to measure adapters performance. Eighteen publications were reviewed independently by two researchers. We found that adaptation techniques are needed to solve different types of problems that may arise due to incompatibilities in Web service interfaces, including protocols, messages, data and semantics that affect the interoperability of the services. Although adapters are non-invasive methods that can improve Web services interoperability and there are current approaches for service adaptation; there is, however, not yet one solution that fits all types of mismatches. Our results also show that only a few research projects incorporate theoretical frameworks and that metrics to measure adapters' performance are very limited. We conclude that further research on software adaptation should improve current adaptation methods in different layers of the service interoperability and that an adaptation theoretical framework that incorporates a theoretical underpinning and measures of qualitative and quantitative performance needs to be created.

***Keywords*—**Web services adapters, software adaptation, web services mismatches, web services interoperability.

I. Introduction

A web service is a piece of software designed to carry out interoperable machine-to-machine interaction in a network [1]. The service is "identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then, interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols" *ibid*. Hence, Web services are distributed information systems that are an essential part of the Internet since they enable software programs to communicate seamlessly and efficiently, delivering platform independent systems with high flexibility [2] and allowing the integration of heterogeneous applications using open standards such as XML and Service-Oriented Architectures (SOA) [3], [4].

Web services can be systems providing diverse functionalities such as payment services, maps, on-line travel and reservations [5]. Furthermore, Web services can wrap active applications as services; consequently developers can use Web services to enhance interoperability through standard protocols such as SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) and SOA architectures [6]. Nevertheless, despite advances in communication protocols and software architectures there are still interoperability issues among Web services. Interoperability can be defined as the capability among two or more systems, networks, devices, components or applications to exchange and use information across and within them [7]. Interoperability issues are reflected in Web services mismatches. Mismatches are caused due to differences in the specifications, interfaces, protocols, behavior or formats of the interchanged messages, causing a disruption in the communication [8]. For example, a behavioral mismatch appears when two services keep waiting for each other to send a message (deadlock mismatch). Mismatches can be resolved using for example, schema mapping and transformation tools implemented in a Web service adapter that provides a solution to the interoperability issue. This is known as software adaptation [9].

An adapter is a Web service that sits between two components and compensates for the differences between their interfaces (i.e., their specifications, protocols or behaviors) [10]. Benatallah et al. [11] identified that the development of adapters can be based on mismatch patterns, which are design patterns used to capture differences among services (interfaces and protocols). However, [12] states that the identification of these patterns is not sufficient, thus gaps exist in the creation of the adapter that solves any type of mismatch.

The role of an adapter, which is considered a mediator service, makes the interaction seamless [13]. The use of adapters can involve performing activities like storing and receiving messages, transforming message data and invoking service operations [3]. The adapter code can be generated semi-automatically or automatically once the mismatch pattern has been identified [13].

This paper provides results of a literature review aimed to investigate what are the main adaptation methods published in the academic literature and the type of theoretical frameworks underpinning software adaptation research. This paper may contribute to identify gaps in the development of algorithms that can improve interoperability of Web services. The

M. Velasco, A. Amini and L. Hernandez-Munoz are with the Enterprise Systems Lab at Birmingham City University, Millennium Point, Birmingham, B47XG, United Kingdom (Phone: +44 121 331 7542; e-mail: Margarita.Velasco@bcu.ac.uk, Ardavan.Amini@bcu.ac.uk, Luis.Hernandez-Munoz@bcu.ac.uk).

D. While and J. Krasniewicz are with the Birmingham City University, Millennium Point, Birmingham, B47XG, United Kingdom (e-mail: David.While@bcu.ac.uk; Jan.Kransniewicz@bcu.ac.uk).

P. Raju is with the KBE Lab at Birmingham City University, Millennium Point, Birmingham, B47XG, United Kingdom (e-mail: Path. Raju@bcu.ac.uk).

following section explains the methodology of this review. Later, the results of the literature review are shown and discussed. Finally, further work is outlined and conclusions are provided.

II. Methodology

A. Sources of Information

A search for publications was performed with query strings in the following sources: Web of Knowledge/Web of Science, ACM Digital Library, IEEE xplore and Google Scholar. Furthermore, we explored the following publications manually: IEEE Transactions on Software Engineering, the International Journal of Web & Semantic Technology (IJWesT), the International Journal of Web Services Research (IJWSR) and The Journal of Systems & Software.

