International Journal
of Applied Information Technology

http://journals.telkomuniversity.ac.id/ijait/

Survey Paper on User-Centric Service Composition

Henry Rossi Andrian [a], Wardani Muhamad [b]

[a] *Diploma of Computer Engineering, School of Applied Science, Telkom University, Indonesia*
[b] *Diploma of Information System, School of Applied Science, Telkom University, Indonesia*

rossi@tass.telkomuniversity.ac.id, wardani.muhamad@tass.telkomuniversity.ac.id

ARTICLE INFO

Received September 1st, 2020
Revised November 5th, 2020
Accepted November 9th, 2020
Available online March 4th, 2021

Keywords
service composition, user-centered, preferences

ABSTRACT

Service composition combines several services to complete more complex tasks and get increased service value. The service user determines the task be completed. Several standards have been presented to support the composition of services, including BPEL and WSCDL. However, existing standards can only be used or operated by IT professionals who understand software design and development well. The number of end-users who do not have specific skills is far greater than that of IT professionals. Therefore, to improve service composition capabilities, end-users must be given the opportunity to interact directly with the service composition system to obtain the new services they need. To create a user-centered service composition, some general needs must be met, namely: making it easier for users to obtain appropriate services, guaranteeing the fulfillment of Quality of Service (QoS), and visualizing composite services that are generated instantly. The resulting services meet external requirements submitted directly by users and pay attention to user preferences as internal (implicit) requirements. This paper presents various studies that have been produced to support the composition of user-centered services and problems that are still open as to research challenges in the future. This study's results are exspected to help service composition platform developers understand the various approaches to developing user-centered service composition.

* Corresponding author at:
School of Applied Science, Telkom University,
Jl. Telekomunikasi No. 1, Terusan Buah Batu, Bandung, 40257
Indonesia.
E-mail address: wardani.muhamad@tass.telkomuniversity.ac.id

ORCID ID:
- First Author: 0000-0002-5328-4540
- Second Author: 0000-0002-6420-1683

https://doi.org/10.25124/ijait.v4i01.3213
Paper_reg_number IJAIT000040105 2020

1. Introduction

The term service has been known for decades and is influenced by the global industrial economy's development. Along with the development of technology, especially information and communication technology (ICT), the term service is no longer used only in management disciplines but is used in engineering and computer disciplines. In Service-Oriented Architecture (SOA), services are software components distributed over computer networks and can be found by consumers. SOA provides a logical way of designing software systems to produce distributed services in a network and can be found [1, 2]. Capabilities are translated into service interfaces that end-users or other services can bind and run. Service can complete a specific task. As user requirements' complexity increases, the use of a single service may not be sufficient to complete complex tasks. Therefore, it is necessary to combine several services through the service composition process. Service composition involves combining and coordinating a series of services to achieve functionality that cannot be realized through existing services [3] and create new services that can increase the value [4].

In general, service composition involves several activities carried out sequentially, including defining user requirements, searching services, selecting services, and arranging selected services. Due to the complexity of producing new services, IT professionals generally run the composition of services who have expertise in software design and development. Meanwhile, end-users who have requirements that must be met cannot be directly involved in the service preparation process. As a result, composite services' compatibility will be reduced if it does not directly involve the end-user [5]. Services computing, web services, and developments in web technology, especially with the advent of web 2.0, have created potential opportunities for end-users to build their applications [6]. There are general needs that must be met to facilitate end-users to be directly involved in the composition of services, including [5]:

1. making it easier for users to find suitable services without having to use burdensome time and energy;
2. must be able to guarantee an efficient Quality of Service (QoS); and
3. able to provide instant visualization of the combined service.

Knowing the various approaches, technologies, and service composition mechanisms is the basic knowledge that must be mastered by service composition platform developers. More specifically, if those who will use the platform are end-users who do not have enough information technology expertise, an understanding of the approach and technology must be sufficient. Various solutions have been produced by researchers and published in scientific articles. The provision of survey papers will help other researchers to gain concise and comprehensive knowledge. One survey paper related to user-centered service composition was proposed by [7] in investigating related technologies and methods on service composition and their impact on user-centered service development. This paper contributes to determining the essential requirements when designing a service composition environment and comparing each requirement in a service composition mechanism. However, this paper does not explain the service composition visualization approach that simplifies end-users' service composition process. This paper aims to present the approaches to developing user-centered service composition research and research opportunities that could be developed in the future.

