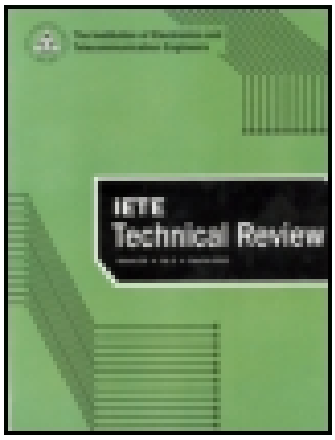

IETE Technical Review
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/titr20

Web Services Composition Mechanisms: A Review

Martín Garriga[ab], Andres Flores[ab], Alejandra Cechich[a] & Alejandro Zunino[bc]
[a] GIISCo Research Group, Facultad de Informática, Universidad Nacional del Comahue, Neuquen, Argentina
[b] CONICET (National Scientific and Technical Research Council), Argentina
[c] ISISTAN Research Institute, UNICEN, Tandil, Buenos Aires, Argentina
Published online: 16 Mar 2015.

To cite this article: Martín Garriga, Andres Flores, Alejandra Cechich & Alejandro Zunino (2015): Web Services Composition Mechanisms: A Review, IETE Technical Review, DOI: 10.1080/02564602.2015.1019942

To link to this article: http://dx.doi.org/10.1080/02564602.2015.1019942

Web Services Composition Mechanisms: A Review

Martín Garriga[1,2], Andres Flores[1,2], Alejandra Cechich[1] and Alejandro Zunino[2,3]

[1]GIISCo Research Group, Facultad de Informática, Universidad Nacional del Comahue, Neuquen, Argentina, [2]CONICET (National Scientific and Technical Research Council), Argentina, [3]ISISTAN Research Institute, UNICEN, Tandil, Buenos Aires, Argentina

ABSTRACT

Web Service composition is becoming the most promising way for business-to-business systems integration. However, current mechanisms for service composition entail a trade-off on multiple and complex factors. Thereby, existing solutions based on business Web Services, semantic Web Services, or the recent RESTful services, lack of a standardized adoption. This paper gives an overview of current approaches according to a set of features. Moreover, related core problems and future directions of service composition mechanisms are pointed out.

Keywords:
Choreography, Mashup, Orchestration, Service Composition, Web Services, Workflow.

1. INTRODUCTION

The Service-Oriented Computing (SOC) paradigm has been mostly adopted by means of the Web Services technology [1]. Web Services aim to achieve safe and seamless interoperability, moving the elementary framework based on SOAP, WSDL, and UDDI [2], towards outsourced Web Services composition. Composition is different from traditional application integration where software pieces end up tightly coupled and physically combined. Service composition encompasses roles and functionality to aggregate multiple services into a single composite service, which can be even used as a basic service in further service compositions [1]. Several initiatives provide platforms and languages for interenterprise integration of heterogeneous systems. Some of them make use of OWL-S for semantic markup of services and compositions, and BPEL for workflow representation of service compositions, where services bindings are known a priori [3]. Lately, RESTful services gained interest as a lightweight composition solution using hypermedia as the engine of application state [4].

Web Service composition involves highly complex factors, making it beyond the human capability to deal with the whole process manually [5]. A strong barrier of enterprises transition to SOC systems is the *trust* issue in its many dimensions – e.g., correct functioning of service compositions, service security, and aggregate Quality of Service (QoS) [6]. The fact that every Web Services developing organization may apply different concept models, affects to define and evaluate outsourced service compositions under a common language and in an unbiased way.

Current specification proposals for service composition could be actually seen as a transition towards a robust SOC frame-work. However, seamless composition still carries disagreement between researchers and practitioners about common standards to address real interoperation problems [7]. Hence, another challenging factor is to verify and satisfy compatible service interoperation [8].

Many solutions have been proposed to address composition issues, both from business and academic perspectives. Based on previous research in the topic, we have defined eight relevant features to characterise current service composition mechanisms. We give an overview of current proposals, discussing core problems and future directions.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 presents the set of features of service composition mechanisms. Section 4 presents a review of current service composition approaches, highlighting research challenges in the field. Conclusions and future work are presented afterwards.

2. RELATED WORK

Initial standards for service composition in terms of orchestration and choreography are presented in [9] – e.g., Business Process Modeling Language (BPML) for orchestrations, and Web Service Choreography Interface (WSCI) for choreographies. Current standards are surveyed in [7] classifying proposals in three groups: (1) choreography protocols to describe external message exchange of a Web Service and its client or among several collaborating services; (2) orchestration/workflow process models like BPEL4WS and BPML, or focused on QoS such as Meteor-S, MAIS and QoSOnt; and (3) semantic basis represented by web ontology

language for services (OWL-S) and web service modeling ontology (WSMO).