B. Entry Strings

Our query strings included "web services adaptation", "software adaptation", "web service adapters", "web services composition", "service composition adaptation" and "web service mismatches". Since concepts such as Web services and adaptation or adapters or mismatches must appear on the papers, we linked the concepts with AND and OR operators.

C. Inclusion and Exclusion Criteria

Since hundreds of thousands of publications were retrieved from our queries, our selection strategy involved three phases. First, two researchers selected paper abstracts including: 1) publications from 1995 to 2010 with the highest number of citations and related to software adaptation (100 abstracts); and 2) the most relevant abstracts on software adaptation from 2010 to 2015 (i.e., highly or lowly cited abstracts proposing a strategy for software adaptation) (50 abstracts). Second, papers written by similar authors which did not provided extra information (47 papers) and papers not available for download (33 papers) were eliminated. Third, from the 70 abstracts remaining, the two researchers agreed to read in full the most relevant (18 papers) in order to answer our research question.

D. Data Extraction

The 18 identified papers were analyzed using NVivo v10. Thematic analysis was carried out to identify pre-selected themes and themes emerging from the papers. Pre-selected themes included: the need of software adaptation; type of adaptation methods, main characteristics and limitations; and theoretical frameworks used in the adaptation approach. While, emerging themes found in the papers included: software adaptation layers, adapter evaluation metrics and opportunities for further research in software adaptation.

III. Results

A. The Need of Software Adaptation

Software adaptation is needed since the number of Web services is growing very rapidly, thus the issue of Web service adaption has become of special importance in research and practice [14]. In Web service adaptation the functional description of a Web service includes their interface, protocol, data and behaviour [14]. These elements are used in SOA architectures to create service compositions as mechanisms of seamless integration, rapid deployment, reuse of heterogeneous computing resources and service coordination across organizations [8]. Service composition includes systems of Web services that fulfill basic communication requirements or larger networks of Web services that accomplish more complex tasks [9]. Therefore, Web services have to be designed with reusable interfaces that allow access to their functionalities. The interfaces (i.e., messages, functionalities, protocols and dependences) of Web services are stored in an XML file known as WSDL (Web Service Description Language) [15]. Different interoperability levels can be identified in WSDLs. For example, signature, protocol, quality of service and semantics [9]. Brogi and Popescu [16] points out that service adaptation may be undertaken at various levels of the interoperability services stack. For example, *signature-based adaptation* aims to overcome mismatches due to syntactic differences among the exchanged messages, for instance, the order in which the part of a message are delivered; *ontology-based adaptation* helps improving semantic mismatches among the messages, for instance, messages related to different ontology concepts; and *behavior-based adaptation* helps with the integration of services that have mismatches in their communicating protocols, for instance, the order in which messages are exchanged. The authors, nevertheless, highlighted that Web service adaptation is an active area of research due to the existence of only limited solutions to current mismatches needs. In practice, Business Process Execution Language (BPEL) is a compositions language industry standard that allows the executions and communications of processes across organizations and enables invoking web service operations in a specific order [17]. BPEL is used to implement and test software adaptation methods.

Since composition of Web services is rarely achieved faultlessly because mismatches may occur at the different interoperability levels, research has been undertaken to create methods (or algorithms) aimed to solve those issues. These software adaptation processes include methods containing mismatching interfaces that generate mediating adapter services [9]. Table I shows different types of adaptation approaches found in our reviewed papers.

B. Software Adaptation Methods

Early research in software adaptation was carried out by [10]. They used finite state machine algorithms to model the adaptation process and pointed out that protocols are responsible for the relationship among Web services and their messages and that adapters can be formally defined to eliminate the difference in their protocol interfaces.