The paper organization is explained as follows. An overview of the basic concepts is provided in section 2 to provide an overview of the service composition process. Furthermore, section 3 conveyed the development of research on the composition of user-centered services. Section 4 describes several research opportunities that could be developed in the future. The last section concludes the result and future works of this study.

2. Literature Review

2.1. Web Services

Web services are the main technology to support SOA [3] and a tangible manifestation of the service computing concept [8]. World Wide Consortium (W3C) [9] defines web services as software systems designed to support machine-to-machine interactions over a network. The defining characteristic of web services is the use of the internet and the world wide web (Web) as a communication medium for services to interact with each other and with service consumers. By using the Web, Web services make use of the Uniform Resource Identifier (URI) infrastructure so that anyone who has access to the Web can find it. The URI schema assigns a name to each web service that uniquely identifies and allows a person to use all URI operations to access it.

Compared to other software entities, web services have the following features [8]:

- Descriptive: The service description language can describe web services.
- Released: Web services can be registered in the registry and released.
- Discoverability: Users can send search requests to the registry to find services and access information.
- Bind-able: Web service description information can be tied to another service that is running.
- Invoke-able: Web service can be called by remote code with description information.
- Compos-able: Web services can be combined with services to create larger (complex) services.

Up to now, SOAP and RESTful are two styles of web services commonly used. RESTful is increasingly being used to create web services because it is simpler than using SOAP. RESTful web services used Hypertext Transfer Protocol (HTTP) as a uniform interface to access the resources. GET, PUT, DELETE, and POST become operations used to manipulate the resources in the Web context. Message exchanging via responses based on client requests express the service capabilities in processing the message. The heavyweight protocol implemented by SOAP-based web services is a drawback that solved by RESTful web services. Its offers lightweight and stateless services that are particularly suitable for ad hoc integration over the Web.

2.2. Web Services Description

Service descriptions are useful for defining service interfaces that expose the operations provided by the service and binding mechanisms to each operation. Currently, there are several standards used as a language to describe web services, namely: Web Services Description Language (WSDL)[1], Web Application

[1] https://www.w3.org/TR/2007/REC-wsdl20-20070626/

Description Language (WADL), Semantic Annotation for WSDL and XML Schema (SAWSDL), Universal Service-Semantics Description Language (USDL), OWL-S, and Web Service Modeling Ontology (WSMO).

WSDL is an XML-based language used to create web service interface specifications. The WSDL document describes web services' functionality, how to interact with web services (concrete level), and the input and output messages provided by web services (abstract level). A web service's capabilities are described through a set of operations that describe the message exchange pattern as input and output parameters. WADL is designed to provide a process-engine description of an HTTP based web application. WSDL and WADL provide descriptive service descriptions, which has the main drawback of ambiguity.

This drawback could be overcome if the service could provide more detailed semantic descriptions. Adopting the semantic concept as a description of web services can improve web services' capabilities so that search, discovery, selection, composition, and integration are better. The semantic-based service description approach can be divided into 2 (two) groups, namely: annotation-based approach and semantic language-based approach. The annotation-based approach aims to enrich and complete service descriptions by establishing a correspondence between description elements and concepts from a set of referenced ontologies. The descriptive languages included in this group are SAWSDL and USDL.

Meanwhile, the semantic language-based approach has a major advantage in describing web services that can be interpreted clearly (unambiguously) by the program (machine). OWL-S and WSMO are the service description languages that can be selected in the semantic language approach group. SAWSDL defines how to add semantic annotations to various parts of a WSDL document, such as input and output message structure, interfaces, and operations. SAWSDL provides a mechanism by which concepts from a semantic model can be referenced using annotations. USDL [10] is a language that provides the formal semantics of web services enabling sophisticated conceptual modeling and automation in the search for available services, automated composition, and service integration. USDL uses an ontology-based on OWL WordNet for basic ontology concepts that can be easily searched and combined for easy setup flexibility. USDL is a language that can discuss all types of services by covering 3 (three) aspects: technical, operational, and business. OWL-S is an OWL-based web service ontology that facilitates service providers to describe web services' properties and capabilities in a clear (unambiguous) manner and can be interpreted by computers. OWL-S mark-up on web services facilitates automation of web service tasks, including automatic discovery, execution, composition, and interoperation of services. WSMO is a Web Service Modeling Framework (WSMF) based ontology that describes different aspects of dynamic web service composition, including dynamic search, selection, mediation, and use. WSMO provides 4 (four) components that differentiate web service semantics: ontologies, goals, services, and mediators.