Approaches of automatic service composition are surveyed in [5], focused in workflows (business perspective) and planning (Artificial Intelligence field). A flexible workflow may support automatic process adaptation and cross-enterprise integration. Dynamic workflow methods allow automatic binding of abstract nodes to concrete resources/services. Dynamic composition methods could also focus on executable plans or processes automatically generated – where a Web Service is specified by its preconditions and effects (input/output) – without knowledge of any predefined workflow. A survey in the same topic is given in [10], discussing two main issues: services semantics and formalization of pre/post conditions. A clear trend is identified on QoS, user-oriented properties (constraints/ preferences/context), services complex dependencies, and compositions transactional behaviour. Another related survey is presented in [11], exploring Sematic Web Services based composition that rely on a machine-readable specification encoding services' interface/properties/capabilities. According to the level of automation/dynamicity/user involvement, approaches are classified as based on instance/template/declarative/workflow/planning/context/agent.

Service adaptation is useful for services interacting in unpredictable ways – i.e., disregarding predefined service workflows or conversations. Two levels of mismatches are identified in [8]: service interfaces, implying syntax adaptation (e.g., identifiers/data types) to perform message transformation (data mapping); and service business protocols (behaviour), related to constraints that services impose on message exchange sequences. Behavioural adaptation techniques are deeply analysed in [12]. Three related concepts are presented for service compatibility into choreographies: (1) realizability, when services collectively enforce control-flow constraints; (2) substitutability, where replacing one service with another causes no incompatibilities; (3) controllability, as the ability to control branches in service communication protocols. For composition flows, adaptation or re-binding should be transparently done with minimal or no user intervention, according to environment changes/customers' needs. Also, service processes should be dynamically modified in simple/cost-effective ways [13].

Web Service based business processes outlined with QoS have four main advantages [14]: (1) translates organizational visions more efficiently into business processes; (2) eases selection and execution to fulfil customers' expectations and user requirements; (3) assures compliance monitoring both to initial QoS requirements and targeted objectives; (4) helps generating potential alternatives to adapt a composite service to hold initial QoS requirements. In [15], a survey of approaches is given according to the following activities: defining functionalities (or *tasks*) required by composite services and their interactions (control/data flow); selection and binding of a proper implementation for each task; and defining an execution plan of the composition. The goal is to select the optimal execution plan to maximize the end-to-end QoS of the composition. As this is an NP-hard problem, current approaches make some simplifications: linearity of the objective function by Linear Integer Programming, local QoS-maximization (instead of global), absence of QoS-constraints (only goals), single objective optimization – maximizing one QoS dimension per time – and sub-optimal execution plan by using heuristics.

RESTful services are an alternative for building Web Services besides the WS-* standards (SOAP), where the basic HTTP methods (*put*, *post*, *get*, *delete*) are used to access resources [16]. A resource may abstract application state and functionality and/or provides any information – e.g., a document, an image, a tweet, or a weather forecast. Lightweight RESTful services are easier to consume, and often used to build community-driven services (*mashups*). Mashups are websites that combine information and services (typically RESTful) from multiple sources on the Web [17]. A comparison to "heavyweight" (SOAP-based) approaches is given in [4]. For resource discovery/composition is stated that no UDDI like technology exists for REST [18], but to manually inspect Web catalogues of RESTful services. Also, a combination of discovered services into mashups involves manually creating a mediation layer – data/process (protocol) mapping, or user interface customization [19]. Making use of semantic annotations as a lightweight ontology – e.g., SA-REST or MicroWSMO – could help overcoming those issues, but service's native data format must be properly mapped to the ontology data structure. Besides, companies require a standardized process/specification to embody mashups, for their broadly adoption [20]. Yet, mashups can be treated as compositions of loosely coupled and reusable services; then business modelling languages could be the standard way to connect and manage these services together with SOAP-based services.

3. FEATURES OF SERVICE COMPOSITION MECHANISMS

Considering previous surveys on Web Services composition (see Section 2), we have defined a set of eight features to characterize approaches in the field. Table 1 lists the set of features along with their accepted values. A description for each feature collecting

Table 1: Features of service composition mechanisms

Feature	Accepted values
Composition view	Orchestration/choreography/ workflow/other
Automation level	Manual/semi-auto/auto
Composition time	Static/dynamic
Standards conformance	Yes/no
Verification & validation	V&V/testing/monitoring/simulation
Service (composition) specification	Compositional/WS-*/other/ad hoc
QoS awareness	NFP/QoS
Adaptation level	Interface/protocol/workflow re-binding

specific factors discussed in the previous section is given below.

Composition View. Technologies for service composition are defined in general as follows [7]: *Orchestration* is an executable business process that can interact with internal/external services at message level [9]. *Choreography* tracks message sequences among multiple parties/sources (as a collaborative view) instead of a specific business process from a single party [9]. *Workflow* specifies the information flow among work tasks. Composite service as a workflow includes a set of atomic services (or tasks) together with the control/data flow among the services [5].