TABLE I
MAIN ADAPTATION METHODS FOUND IN THE REVIEWED PAPERS

Publication	Type of adaptation	Main characteristics	Limitations
Kongdenfha et. al. (2009) [3]	Templates	Fast adaptation of Web services through the combination of mismatch patterns and aspect-oriented approach.	Other higher level of service specifications such as policies are needed.
Seguel et al. (2009) [8]	Interaction Analysis Matrix	The method seemed to make service adaptation more efficient at both design and at run-time.	To test the prototype to more real-life case studies and to build an adaptor for more than two BPEL protocols is needed.
Yellin and Strom (1997) [10]	Finite state machine	Adapters were automatically generated from a high-level description.	Specifications were manually written and they required the designer to fully understand the details of the components involved.
Benatallah et al. (2005) [11]	Mismatch patterns	Identified and classified different kinds of adaptation requirements to characterize the problem of adaptation of web services.	The language design and the code generation was not finalized.
Nezhad et al. (2010) [14]	Depth-based approach, Iterative reference-based approach	Their approach significantly improved the effectiveness and the accuracy of matching results. Reduced the efforts for adapter development in Web services.	To extend the algorithms to identify other classes of mismatches is needed and interface and protocol matching approaches can be incorporated.
Dumas et al. (2006) [15]	Finite state machine	Facilitated message interception, transformation and buffering to approve the adaptation logic.	Techniques to semi-automatically deduce possible links between both provided and required interfaces need to be developed.
Brogi and Popescu (2006) [16]	Service execution trees	Their methodology seems to be successfully employed to generate replaceability adapters.	Behavioral mismatches among numerous interacting BPEL processes need to be solved.
Tan et al. (2009) [17]	Workflow net	Analyzed web service compatibility. Automatically composed services.	The approach should be validated in real-life case studies.
Shan (2010) [18]	Patterns	Message and control flow adaptation were integrated. Adapters could be created on the fly for matching parts of two services.	Verification methods need to be implemented.
Nezhad et al. (2007) [19]	Mismatch Tree	Their solution seemed to simplify adapter development.	Real-world experiments need to be performed and the overhead of adapter generation needs to be measured.
Wang et al. (2008) [20]	Finite state machine	They introduced two metrics: ability and intention trust.	Techniques to statically detect and reconcile all differences between two service interfaces need to be designed.
La and Kim (2011) [21]	Design patterns	Their middleware-level runtime module ran in an autonomous manner. Technical details for static and dynamic adapters were defined.	Need to assess the effectiveness and correctness of the adapters.
Lin et al. (2011) [22]	Pushdown System Model	An adaptor is verified at the same time of being generated, saving time and cost.	Real-world experiments with large-scale systems are needed in order to evaluate real-time issues.
Mateescu et al. (2012) [23]	On-the-fly reduction	Adapter generation seems to be automated.	Evaluation of algorithm need to be evaluated in different contexts.
Lahmar and Belaid (2013) [24]	Fine-grained template	Adapters provide an extra-functional behavior compared to the functional behavior. Adapters can be generated in few milliseconds.	Validation and evaluations of the generated adapters is required.
Velasco-Elizondo et al. (2013) [25]	Architectural style	They seemed to automate tasks by combining architectural modeling, model-generation and utility analysis.	Consider real-life situations in different domains.
Alférez et al. (2014) [26]	Variability models	Variability models were used to facilitate the reasoning of dynamic adaptations	Verification operations at runtime need to be implemented to avoid inconsistent service recompositions in the open world.
Taher et al. (2011) [27]	Complex Event Processing	Automatically generated adapters capable of intercepting incoming messages.	Adapters on real services need to be evaluated. Tools to assist designers to generate adapters need to be created.

Kongdenfha et al. [3] proposed a tool and a framework to support service adaptation based on the aspect oriented programming approach. Their framework includes a classification of different types of mismatches at interface and protocol levels. They proposed two ways of using adapters to solve the mismatches, developing a new Web service (stand-alone) or modifying one of the Web services to make it compatible with the other. They used IBM WebSphere Integration Developer tool for schema mapping functionalities to instantiate the templates and concluded that the second approach was better to solve the mismatch.

Benatallah et al. [11] proposed the use of mismatch patterns for the development of replaceability adapters. Nevertheless, their approach is limited to the designer providing a template of parameters, but complex behavioral mismatches were not addressed.

Nezhad et al. [14] proposed a method to solve mismatches semi-automatically at the service interface level. Their method included a protocol-aware approach for Web service interface. They used Java and Eclipse as the Interface Development Environment; OntoBuilder library was used to implement the static interface matching approach. Their results show that their method enhanced the quality of matching among services.

Brogi and Popescu [16] proposed a methodology to solve mismatches that can appear at service behavioral levels between two BPEL processes. Their algorithm generates adapters automatically based on service execution trees. A BPEL process adapter is built in order to allow the two processes continue interoperating. However, [18] argues that this type of adaptation seems to be too extensive because they generate a service execution tree for every possible path and because of the inconvenience created when buffering every single activity.