2.3. Web Services Composition

The capability of a web service generally describes a business function of a business process. In some cases, the service capability must be fulfilled by combining (composition) several web services. Web services that involve several other web services are called composite services while combining several web services is called web service composition. Web services' composition involves combining and coordinating a series of services to achieve functionality that cannot be realized through existing services [3]. The composition is oriented to creating new services that can add value to improve the interface of existing services to be better [4]. Service compositions allow service system developers to arrange a collection of existing services and define their interfaces as aggregate solutions. The general form of service composition is orchestration and choreography, as shown in Figure 1. Service orchestration places a central coordinator (known as an orchestrator) who can invoke and combine single sub-activities. In contrast, service choreography defines complex tasks by defining interactions or communications that must be done by each service without adding a central coordinator. Each party or business process endpoint must define the public exchange message, the rules of interaction, and the agreement to support the choreography.

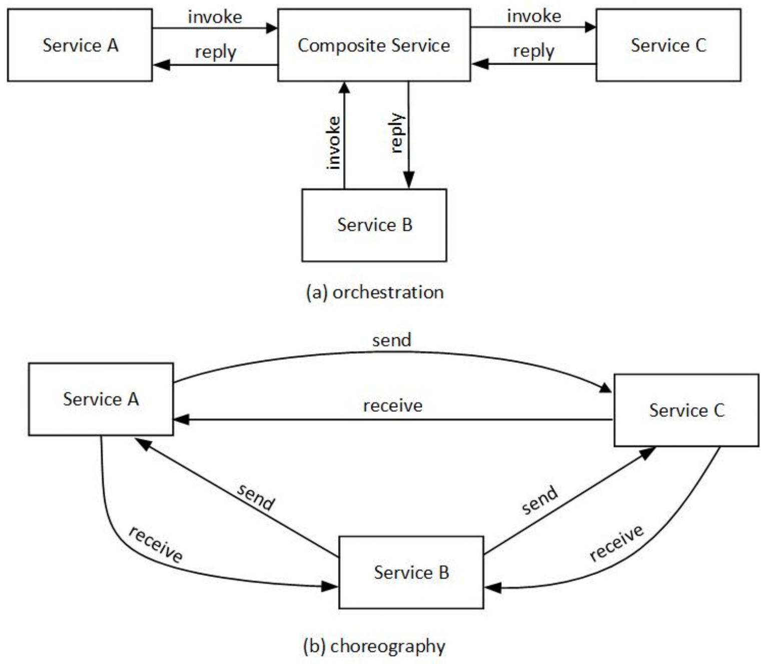

Figure 1 Service Composition Form [11]

Service composition has the following characteristics [12]:

- Web services are not like application libraries, which must be compiled and linked as part of the application.
- Basic components (individual services) remain separate (independent) from composite services.
- The web service composition mainly determines which services need to be called, the order of calls, and how to handle exceptions, etc. This can be viewed as a Web service-based workflow.
- The composition of the Web Service is nested. Service composition involves single/atomic services and involves other composite services as the basic components to be combined.

A framework can be used as a reference for activities that must be carried out sequentially (sequentially) to support the entire process in web services composition. Figure 2 presents a general web service composition framework.

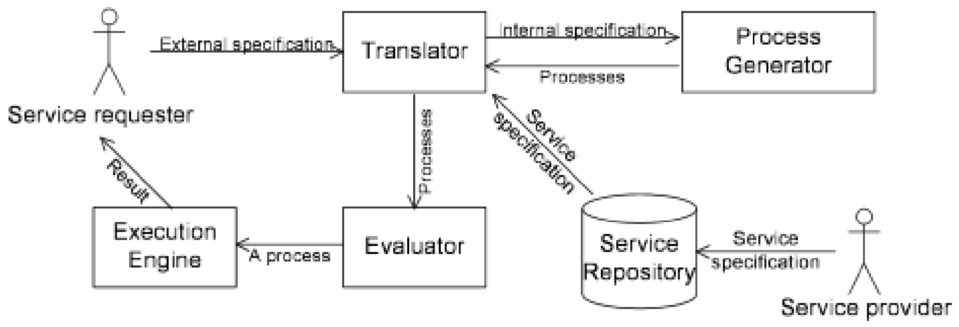

Figure 2 Service Composition General Framework [13]