Automation Level. Service composition can be broadly classified into three categories [21]. *Manual* composition expects users to (graphically/textually) edit workflow scripts, for some workflow execution engine. Drawbacks include unscalability, manual rebinding, and the need of highly expert users. *Semi-automated* composition could make semantic advices for service selection. However, users still need to select services and link them up to then execute the generated workflow. These systems are still unscalable and lack fault-handling. *Automated* composition exploits planning or similar technology. Fully automation requires to understand context, semantics, and the problem domain space. Complexity of the service composition domain makes approaches to deal with small, restricted parts of fully automated composition [22].

Composition Time. Services can be composed either proactively and a priori (i.e., before requested by users), or reactively and on the fly (i.e., when requested by users) [23]. *Static* composition suits better when the composite service is used as is at a very high rate, providing a stable, available and fast composition. *Dynamic* composition provides more flexible and adaptive applications according to user requests, preferences, and context (e.g., location, time, profile).

Standards Conformance. Several standardization bodies currently address key issues towards a proper new Web Service based paradigm. Although the community already deals with service composition and inter-enterprise integration, yet standardization of other system aspects is being discussed – e.g., QoS-awareness, verification and validation, and adapters development. Without robust and common agreement on standards, real/seamless interoperation and composition problems cannot be addressed adequately [7]. Standards are, by definition, well-founded and documented. Thereby, composition systems conforming to standards may be potentially easier to implement and adopt than ad hoc solutions. This feature then implies relying on standards to implement such key aspects for composition systems.

Verification and Validation (V&V). *Validation* implies ascertaining the correct behaviour of service compositions. *Verification* involves checking maintenance of certain desirable properties of composite services [24], which is mandatory for composite Web Service flow prior to its execution [25]. An erroneous flow description may consume a huge amount of network resources (publicly shared). *Testing* Web Service compositions helps checking service interoperability/functionality/QoS [6]. *Monitoring* or *Simulation* (e.g., for performance analysis) are required to assess the correctness/effectiveness/safety/efficiency of composite services [24].

Service (Composition) Specification. A composition system may support certain type of specifications for services and/or compositions, through standard or ad hoc languages, or their extensions. *Compositional* specifications describe services interactions according to a Composition View. *WS-** service specifications describe functional/non-functional facets at atomic level. Non-functional descriptions may require extra effort on building domain ontologies and annotations [26]. Some approaches make use of *other* languages out of the Web Services scope – e.g., UML [27]; or their own ad hoc languages.

QoS-Awareness. Considering Non-Functional Properties (NFP) and particularly QoS is crucial for companies to meet their customers' requirements, becoming key factors upon the increasing number of Web Services of similar functionality [7], [28]. *NFPs-based* composition approaches should ideally fit available standards – e.g., the ISO/IEC 25000 series (SQuaRE) [29]. *QoS-based* composition approaches usually consider response time, results accuracy, completeness of covered data, price, availability, and reputation.

Adaptation Level. Mediation or adaptation implies an economic and effort-saving alternative to address

partial compatibility in real-life Web Service composition [8]. For service *interface* is associated with incompatibilities in service signatures – e.g., message/operation name, number/type of operations' input/output parameters. Adapters should perform message transformation (data mapping) according to previously defined rules. For service business *protocol* (behavioural) is related to ordering constraints that services impose on message exchange sequences. Adapters should rearrange exchanged messages upon business protocol mismatches. *Re-binding* (inherited from workflows) refers to changing parts of a composition upon detecting context changes, QoS infringements, or detecting failures in executing some components

4. CHARACTERIZATION OF APPROACHES

In this section, 42 significant approaches of Web Service composition mechanisms (Table 2) are analysed according to the set of eight features presented in the previous section. Making use of a public governmental digital library [30], a deep search was done in accessible online sources – e.g., Scopus, Science Direct, IEEE Xplore, ACM Digital Library, SpringerLink, and WileyOnline. A first search retrieved about 200 potentially relevant articles. For the final selection, we considered the most relevant publication – i.e., top tier conferences and journals, without repeated works from the same author/group.

4.1 Findings

Characterization of selected approaches according to the eight features is given in Table 3. Main results and findings are detailed as follows:

Workflow view of compositions. Most approaches neither focused on choreography nor on orchestration, but they represented the composition problem as a workflow (19 approaches). The parallelism of composite services with business processes and information flows in organizations is straightforward. Besides, several RESTful composition approaches adopted the workflow view by itself, or in combination with the mashup view – e.g., (Id.31).

Compositional specifications. Most approaches selected a business-oriented language for representing composite services. Particularly, the trend is the BPEL language (16 approaches), mostly in accordance with a workflow view (nine approaches), even for RESTful compositions approaches – e.g., (Id.29), (Id.31). Therefore, a broad adoption of business service compositions is feasible when stakeholders can use well-known business languages and process definitions.