Tan et al. [17] focused on behavior mismatches. They used BPEL to facilitate the logic of the Web services. First, they evaluated the compatibility of two Web services, then they transformed them into a service workflow net and finally they generated an adaptor between the two Web services. While, Nezhad et al. [19] presented a model for service adaptation based on adapter protocol and interface mappings. SOA is used as the platform to solve mismatches at both interface and protocol level. Adapters are generated semi-automatically by identifying parameters of mapping functions. Their algorithms were developed using IBM WID (WebSphere Integration Developer), where XML was used to represent rules and Eclipse/Java for the implementation of the algorithm.

Wang et al. [20] developed an adaptation machine for service interface adaptation at runtime. The adaptation machine was placed between two services to resolve mismatches by capturing, transforming and sending messages with specific mapping rules. They used Finite State Machines to represent the behavioral service interface. Moreover, they introduced a tool called Megine (Service Mediation Engine) to manage a repository of mapping rules that could capture interfaces of both WSDL descriptions and finite state machines.

La and Kim [21] defined a framework for service adaptation (SAF) and also designed both, static and dynamic adapters. They considered that there are two main problems when Web services interoperate: mismatch at development time and fault at run time. Furthermore, they derived mismatches, faults, their causes and adapters by defining two causal-effect relationships. The first one was including mismatches and static adapters and the second one was involving faults and dynamic adapters, both including causes. For the development of static adapters, they defined practical instructions and specified step-wise algorithms for the dynamic adaptation.

Lin et al. [22] generated non-regular behavioral adapters specifying the behavior of service interfaces. Behavior interfaces were represented by Interface Automata and a pushdown system model was used to represent adapters. Pushdown systems were used to effectively detect that messages were successfully sent and received through the adapter. To support the automation of service adaptation, a prototype tool was implemented. Their tool was developed in C programming language and was able to read the behavior interfaces of services that were described in an input file.

Mateescu et al. [23] proposed a model to support service adaptation by removing incorrect behaviors in complex adaptation scenarios. They focused on service interfaces rather than service implementation. A tool that generated the adapter was developed by considering the service protocol and interfaces with adaptation contract and value passing. While, [24] used an adaptation approach that was dynamic structural based on the adaptation of applications through the introduction of an adapter in the description of the service to modify its functional behavior. The adapter was defined through a fine-grained template. In addition, a proxy component was used to compose the adaptive logic of the adapter. They evaluated their approach by calculating the time taken to generate the proxy.

Velasco-Elizondo et al. [25] presented an automatic detection and solution of data mismatches. They considered previous categorization of mismatches and extend them with their data mismatch resolution approach. They formalized their mismatches as rules to make them work between architectural components. Furthermore, an architectural style called SCORE was used for representing end-users composition and for mismatch detection. They used the constrains and descriptions of SCORE to solve data mismatches in a composition through a Mismatch Detection Engine.

Alférez et al. [26] presented a framework to support dynamic adaptation compositions based on variability models at runtime. Their model was used at the context level when a mismatch arises. The composition model changed according to the variability model featuring activated or deactivated states. Changes could be seen in the composition by including or deleting fragments of BPEL composition schema at runtime. Finally, [27] used complex event processing to automatically generate adapters, but their approach would need to be validated on real web services.

C. Layers of Service Interoperability

Three main layers were used in our reviewed papers. 1) Table II shows that 13 papers reported work in the interface layer, which comprises the set of operations the services support [19]; 2) eight papers carried out research in the protocol layer, which comprises the order in which operations should be invoked to achieve a successful interaction [19]; and 3) seven papers involved work on the behavioral layer, which comprises the interactions among the protocols [23]. These results show that the interface layer was the most popular. Hence, our review seems to show that there are many opportunities to carry out further investigations in other interoperability layers, including for example the policies and nonfunctional properties layer.