Following Figure 4, the main actors in the service composition system are the service providers and service users. The service provider creates a variety of services and is then registered in the service repository. Users from the service repositories can find the essential services. Service users can use all the information they need according to the service description that has been made by the service provider. The framework is equipped with 4 (four) other components to support the process of creating a composite service according to the specifications or needs of service users and the service repository, namely: translators, process generators, evaluators, and execution engines. The translator is tasked with translating between the external language (specifications or requirements) proposed by service users (generally by specifying input and output pairs) and the process generator's internal language. The process generator then creates a composition plan that lists the operations to be fulfilled and compares them with the repository services. If more than one plan is found, the evaluator evaluates all plans and proposes the best outcome for the composite service construction. The execution engine executes the plan and returns the composite service to the service user.

3. User-Centric Web Service Composition

Currently, many solutions have been presented to support user-centered service compositions., Various methods are implemented to facilitate service discovery, such as service pool [12], service community [10], and skyline [14], all of which have the main objective of being able to deliver services according to end-user needs efficiently. The static composition of services is supported by the presence of the What You See is What You Get (WYSIWYG) editor, which provides the ability to visualize a web application presented as a solution by [5] [15] in addition to the creation of a guided service (wizard) as produced by [16]. Meanwhile, to support the dynamic composition of services, the provision of high-level graphic languages such as VINCA [17] and Flow Editor [18] have also been produced. The requirements that must be met in the composition of user-based services are limited to external requirements conveyed directly by the user and consider user preferences as internal (implicit) requirements. The best composition results present a suitable choice of available Web services considering global and local constraints related to QoS and user preferences [19]. Service composition that involves end-users directly has at least some general requirements [5]:

1. the similarity in functionality offered by several service providers causes the possibility of finding several service candidates to be composed. This causes the service selection process to spend time and effort for end-users;
2. must be able to provide a QoS guarantee so that QoS negotiations can be carried out efficiently, and

3. the service composition application must make it easier for end-users and be able to provide instant visualization.

Providing easy web service search facilities according to user needs is the first challenge that must be resolved by finding suitable services resulted from many web services with similar functionality and provided by different service providers. It will take a long time and a great deal of effort for the end-user who does not have specialized skills to find the services he will incorporate. Previous researchers have developed tools or methods to solve this problem. It takes for end-users to search for services with similar functionality from different service providers to avoid a large amount of time. A second requirement that must be met is the compilation of composite services that are easy to use or operate by end-users who do not have special expertise in service composition. Composite service arrangements in the service composition process can be made statically or dynamically. Static service composition is very dependent on user skills, especially information technology skills. Users are required to have skills in designing composite services, including selecting primitive services, compiling composition logic, and defining flow and control between each service. Procedure-generated composite services will follow the design and configuration according to the design results. On the other hand, the dynamic service composition will produce a composite service according to user requests in both visual form and natural language.

The static service composition provides tools such as a form or wizard and a WYSIWYG editor. Forms or wizards are used to help users define composite services by providing key information such as the location of each primitive service, the call order for each service, and the output format of the service composition process and execution results to the system. In Easy SOA [20], users can easily build applications because the installation of output and input between services can be done easily using a web browser. Users can export applications like web applications or web services without having to run deployment operations. Chafle et al. [16] observed the absence of an Integrated Development Environment (IDE) to facilitate the composition process, reducing development time and integration efforts. The IDE built is equipped with 2 (two) wizards, namely: Synthy Project Builder Wizard and WSC Wizard. The Synthy Project Builder Wizard is used to start the project and provide direction to the user in determining the location of required resources such as ontology concepts, service ontology types, registry examples, and resource files containing ontology processing rules. While the WSC Wizard is used to capture the information needed for setting up a new service composition. The WYSISYG editor produced [5, 6] supports the visualization of service composition, making it easier for users to design a web service layout to be composed by selecting services (in the form of interface components) and presenting an overview of the use of selected services on the client-side. Apart from that, end users can also modify the generated composite service instantly. Xiang and Madey [15] provide a task editor in an HTML template or a graphical composition tool for running composition services. A framework is also provided to facilitate portal developers to integrate new services into their portals for public use.