Common Practices. Table 4 shows technologies, techniques, and practices supported by service composition systems. Some approaches use a formal mathematical foundation, mostly based on process algebra. Others focus on user-context representation –

Table 2: Service composition approaches

Main focus	Id – approaches	Id – approaches
V&V	01- Narayanan & M., 2002 [24]	06- Baresi, Di Nitto, 2007 [35]
	02- Brogi et al., 2004 [31]	07- Oh et al., 2006 [36]
	03- Foster et al., 2010 [32]	08- Smit & Stroulia, 2013 [37]
	04- Chan & Lyu, 2008 [33]	09- Bertolino et al., 2013 [38]
	05- Kazhamiakin et al., 2006 [34]	
QoS-Aware	10- Grùnmo et al., 2005 [39]	15- Paik et al., 2012 [22]
	11- Berbner et al., 2006 [28]	16- Fujii & Suda, 2009 [42]
	12- Cardoso et al., 2004 [14]	17- Alrifai et al., 2010 [43]
	13- Canfora et al., 2006 [40]	18- Wen et al., 2014 [44]
	14- Yang et al., 2012 [41]	
Adaptation	19- Cardellini et al., 2012 [45]	24- Seguel et al., 2010 [50]
	20- Tan et al., 2009 [46]	25- Pastrana et al., 2011 [51]
	21- Lin et al., 2011 [47]	26- Gutierrez-G. et al., 2011 [52]
	22- M. Nezhad et al., 2007 [48]	27- Charfi et al., 2007 [53]
	23- Wang et al., 2008 [49]	28- Koning et al., 2009 [54]
RESTful	29- Pautasso et al., 2009 [55]	34- Zhao et al., 2011 [58]
	30- Alarcon et al., 2011 [27]	35- Farokhi et al., 2012 [59]
	31- De Giorgio et al., 2010 [56]	36- Zhao & Doshi, 2009 [17]
	32- Peng et al., 2009 [20]	37- Zuzak et al., 2011 [60]
	33- Rosenberg et al., 2008 [57]	
Others	38- Barbosa & B., 2009 [61]	41- Zou et al., 2012 [64]
	39- Bauer & Müller, 2004 [62]	42- Menascé et al., 2011 [65]
	40- Mayer et al., 2008 [63]	

Table 3: Characterization of service composition approaches

Features	Id – Approaches	Total	
Composition view	Orchestration	03, 14, 15, 19, 25, 28, 29, 38, 39, 40	10
	Choreography	02, 03, 04, 09, 22, 23, 39, 41	8
	Workflow	05, 06, 07, 08, 10, 11, 12, 13, 16, 18, 20, 21, 24, 27, 31, 32, 33, 34, 35	19
	Other	01, 18, 26, 32, 33, 42	6
Automation	Manual	05, 06, 09, 13, 14, 17, 27, 28, 29, 30, 33, 37, 38	13
	Semi-auto	02, 03, 10, 11, 12, 19, 20, 22, 24, 31, 32, 35	12
	Automatic	01, 04, 08, 15, 16, 18, 21, 23, 25, 26, 34, 36, 39, 40, 41, 42	16
Comp. Time	Static	01, 02, 03, 05, 07, 08, 10, 12, 14, 17, 18, 20, 21, 22, 24, 32, 33, 34, 36, 37, 38, 39, 40, 41	24
	Dynamic	02, 04, 06, 09, 11, 13, 15, 16, 19, 21, 23, 25, 26, 27, 28, 29, 30, 31, 35, 39, 42	21
Stan.	Yes	01, 03, 06, 10, 11, 15, 39	7
V&V	V&V	01, 02, 03, 04, 05, 06, 20, 21, 25, 32, 34, 36, 37	13
	Testing	04, 07	2
	Monitoring	09, 12, 19, 27, 41	5
	Simulation	01, 08, 11	3
Specification	Compositional	02, 03, 04, 05, 06, 09, 11, 12, 13, 14, 19, 20, 21, 23, 24, 27, 28, 29, 31, 32, 39, 40, 42	23
	WS-*	04, 06, 07, 08, 10, 17, 18, 22, 25, 32, 38, 40, 41	13
	Other	01, 15, 25, 32, 34, 36, 37, 39, 40, 41	10
	Ad-hoc	08, 10, 15, 26, 27, 28, 29, 30, 31, 33, 35, 36, 38, 42	14
QoS	NFP	06, 15, 27, 40	4
	QoS	10, 11, 12, 13, 14, 16, 17, 18, 19, 25, 31, 35, 42	13
Adapt.	Interface	20, 22, 23, 25, 30, 32	6
	Protocol	02, 12, 19, 20, 21, 22, 23, 24, 25, 26, 31, 42	12
	Re-binding	04, 13, 16, 27, 28, 35, 42	7

e.g., (Id.11). In addition, business-oriented and academic-oriented proposals make distinctive choices. The former mostly adopts a workflow view, a business language, and a lightweight composition method, with syntactic service descriptions – e.g., (Id.12). The latter is oriented to dynamic and automatic service composition, mostly through a choreography based on semantic-enriched descriptions of services – e.g., (Id.15). The Web Service environment is highly complex making hard to automatize the whole generation process – e.g., workflow, bindings, and adapters [4]. This creates a trade-off between composition automation and complexity of required specifications and methods. Thus, some approaches use Finite State Machines as a simple abstract behavioural model, easy to derive from any more complex formalism.