D. Theoretical Frameworks and Evaluation Metrics Used in Software Adaptation

Table III shows that the most popular theoretical framework underpinning adaptation approaches was finite-state machines (three papers) [10], [14], [19]. While, other 12 papers used different theoretical models including mismatch patters [3], structured process models [8], CORBA [10], concurrency theory [15], Colored Petri nets [17], pattern based analysis [18], abstract state machines [20], life cycle model [21], pushdown automata [22], model driven [23], architectural constructs [25], dynamic adaptations [26] and model transformation [27]. While three papers did not report the use of a theoretical framework [11], [16], [24]. Because the wide variety of frameworks used in our reviewed papers, results may reflect the lack of a standardized theory that can provide the grounds needed to support software adaptation research and perhaps a lack of consensus across different authors. Even worst, because there are research projects not even grounded

in any theoretical underpinning, future work should involve the creation of an integrated framework that can help improving web services interoperability grounded in a solid theoretical perspective, which in turn may produce more effective adaptation outcomes. Table III also shows that while theoretical frameworks were somewhat used in our reviewed papers, not many research reported the type of metrics used to evaluate their adaptation methods (11 papers not reported their metrics, though two reported using synthetic and real-life cases). Nevertheless, there were interesting metrics such as [3] that include the qualitative comparison of stand-alone and aspect-oriented characteristics, complemented by a quantitative evaluation based on the adoption of the CK metrics (e.g., Line Of Code (LOC) and Number Of Classes (NOC)). While other research highlighted the evaluation of adapters effectiveness and the usage of protocol-aware approaches [14], adapters complexity [18], protocol correctness [23], structural adaptation time [24], efficiency and scalability [25], and generation efficiency and complexity reduction [26]. This limited number of instruments to quantify the benefits of adaptation processes may reflect the need to create further mechanisms that help quantifying mismatch identification, adapters creation and performance.

TABLE II
LAYERS OF SERVICE INTEROPERABILITY USED IN THE REVIEWED PAPERS

Publication	Interface layer	Protocol layer	Behavioral layer
Kongdenfha et. al. [3]	√	√	
Seguel et al. [8]	√		
Yellin and Strom [10]	√	√	
Benatallah et al. [11]	√	√	
Nezhad et al. [14]	√	√	
Dumas et al. [15]	√		
Brogi and Popescu [16]			√
Tan et al. [17]			√
Shan [18]	√	√	
Nezhad et al. [19]	√		
Wang et al. [20]	√		
La and Kim [21]	√	√	√
Lin et al. [22]	√		√
Mateescu et al. [23]	√	√	
Lahmar and Belaid [24]			√
Velasco-Elizondo et al. [25]			√
Alférez et al. [26]			√
Taher et al. [27]	√	√	

E. Opportunities for Further Research

Tables I-II show that there many opportunities for further research. For example, Table I shows that work is needed in the design of adaptation methods to create adapters semi-automatically and automatically [10] and that behavioral mismatches have to be identified [16]. Table II shows that the identification of mismatches and the creation of adapters in different layers need to be investigated, including the interface, protocol and behavior layers and the service specifications components that include policies [3], [14]. Furthermore, tools to assist designers to generate appropriate adapters need to be created [27]. Very importantly, real-life experiments in different domains have to be carried out in large-scale systems to validate and evaluate the detection of mismatches and the creation of adapters [8], [17], [19], [22]-[28]. Finally, Tables I and III illustrate that future research needs to be grounded in appropriated theoretical frameworks to create appropriate, reliable and standardized metrics that help validating, evaluating and quantifying the creation and performance of adapters and their interoperability with other type of software services.

TABLE III
SOFTWARE ADAPTATION THEORETICAL FRAMEWORKS AND EVALUATION METRICS

Publication	Theoretical Framework	Evaluation Metrics
Kongdenfha et. al. [3]	Mismatch patterns framework and aspect-oriented approach.	Qualitative-comparison of the stand-alone and aspect-oriented. Quantitative evaluation is based on the adoption of the CK metrics (Line Of Code (LOC) and Number Of Classes (NOC).
Seguel et al. [8]	Structured process models	Not reported.
Yellin and Strom [10]	Corba model and finite-state machine model.	Not reported.
Benatallah et al. [11]	Not reported	Not reported
Nezhad et al. [14]	Finite state machines (FSM) as the modeling formalism for business protocols.	They validated the effectiveness and usage of protocol-aware approaches.
Dumas et al. [15]	Concurrency theory.	Not reported
Brogi and Popescu [16]	Not reported.	Not reported.
Tan et al. [17]	Colored Petri nets (CPNs) as a formal model.	Not reported (validations of their approach through a real-life case).
Shan [18]	Pattern-based analysis	They evaluate the complexity of the created adapters.
Nezhad et al. [19]	Finite state machines.	Not reported (Used of synthetic and real-world scenarios).
Wang et al. [20]	Abstract State Machines (ASMs).	Not reported.
La and Kim [21]	Life-cycle model.	Not reported.
Lin et al. [22]	Pushdown automata model.	Not reported.
Mateescu et al. [23]	Model- driven approach.	Use a 'CADP' toolbox, which is used to verify the correctness of the contract protocol.
Lahmar and Belaid [24]	Not reported.	They evaluated the structural adaptation by calculating the time required for the generation of the byte code.
Velasco-Elizondo et al. [25]	Model architectural constructs in Alloy and architectural specification.	Efficiency and scalability aspects.
Alférez et al. [26]	Model-driven dynamic adaptations	Generation efficiency and complexity reduction of the adaptation space. Goal/Question/Metric (GQM) method
Taher et al. [27]	Model transformation	Not reported