Dynamic service composition attempts to eliminate or reduce user contributions in manually writing service composition document semantics. One way to solve this problem is to provide a service composition using a high-level graphic language that makes it easy for users to determine the desired composite service. VINCA [17] developed a high-level graphic editor that allows end-users to define services at the business level. The graphical editor provided by Flow Editor [18] helps users describe the ideal composite service in the form of flowcharts. Meanwhile, Saifipoor et al. [21] adopted the Reo coordination language to produce a service composition platform. The visual approach is also applied in the system's visualizer component built by Rao et al. [22], which is used to represent the composition logic determined by the user in a graphical form. Fujii and Suda [23] taken a relatively different approach, wherein the service composition process provides natural language processing capabilities to analyze user requests written in natural language documents.

Table 1 summarizes the approaches used to make it easier for end-users to compose services.

Table 1 Service Composition Visualization Approaches

Approach	Characteristic	Examples
WYSIWYG	Provides an overview of web service layouts by presenting a preview of the capabilities (response) of the selected web service	[5, 6]
Wizard	Provides a composite web service creation stage including the location of each primitive service, service invocation sequence, and service composition output	[16] [20] [24]
High-level graphic language	Provides a graphic language that can be understood by end-users to be able to describe composite services at a business level according to user requirements	[17]
Workflows	Implement a flowchart to assist end-users in determining the order of the web service arrangement	[18] [22] [21] [23]
Spreadsheets	Adopt spreadsheets capability in storing the web services in cells and linking between cells using formulas	[25]
Web (Service) Mashups	Develop end user own applications by repurpose existing web data and APIs	[26] [27] [28] [29] [30]

It is necessary to consider user preferences to complete user satisfaction with the results of service compatibility. User preference becomes an implicit requirement, which complements the explicit requirements directly provided by the user. The best composition results present a suitable choice of available Web services considering global and local constraints related to QoS and user preferences [19]. PASS [31] uses user preferences and initial conditions as a personalized requirement. The requirements that have been compiled are used as the basis for automatically composing composite services. Practical examples of applying preference are provided by Hua et al. [32] and Zhao et al. [33]. Hua et al. [32] used preferences to help tourists plan travel plans. The complexity of the problem in determining the travel plan is influenced by the variety of transportation services and transport network changes.

4. Research Opportunities

Service composition is still the prerogative of professional programmers. Although some service composition standards provide intuitive visual abstraction, composing composite services still requires development expertise and software engineering knowledge. Both skills are not owned by the end-user (knowledge worker) as the party who has the requirements for composite services. An often-overlooked limitation of current systems is that they do not make composition languages accessible to end-users [34] [5]. As the main users of services, end-users should be given more support to easily compile composite services and get the result according to their needs. Also, web service technology until today has focused exclusively on machine computing and does not consider the special needs that arise when humans are involved in applications. Therefore, more needs to be done to reconcile humans, and machines need to enable the two [34] seamless integration.

Current developments in IT concepts and technologies such as big data, cloud computing, mobile device technology, IoT, and social networks also influence the emergence of research opportunities in the area of service composition. IoT and mobile technology are driving the emergence of web services at scale. This web service also expands the use of the language used to describe web services no longer using WSDL but using plain text. The shift in languages outside the standard causes the composition of services to use new techniques such as web information extraction, natural language processing, data and text mining, collaborative tagging, and information retrieval [35]. The success of the concept of big data and cloud computing also provides new challenges in the area of service composition. The development of existing algorithms and models can create composite services from online services in a huge number and spread in the cloud. They must be accessed simultaneously into a coherent system, which is a challenge that must be answered [35] and causes the formation of an abstraction layer that shifts the focus from infrastructure and operations to cloud services [34]. Online service composition can provide a promising direction for achieving scalable and adaptive composition solutions for handling large-scale, highly dynamic, and diverse big data services. Social networks also support the availability of a very large number of services such as big data. The growth of social network users who use the various services provided raises the challenge of providing composite services based on social relationships. One of the emerging opportunities is how to present potential composite services to service users by detecting hidden relationships between services through recording service user interactions with service data [35]. Arranging services across multiple mobile devices in ubiquitous environments presents new challenges that do not occur in traditional service composition settings—particularly composition mechanisms in pervasive environments that address context awareness, heterogeneity, and device contingencies. For example, unpredictable availability of mobile services and devices), and personalization (e.g., service provision based on user preferences) [11].