Lack of standards agreement. Standards have been key enablers of Web Services deployment/adoption. However, the field of service composition still needs robust and common agreement on standards. Thereby, it becomes difficult to bring QoS and to trustify composition solutions. Excluding widely adopted standard languages (e.g., WSDL, BPEL), only 7 approaches adopted standards, particularly for Service Level Agreement, QoS policies, adapter development or visual modelling – e.g., (Id.11). A recent interesting effort is the OMG's SoaML [66], providing a UML profile for design and specification of atomic or composite structures. Standardization for RESTful approaches did not gain broad support yet [4], maybe considering that first RESTful approaches appeared in 2008.

Table 4: Common practices in service composition approaches

Practice – Id.	Practice – Id.
Petri-Nets – 01, 04, 20, 30	Planning – 07, 15, 41
Model-checking – 03, 05, 06, 21	Process calculus/algebra – 02, 03, 34, 38
Linear programming/optimization – 11, 17, 18, 19	
Common QoS Formulas – 13, 14	Model-driven (MD*) – 35, 39, 40
Aspect-oriented – 27	Finite-state machines – 9, 23, 37, 42

4.2 Future Research

Research possibilities/challenges are discussed as follows:

QoS-aware service composition. Formal/machine-readable specifications for external services describing functionality/QoS are still unrealistic [67]. Current sources of QoS for Web Services are not trustable, since in a trustful Web Service, QoS values received by consumers and periodically tested by service registries fulfil QoS values promised by providers [68]. A mature Web Service composition platform should support the specification/monitoring/dynamic agreement of QoS levels [69]. Also, service composition and atomic service specifications should be simple while computing QoS to make the whole process repeatable and applicable to different contexts.

Seamless service composition. Handling a completely heterogeneous and pervasive environment where software artifacts/devices of different capabilities could be seamlessly added is essential [70]. Most existing service composition techniques require programming effort to build an orchestration model – e.g., (Id.10), (Id.19). Composers need the right know-how on the application domain with high expertise on the service description language, the orchestration algebra, and the corresponding programming frameworks. All of this hinders common users from composing Web Services.

Adaptable, (Self)Adaptive service composition. Dealing with trusted automated adaptation of services contracts is hard as the basis is usually a manual specification – e.g., by applying model checking. Integrating service adaptation at verification time may decrease time/cost [47]. Services reuse/evolution often requires complex adapters [49]. Services may encapsulate conversational interfaces capturing interactions by control-flow dependencies. Mismatches may outrange services individual interactions up to the context of a business process conversation. However, minimal adapters are less complex and more efficient than processing all interactions, reducing message exchange overhead [50]. Self-adaptive compositions should be proactive, context-aware, and QoS-aware, combining runtime monitoring/(re-)planning [65]. However, current approaches can only perform simple tasks automatically [69].

Trustful service composition. The trust issue could be addressed through several testing and V&V techniques. Web Service testing still has open issues, such as testing without disrupting a runtime service operation, and deciding when testing is required and which operations need to be tested [6].

5. CONCLUSIONS AND FUTURE WORK

In this review, we have identified a variety of technologies/languages/techniques/practices from current approaches on Web Service composition and its importance for seamless inter-organizational business integration. However, migration from traditional software to SOC-based systems is not at the expected rate yet. Some discouraging issues are: trustworthiness, QoS definition/agreement, standardization, adaptation complexity, and runtime cost-effective V&V techniques.

In this sense, our work focused on testing-based service selection [71,72] is being adjusted to address trustful service composition, providing semi-automatic assistance for developers to confirm the suitability of candidate services, building interface/protocol-oriented adapters, and pragmatically attending the required testing task that inevitably follows any integration process [73].

Funding

This work is supported by projects [grant number ANPCyT-PICT-2012-0045]; [grant number UNCo-DSBR (04-F001)].

REFERENCES

1. M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *IEEE Comput.*, Vol. 40, no. 11, pp. 38–45, 2007.
2. O. Zimmerman, M. Tomlinson, and S. Peuser, *Perspectives on Web Services – Applying SOAP, WSDL and UDDI to Real-World Projects*. Berlin, Germany: Springer-Verlag, 2005.
3. S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WSAddressing, WS-BPEL, WS-Reliable Messaging, and More*. New Jersey, NJ: Prentice Hall PTR, 2005.
4. M. Lanthaler, and C. Gutl, "Towards a restful service ecosystem," in *4th IEEE DEST*, Dubai, 2010, pp. 209–14.
5. J. Rao, and X. Su, "A survey of automated web service composition methods," in *1st SWSWPC*, San Diego, CA, 2004, pp. 43–54.
6. M. Bozkurt, M. Harman, and Y. Hassoun, "Testing and verification in service-oriented architecture: A survey," *Software Testing, Verification Reliability*, Vol. 23, no. 4, pp. 261–313, Jun. 2013.
7. F. Daniel, and B. Pernici, "Web service orchestration and choreography: Enabling business processes on the web," E-Business Models, Services, and Communications, *IGI Global*, Vol. 2, pp. 251–74, Nov. 2008.
8. M. Eslamichalandar, K. Barkaoui, and H. R. Motahari Nezhad, "Service composition adaptation: An overview," in *2nd IEEE IWAISE*, Constantine, Algeria, 2012, pp. 20–7.
9. C. Peltz, "Web services orchestration and choreography," *IEEE Comput.*, Vol. 36, no. 10, pp. 46–52, Oct. 2003.
10. P. Bartalos, and M. Bieliková, "Automatic dynamic web service composition: A survey and problem formalization," *Comput. Informatics*, Vol. 30, no. 4, pp. 793–827, Jul. 2011.
11. S. Kumar, and R. Mishra, "Semantic web service composition," *IETE Technical Rev.*, Vol. 25, no. 3, pp. 105–21, May-Jun. 2008.