IV. DISCUSSION OF RESULTS

The main result of this literature review is that the variation of methods changes from author to author, including the methods of adaptation, the adaptation layer, the software used and the evaluation methods. Some papers focused on the

detection of mismatches and some others proposed methods for detection of mismatches and the creation of adapters. Common adaptation methods include semi-automatic or automatically development of adapters in most cases. Furthermore, adapters can also be created statically or dynamically, stand alone or aspect oriented, using templates, process algebra or automata. Mismatches were identified the interface layer including the protocols, the messages and the behaviors. The most common type of mismatches was signature, ordering, extra message, missing message, merge, split message, quality of service, behavioral and data. While the adaptation methods reviewed comprised single layer or multilayer algorithms.

The most interesting methods found in the literature included a model proposed by [23] to support service adaptation by removing incorrect behaviors in complex adaptation scenarios by using both on-the-fly reduction and exploration techniques to make easy the generation of the adapters; the method presented by [24], which introduced an adapter in the description of the service to modify its functional behavior through a fine-grained template; and the method proposed by [25] that automatically detect and solve data mismatches using rules to make them work between architectural components through a Mismatch Detection Engine.

Although this was a paper comprising a limited amount of publications, our results confirm, actualize and extend previous literature [21], [25], [28]. Nevertheless, our review show that Web service adaptation is in its early stages and current approaches provide only limited solutions for service adaptation and there is none adaptation method that fits all types of mismatches. Additionally, this literature review highlights the need that adaptation frameworks and adaptation metrics should be created to support future research on software adaptation.

V. Conclusions

This papers contributes with a literature review that shows current adaptation techniques used for mismatch detection and web service adaptation. Web service adaptation are mechanisms with great potential to solve mismatches among web services. They can improve Web service interoperability, foster the reuse of Web services and can save development time and costs. The use of adapters can also solve mismatches interoperability and can provide non-intrusive mechanisms for software adaptation and service composition. Although there is a variety of different methods and techniques to detect mismatches and generate adapters at different layers of the service integration, there is none that can solve incompatibilities in all the levels of the service integration stack. Besides, current approaches focus more in adaptation techniques rather than mismatch detection. We hope this review may stimulate research in the area of software adaptation since the use of online web services is increasing. Further work is expected to improve current services adaptation methods and standardized ways to evaluate their reliability, performance and accuracy based on a standardized software adaptation framework.

Acknowledgment

We would like to thank Birmingham City University for all the facilities provided to carry out this research. Specially, we thank the people of the Enterprise Systems Lab for all their support and feedback.

References

[1] WC3, "Web Services Architecture," 2003. (Online). Available: http://www.w3.org/TR/ws- arch/ .

[2] E. Elabd, E. Coquery, and M.-S. Hacid, "Checking Compatibility and Replaceability in Web Services Business Protocols with Access Control," *2010 IEEE International Conference on Web Services*, pp. 409–416, Jul. 2010.

[3] W. Kongdenfha, H. R. Motahari-Nezhad, B. Benatallah, F. Casati, and R. Saint-Paul, "Mismatch Patterns and Adaptation Aspects: A Foundation for Rapid Development of Web Service Adapters," *IEEE Transactions on Services Computing*, vol. 2, no. 2, pp. 94–107, Apr. 2009.

[4] G. S. MacBeth, "Web Services," no. Chapter 1, pp. 397–405, 2004.

[5] F. M. Facca, S. Komazec, C. Guglielmina, and S. Gusmeroli, "COIN: Platform and Services for SaaS in Enterprise Interoperability and Enterprise Collaboration," *2009 IEEE International Conference on Semantic Computing*, pp. 543–550, Sep. 2009.