The provision of a service computing system platform that is reliable and easy to use also contributes to the service composition process, especially for end-users. Such platforms must be equipped with multiple intelligences capable of determining interactions between services and improving processing and data flow [34]. Dependence and conflict between services must also be considered to meet the composition of web services aware of QoS. Until recently, existing methods did not include conflicts and dependencies between web services. The task's service implementations are selected separately from other tasks [36].

5. Conclusion and Future Works

The study that has been conducted aims to create a resume of approaches, constraints, characteristics, and research opportunities on user-centered service composition issues. The analysis was carried out by observing 30 papers that have been published in the international database journal, including proceedings and periodical journals. The results obtained, the most service composition approach proposed by the researcher is web mashups and flowcharts. Web mashups are the preferred approach because mashups can describe the service output easily understood by end-users. Then, the flowchart approach is proposed mainly because the service composition's logic is to arrange a collection of web services in a flow that exchanges messages between one web service and another. After identifying the various approaches proposed in previous studies, the next work to be done is to create a general model of user-centered service composition. This model can be generated by implementing meta-analysis.

Bibliography

[1] M. P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-Oriented Computing: State of The Art and Research Challenges," Computer, vol. 40, no. 11, pp. 64-71, 2007.

[2] M. P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics, and Directions," in Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003.

[3] G. Baryannis and D. Plexousakis, "Automated Web Service Composition: State of the Art and Research Challenges," Information Systems Laboratory Hellas Institute of Computer Science, 2010.

[4] M. P. Singh and M. N. Huhns, Service-Oriented Computing: Semantics, Processes, Agents, John Wiley & Sons, Ltd, 2005.

[5] X. Liu, G. Huang and H. Mei, "Towards End-User Service Composition," in 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), 2007.

[6] X. Liu, G. Huang, and H. Mei, "A User-Oriented Approach to Automated Service Composition," in 2008 IEEE International Conference on Web Services, 2008.

[7] N. Laga, E. Bertin, and N. Crespi, "User-centric services and service composition, a survey," in 32nd Annual IEEE Software Engineering Workshop, 2009.

[8] Z. Wu, S. Deng, and J. Wu, Service Computing: Concepts, Methods and Technology, Elsevier inc, 2015.

[9] D. Booth, H. Haas, F. McCabe, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture," W3C, 2004.

[10] S. Kona, A. Bansal, G. Gupta, and T. D. Hite, "Web Service Discovery and Composition using USDL," in 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06), 2006.

[11] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," Information Sciences, vol. 280, pp. 218-238, 2014.

[12] I. EI Bitar, F.-Z. Belouadha and O. Roudies, "Review of Web Services Description approach," in 8th International Conference on Intelligent Systems: Theories and Applications (SITA), 2013.

[13] M. Vukovic, "Context-aware service composition," University of Cambridge Computer Laboratory, 2007.

[14] H.-y. Paik, A. L. Lemos, M. C. Barukh, B. Benatallah, and A. Natarajan, Web Service Implementation and Composition Techniques, Springer, 2017.

[15] X. Xiang and G. Madey, "A Semantic Web Services Enabled Web Portal Architecture," in IEEE International Conference on Web Services (ICWS'04), 2004.

[16] G. Chafle, G. Das, K. Dasgupta, A. Kumar, S. Mittal, S. Mukherjea and B. Srivastava, "An Integrated Development Environment for Web Service Composition," in IEEE International Conference on Web Services (ICWS 2007), 2007.

[17] Y. Han, H. Geng, H. Li, J. Xiong, G. Li, B. Holtkamp, R. Gartmann, R. Wagner and N. Weissenberg, "VINCA – A Visual and Personalized Business-Level Composition Language for Chaining Web-Based Services," in ICSOC 2003, 2003.

[18] B. Pi, G. Zou, C. Zhong, J. Zhang, H. Yu, and A. Matsuo, "Flow Editor: Semantic Web Service Composition Tool," in 2012 IEEE Ninth International Conference on Services Computing, 2012.