12. M. Dumas, B. Benatallah, and H. R. Motahari Nezhad, "Web service protocols: Compatibility and adaptation," *IEEE Data Eng. Bull.*, Vol. 31, no. 3, pp. 40−4, Sep. 2008.
13. F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan, "Adaptive and dynamic service composition in eflow," in *12th CAISE*, Stockholm, Sweden, 2000, pp. 13−31.
14. J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics: Sci., Services Agents World Wide Web*, Vol. 1, no. 3, pp. 281−308, Apr. 2004.
15. A. Strunk, "Qos-aware service composition: A survey," in *8th IEEE ECOWS*, Ayia Napa, Cyprus, 2010, pp. 67−74.
16. R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. California, CA, 2000.
17. H. Zhao, and P. Doshi, "Towards automated restful web service composition," in *7th IEEE ICWS*, Los Angeles, CA, 2009, pp. 189−96.
18. T. Ma, "Review on grid resource discovery: Models and strategies," *IETE Technical Rev.*, Vol. 29, no. 3, pp. 213−22, May−Jun. 2012.
19. E. Maximilien, H. Wilkinson, N. Desai, and S. Tai, "A domain-specific language for web apis and services mashups," in *5th ICSOC*, Vienna, Austria, 2007, pp. 13−26.
20. Y. Peng, S. Ma, and J. Lee, "Rest2soap: A framework to integrate soap services and restful services," in *IEEE SOCA*, Taipei, Taiwan, 2009, pp. 1−4.
21. S. Majithia, D. Walker, and W. Gray, "A framework for automated service composition in service-oriented architectures," in *Semantic Web: Res. Appl.*, Vol. 3053, pp. 269−83, May 2004.
22. I. Paik, W. Chen, and M. Huhns, "A scalable architecture for automatic service composition," *IEEE Trans. Services Comput.*, Vol. 7, no.1, pp. 82−95, Jan−Mar., 2012.
23. D. Chakraborty, and A. Joshi, "Dynamic service composition: State-of-the-art and research directions," IBM Research India, Tech. Rep. TR-CS-01-19, CSEE, UMBC, Dec., 2001.
24. S. Narayanan, and S. McIlraith, "Simulation, verification and automated composition of web services," in *11th ACM WWW*, Honolulu, Hawaii, 2002, pp. 77−88.
25. S. Nakajima, "Model-checking verification for reliable web service," in *Workshop on Object-Oriented Web Services*, OOPSLA, Seattle, USA, 2002.
26. M. Crasso, A. Zunino, and M. Campo, "A survey of approaches to web service discovery in service-oriented architectures," *J. Database Manag.*, Vol. 22, no. 1, pp. 102−32, 2011.
27. R. Alarcon, E. Wilde, and J. Bellido, "Hypermedia-driven restful service composition," *International Conference on Service Oriented Computing (ICSOC)*, San Francisco, 2011, pp. 111−20.
28. R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for qos-aware web service composition," in *4th IEEE ICWS*, Chicago, 2006, pp. 72−82.
29. ISO/IEC Technical Committee, "ISO/IEC 25000:2005 Software Engineering − Software product Quality Requirements and Evaluation (SQuaRE) − Guide to SQuaRE," Tech. Rep. no. 25000: 2005.
30. MINCyT, "Public digital library," Ministry of Science, Technology and Productive Innovation, Government of Argentina, 2014. Available: http://www.biblioteca.mincyt.gob.ar/.
31. A. Brogi, C. Canal, E. Pimentel, and A. Vallecillo, "Formalizing web service choreographies," *Electronic Notes Theoretical Comput. Sci.*, Vol. 105, pp. 73−94, Dec. 2004.
32. H. Foster, S. Uchitel, J. Magee, and J. Kramer, "An integrated workbench for model-based engineering of service compositions," *IEEE Trans. Services Comput.*, Vol. 3, no. 2, pp. 131−44, Apr−Jun. 2010.
33. P. Chan, and M. Lyu, "Dynamic web service composition: A new approach in building reliable web service," in *22nd IEEE AINA*, Okinawa, Japan, 2008, pp. 20−5.
34. R. Kazhamiakin, M. Pistore, and L. Santuari, "Analysis of communication models in web service compositions," in *15th ACM WWW*, Edimburg, Scotland, 2006, pp. 267−76.
35. L. Baresi, and E. Di Nitto, *Test and Analysis of Web Services*. Heidelberg, Germany: Springer, 2007.
36. S. Oh, H. Kil, D. Lee, and S. Kumara, "Wsben: A web services discovery and composition benchmark," in *4th IEEE ICWS*, Chicago, USA, 2006, pp. 239−48.
37. M. Smit, and E. Stroulia, "Simulating service-oriented systems: A survey and the services-aware simulation framework," *IEEE Trans. Service Comput.*, Vol. 6, no. 4, pp. 443−56, Oct−Dec.2013.
38. A. Bertolino, E. Marchetti, and A. Morichetta, "Adequate monitoring of service compositions," in *9th ACM Joint Meeting on Foundations of Software Engineering*, Saint Petesburg, Russia, 2013, pp. 59−69.
39. R. Grùnmo, M. Jaeger, and H. Hoff, "Transformations between uml and owl-s," in ECMDA-FA, *Model Driven Architecture−Foundations and Applications*, Haifa, Israel, 2005, pp. 269−83.
40. G. Canfora, M. Di Penta, R. Esposito, F. Perfetto, and M. Villani, "Service composition (re)binding driven by application-specific qos," in *4th ICSOC*, Chicago, USA, 2006, pp. 141−52.
41. Y. Yang, M. Dumas, A. Polyvyanyy, and L. Zhang, "Generalized aggregate quality of service computation for composite services," *J. Systems Software*, Vol. 85, no. 8, pp. 1818−30, Aug. 2012.
42. K. Fujii, and T. Suda, "Semantics-based context-aware dynamic service composition," *ACM Trans. Autonomous Adaptive Systems*, Vol. 4, no. 2, pp. 12:1−12:30, May, 2009.
43. M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *19th ACM WWW*, Raleigh, USA, 2010, pp. 11−20.
44. S. Wen, Q. Li, C. Tang, A. Liu, L. Huang, and Y. Liu, "Processing multiple requests to construct skyline composite services," *J. Web Eng.*, Vol. 13, no. 1, pp. 53−66, Mar. 2014.
45. V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. Lo Presti, and R. Mirandola, "Moses: A framework for qos driven runtime adaptation of service-oriented systems," *IEEE Trans. Software Eng.*, Vol. 38, no. 5, pp. 1138−59, Sep−Oct. 2012.
46. 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 Trans. Automation Sci. Eng.*, Vol. 6, no. 1, pp. 94−106, Jan. 2009.
47. H. Lin, T. Aoki, and T. Katayama, "Automated adaptor generation for services based on pushdown model checking," in *18th IEEE ECBS*, Las Vegas, USA, 2011, pp. 130−9.
48. H. R. Motahari Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, "Semi-automated adaptation of service interactions," in *16th ACM WWW*, Bannf, Canada, 2007, pp. 993−1002.
49. K. Wang, M. Dumas, C. Ouyang, and J. Vayssière, "The service adaptation machine," in *6th IEEE ECOWS*, Dublin, Ireland, 2008, pp. 145−54.
50. R. Seguel, R. Eshuis, and P. Grefen, "Generating minimal protocol adaptors for loosely coupled services," in *8th IEEE ICWS*, Miami, USA, 2010, pp. 417−24.
51. J. L. Pastrana, E. Pimentel, and M. Katrib, "Qos-enabled and self-adaptive connectors for web services composition and coordination," *Computer Languages, Systems Struct.*, Vol. 37, no. 1, pp. 2−23, Apr. 2011.
52. J. Gutierrez-Garcia, and F. Ramos-Corchado, "Exception handling in pervasive service composition using normative agents," *J. Web Eng.*, Vol. 10, no. 3, pp. 175−96, Sep. 2011.