[6] H. Nezhad, B. Benatallah, F. Casati, and F. Toumani, "Web services interoperability specifications," *Computer*, 2006.

[7] R. Ambrosio, "A framework for addressing interoperability issues," *Power Engineering Society*, pp. 1–5, 2007.

[8] R. Seguel, R. Eshuis, and P. Grefen, "Constructing minimal protocol adaptors for service composition," *Proceedings of the 4th Workshop on Emerging Web Services Technology - WEWST '09*, pp. 29–38, 2009.

[9] J. Cámara, J. A. Martín, G. Salaün, J. Cubo, M. Ouederni, C. Canal, and E. Pimentel, "ITACA: An integrated toolbox for the automatic composition and adaptation of web services," in *Proceedings - International Conference on Software Engineering*, 2009, pp. 627–630.

[10] D. Yellin and R. Strom, "Protocol specifications and component adaptors," *ACM Transactions on Programming Languages,* vol. 19, no. 2, pp. 292–333, 1997.

[11] B. Benatallah, F. Casati, and D. Grigori, "Developing adapters for web services integration," *Advanced Information Systems Engineering,* 2005.

[12] X. Li, Y. Fan, S. Madnick, and Q. Sheng, "A pattern-based approach to protocol mediation for web services composition," *Information and Software Technology*, no. 3. September, 2008.

[13] S. Ryu, F. Casati, and H. Skogsrud, "Supporting the dynamic evolution of web service protocols in service-oriented architectures," *ACM Transactions on the Web (TWEB)*, vol. 2, no. 2, 2008.

[14] H. R. Nezhad, G. Y. Xu, and B. Benatallah, "Protocol-aware matching of web service interfaces for adapter development," *Proceedings of the 19th international conference on World wide web - WWW '10*. p. 731, 2010.

[15] M. Dumas, M. Spork, and K. Wang, "Adapt or perish: Algebra and visual notation for service interface adaptation," *Business Process Management*, pp. 65–80, 2006.

[16] A. Brogi and R. Popescu, "Automated generation of BPEL adapters," *Service-Oriented Computing–ICSOC 2006*, 2006.

[17] 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 Transactions on Automation Science and Engineering,* pp. 1–13, 2009.

[18] Z. Shan, "Integrated Service Adaptation," *2010 6th World Congress on Services*, pp. 140–143, Jul. 2010.

[19] H. R. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, "Semi-automated adaptation of service interactions," *Proceedings of the 16th international conference on World Wide Web - WWW '07*, p. 993, 2007.

[20] Y. Wang, F. Ishikawa, and S. Honiden, "Business Semantics Centric Reliability Testing for Web Services in BPEL," *2010 6th World Congress on Services*, pp. 237–244, Jul. 2010.

[21] H. J. La and S. D. Kim, "Static and dynamic adaptations for service-based systems," *Information and Software Technology*, 2010.

[22] H.-H. Lin, T. Aoki, and T. Katayama, "Automated Adaptor Generation for Services Based on Pushdown Model Checking," *2011 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, pp. 130–139, Apr. 2011.

[23] R. Mateescu, P. Poizat, and G. Salaun, "Adaptation of service protocols using process algebra and on-the-fly reduction techniques," *IEEE Transactions on Software Engineering*, vol. 38, no. 4, pp. 755–777, 2012.

[24] I. B. Lahmar and D. Belaid, "Developing Adapters for Structural Adaptation of Component-Based Applications," *2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 92–97, Jun. 2013.

[25] P. Velasco-Elizondo, V. Dwivedi, D. Garlan, B. Schmerl, and J. M. Fernandes, "Resolving data mismatches in end-user compositions," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7897 LNCS, pp. 120–136.

[26] G. H. Alférez, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz, "Dynamic adaptation of service compositions with variability models," *Journal of Systems and Software*, vol. 91, no. August 2015, pp. 24–47, May 2014.

[27] Y. Taher, M. Parkin, M. Papazoglou, and W. J. van den Heuvel, "Adaptation of Web Service Interactions Using Complex Event Processing Patterns", Service-Oriented Computing, pp. 601–609, 2011

[28] M. Eslamichalandar, K. Barkaoui, and H. R. Motahari-Nezhad, "Service Composition Adaptation: An Overview," *2012 Second International Workshop on Advanced Information Systems for Enterprises*, pp. 20–27, Nov. 2012.