[19] H. Fekih, S. Mtibaa, and S. Bouamama, "User-centric Web services Composition Approach Based on Swarm Intelligence," in IEEE 18th International Conference on High-Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems, 2016.

[20] T. Yamaizumi, T. Sakairi, M. Wakao, and H. Shinomi, "Easy SOA: Rapid Prototyping environment with web Services for End Users," IEEE International Conference on Web Services (ICWS'06), 2006.

[21] S. Saifipoor, B. T. Ladani, and N. Nematbakhsh, "A Dynamic Reconfigurable Web Service Composition Framework Using Reo Coordination Language," in Fifth European Conference on Web Services, 2007.

[22] J. Rao, P. Kungas and M. Matskin, "Logic-based Web services composition: from service description to process model," in 2004 IEEE International Conference on Web Services, 2004.

[23] K. Fujii and T. Suda, "Dynamic service composition using semantic information," in 2nd international conference on service-oriented computing, 2004.

[24] I. Weber, H.-Y. Paik and B. Benatallah, "Form-Based Web Service Composition for Domain Experts," ACM Transactions on the Web, vol. 8, no. 1, 2013.

[25] Z. Obrenovic and D. Gasevic, "End-User Service Computing: Spreadsheets as a Service Composition Tool," IEEE Transactions on Services Computing, vol. 1, no. 4, pp. 229-242, 2008.

[26] E. M. Maximilien, A. Ranabahu and K. Gomadam, "An Online Platform for Web APIs and Service Mashups," IEEE Internet Computing, vol. 12, no. 5, pp. 32-43, 2008.

[27] C. Cappiello, F. Daniel, M. Matera, M. Picozzi, and M. Weiss, "Enabling End-User Development through Mashups: Requirements, Abstractions, and Innovation Toolkits," in IS-EUD 2011: End-User Development. Lecture Notes in Computer Science, vol 6654, 2011, pp. 9-24.

[28] M. Matera, M. Picozzi, M. Pini, and M. Tonazzo, "PEUDOM: A Mashup Platform for the End User Development of Common Information Spaces," in ICWE 2013: Web Engineering, 2013, pp. 494-497.

[29] D. Lizcano, F. Alonso, J. Soriano, and G. López, "A component- and connector-based approach for end-user composite web applications development," Journal of Systems and Software, vol. 94, pp. 108-128, 2014.

[30] A. Namoun, T. Nestler and A. D. Angeli, "Service Composition for Non-Programmers: Prospects, Problems, and Design Recommendations," in 2010 Eighth IEEE European Conference on Web Services, 2010.

[31] Y. Li, J. Huai, H. Sun, T. Deng and H. Guo, "PASS: An Approach to Personalized Automated Service Composition," in 2008 IEEE International Conference on Services Computing, 2008.

[32] Y. Hua, J. Cao, Q. Gu and Y. Tan, "PD-TRP: A Service Composition Approach for the Personalized and Optimized Door-to-Door Travel Plan Recommendation," in 2017 IEEE International Conference on Web Services (ICWS), 2017.

[33] Y. Zhao, S. Wang, Y. Zou, J. Ng, and T. Ng, "Automatically Learning User Preferences for Personalized Service Composition," in 2017 IEEE 24th International Conference on Web Services, 2017.

[34] A. L. Lemos, F. Daniel and B. Benatallah, "Web Service Composition: A Survey of Techniques and Tools," ACM Computing Surveys (CSUR), vol. 46, no. 3, 2016.

[35] A. Bouguettaya, M. Singh, M. Huhns, Q. Z. Sheng, H. Dong, Q. Yu, A. G. Neiat, S. Mistry, B. Benatallah, B. Medjahed, M. Ouzzani, F. Casati, X. Liu, H. Wang, D. Georgakopoulos, L. Chen, S. Nepal, Z. Malik, A. Erradi, Y. Wang, B. Blake, S. Dustdar, F. Leymann, and M. P. Papazoglou, "A Service Computing Manifesto: The Next 10 Years," Communication of The ACM, vol. 60, no. 4, pp. 64-72, 2017.

[36] C. Jatoth, G. Gangadharan, and R. Buyya, "Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review," IEEE Transactions on Services Computing, vol. 10, no. 3, pp. 475-492, 2017.