53. A. Charfi, and M. Mezini, "Ao4bpel: An aspect oriented extension to bpel," *World Wide Web*, Vol. 10, no. 3, pp. 309–44, Sep. 2007.
54. M. Koning, C. Sun, M. Sinnema, and P. Avgeriou, "Vxbpel: Supporting variability for web services in bpel," *Information Software Technol.*, Vol. 51, no. 2, pp. 258–69, Feb. 2009.
55. C. Pautasso, "Restful web service composition with BPEL for REST," *Data Knowledge Eng.*, Vol. 68, no. 9, pp. 851–66, Sep. 2009.
56. T. De Giorgio, G. Ripa, and M. Zuccala, "An approach to enable replacement of soap services and rest services in lightweight processes," in *Current Trends in Web Engineering (CWE)*, Vienna, Austria, 2010, pp. 338–46.
57. F. Rosenberg, F. Curbera, M. Duftler, and R. Khalaf, "Composing restful services and collaborative workflows: A lightweight approach," *IEEE Internet Comput.*, Vol. 12, no. 5, pp. 24–31, Sep–Oct. 2008.
58. X. Zhao, E. Liu, G. Clapworthy, N. Ye, and Y. Lu, "Restful web service composition: Extracting a process model from linear logic theorem proving," in *7th IEEE NWeSP*, Salamanca, Spain, 2011, pp. 398–403.
59. S. Farokhi, A. Ghaffari, H. Haghighi, and F. Shams, "Mdches: Model-driven dynamic composition of heterogeneous service," *Int. J. Commun., Network System Sci.*, Vol. 5, no. 9A, pp. 644–60, Sep. 2012.
60. I. Zuzak, I. Budiselic, and G. Delac, "Formal modeling of restful systems using finite-state machines," in *International Conference on Web Engineering (ICWE)*, Paphos, Cyprus, 346–60, 2011.
61. M. A. Barbosa, and L. Barbosa, "A perspective on service orchestration," *Sci. Comput. Programming*, Vol. 74, no. 9, pp. 671–87, Jul. 2009.
62. B. Bauer, and J. Müller, "Mda applied: From sequence diagrams to web service choreography," in *4th ICWE*, Munich, Germany, Vol. LNCS 3140. Springer, 2004, pp. 132–6.
63. P. Mayer, A. Schroeder, and N. Koch, "Mdd4soa: Model-driven service orchestration," in *12th IEEE EDOC*, Munich, Germany, 2008, pp. 203–12.
64. G. Zou, Y. Chen, Y. Xu, R. Huang, and Y. Xiang, "Towards automated choreographing of web services using planning," in *26th AAAI*, Toronto, Canada, 2012.
65. D. Menasce, H. Gomaa, S. Malek, and J. Sousa, "Sassy: A framework for self-architecting service-oriented systems," *IEEE Software*, Vol. 28, no. 6, pp. 78–85, Nov–Dec. 2011.
66. OMG Consortium, "Service oriented architecture modeling language (soaml) specification," Object Management Group, 2012. Available: http://www.omg.org/spec/SoaML/1.0.1/PDF/.
67. L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, "Validation of web service compositions," *IET Software*, Vol. 1, no. 6, pp. 219–32, Dec. 2007.
68. Z. Pan, and J. Baik, "A qos enhanced framework and trust model for effective web services selection," *J. Web Eng.*, Vol. 9, no. 4, pp. 327–46, Dec. 2010.
69. E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Eng.*, Vol. 15, no. 3-4, pp. 313–41, 2008.
70. B. Abdulrazak, and M. Yasir. "Review of challenges, requirements and approaches of pervasive computing system evaluation", *IETE Technical Rev.*, Vol. 29, no. 6, pp. 506–22, Oct–Dec. 2012.
71. M. Garriga, A. Flores, A. Cechich, and A. Zunino, "Testing-based process for service-oriented applications," in *30th IEEE SCCC*, Curico, Chile, 2011, pp. 64–73.
72. M. Garriga, A. Flores, C. Mateos, A. Zunino, and A. Cechich, "Service selection based on a practical interface assessment scheme," *Int. J. Web Grid Services*, Vol. 9, no. 4, pp. 369–93, 2013.
73. A. Flores, and M. Polo, "Testing-based process for component substitutability," *Software Testing, Verification Reliability*, Vol. 22, no. 8, pp. 529–61, Dec. 2012.

Authors

Martin Garriga received a BS degree in computer science from the National University of Comahue, Argentina (2010). Since 2011 he has pursued a PhD degree in computer science from UNICEN, Argentina. He is lecturer assistant at Informatics Faculty, National University of Comahue. His research interests include software engineering, SOC, component-based systems and software testing.

E-mail: martin.garriga@fi.uncoma.edu.ar

Andrés Flores received BS and MS degrees in computer science from the National University of South, Argentina (1999, 2005). He received his PhD degree in informatics from the University of Castilla-La Mancha, Spain (2009). He is adjunct professor at the Informatics Faculty, National University of Comahue. His research interests include software engineering, SOC, component-based systems and software testing.

E-mail: andres.flores@fi.uncoma.edu.ar

Alejandra Cechich received a BS degree in computer science from CAECE University, Argentina (1984). She got her MS degree in computer science from the National University of South, Argentina (2001) and a PhD degree in Informatics from the University of Castilla-La Mancha, Spain (2005). She is professor at the Informatics Faculty, National University of Comahue. Her research interests include software engineering, quality and software architecture.

E-mail: alejandra.cechich@fi.uncoma.edu.ar

Alejandro Zunino received his BS, MS, and PhD degrees in computer science from UNICEN, Argentina (1998, 2000, 2003). He is an adjunct professor at the Faculty of Exact Sciences, UNICEN. His research interests include software engineering, artificial intelligence, SOC and grid computing.

E-mail: azunino@exa.unicen.edu.ar

DOI: 10.1080/02564602.2015.1019942; Copyright © 2015 by the IETE