Modern Applied Science; Vol. 12, No. 12; 2018
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Service Composition in Service Oriented Architecture: A Survey

Fatima Aladwan[1], Ahmad Alzghoul[2], Emad Mohammed Mahmoud Ali[1], Hussam N. Fakhouri[3] & Israa Alzghoul[3]

[1] Faculty of Educational Sciences, The University of Jordan, Amman, Jordan

[2] Facultad De informática, Universidad Politécnica de Madrid, Madrid-Spain

[3] King Abdullah II School of Information Technology, The University of Jordan, Amman, Jordan

Correspondence: Hussam N. Fakhouri, King Abdullah II School of Information Technology, The University of Jordan, Amman, Jordan. E-mail: h.fakhouri@ju.edu.jo

Received: Jun27, 2018 Accepted: July 6, 2018 Online Published: November 12, 2018

doi:10.5539/mas.v12n12p18 URL: https://doi.org/10.5539/mas.v12n12p18

Abstract

Service-Oriented Architecture (SOA) is a modular approach to software development based on the use of distributed, loose coupling replaceable components equipped with standardized interfaces for interaction over standardized protocols Component interfaces in a service-oriented architecture encapsulate the implementation details (operating system, platform, programming language) from the rest of the components, thereby enabling the combination and reuse of components to build complex distributed software packages, ensuring independence from the platforms and development tools used, facilitating scalability and manageability of the systems being created. In this paper we introduce a Service composition in service oriented architecture, it is present service composition with different approach used for composing services and provided.

Keywords: Service Oriented Architecture, Service Composition, SOA

1. Introduction

Service Oriented architecture (SOA) services are composed of grouping services together, service represent the basic part of service composition provided by service providers with functional features or non functional features Known as Quality of services utilized to clarify identical functionality of services, and its mission in every part of service collection is to determine services form a service composition and service quality. The features of service composition can be changed by modifying services. Best collection of services must be selected by Software engineering team to implement system non-functional and functional requirements (Syu,et al, 2014).

Service composition represent transferring user functional and non functional requirement to execution plan of service composition as Service Level Agreement (SLA) to detect the services that achieve those requirements, Service Level Agreement can bestructure, scenario or optimal execution plan (Shrivastava et al., 2015).

Composition of services can be either manual service composition or automatic service composition as shown in figure (1). In manual service composition developer accomplished it in order to increment the web services number for using that is the reason to the delay and greater costs. Since it is not easy to detect suitable web services because of the complexity of services the decision has been taken to use computing system, which is more beneficial approach with the privileges of lower costs, decrease the risk of using manual service composition (Casati et al., 2000).

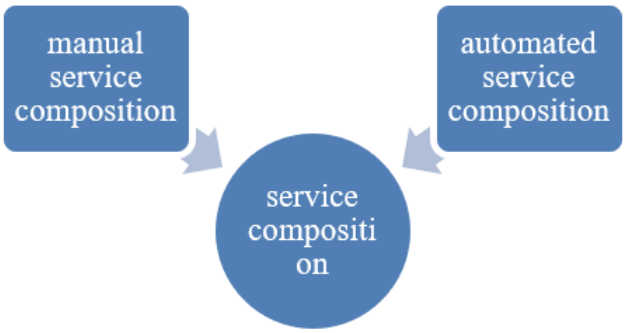

Figure 1. Service composition types

Automatic composition of process web services user who make the request which have the objective characterization or determine specific entries output, the process is continue until workflow and its procedures have been created. a lot of solution may be proceeded There is many solution probably that have to be taken by considering user conviction based on service quality or process execution quality when attributes of Quos impact it (Shen et al., 2013).

Service compositions can be classified into primitive and complex variations. In early service-oriented solutions, simple logic was generally implemented via point-to-point exchanges or primitive compositions. As the surrounding technology matured, complex compositions became more common (Shrivastava et al., 2015).

Much of the service-orientation design paradigm revolves around preparing services for effective participation in numerous complex compositions. So much so that the Service Compose ability design principle exits, dedicated solely to ensuring that services are designed in support of repeatable composition (Casati et al., 2000).

Tacble 1. Summary of Features and Benefits (opengroup)

Feature	Benefits	Supporting Infrastructure
Service	Improved information flow Ability to expose internal functionality Organizational flexibility	
Service Re-use	Lower software development and management costs	Service repository
Messaging	Configuration flexibility	Messaging program
Message Monitoring	Business intelligence Performance measurement Security attack detection	Activity monitor
Message Control	Application of management policy Application of security policy	PDPs and PEPs
Message Transformation	Data translation	Data translator
Message Security	Data confidentiality and integrity	Encryption engine
Complex Event Processing	Simplification of software structure Ability to adapt quickly to different external environments Improved manageability and security	Event processor
Service Composition	Ability to develop new function combinations rapidly	Composition engine
Service Discovery	Ability to optimize performance, functionality, and cost Easier introduction of system upgrades	Service registry
Asset Wrapping Virtualization	Ability to integrate existing assets Improved reliability Ability to scale operations to meet different demand levels	
Model-driven Implementation	Ability to develop new functions rapidly	Model-implementation environment

This paper is structured as follows: Section II introduces the existing service composition surveys. Section III discusses the service composition techniques. Finally, section IV presents our conclusions

2. Related Terminology

2.1 Service-Oriented Architecture (SOA)

Service-oriented architecture is an approach to software development, based on the use of services with standardized interfaces.SOA is based on the principles of reusing IT functional elements, eliminating duplication of functionality in software, unifying typical operating processes, ensuring the translation of the company's operating model to centralized processes and a functional organization based on an industrial integration platform (Syuet al., 2014) (Al-Sayyed, et al, 2017).

The components of the program can be distributed to different nodes of the network, and are offered as independent, loosely related construction of complex distributed software systems.SOA has proven itself for building large enterprise software applications. A number of developers and integrators offer tools and solutions based on like the WebSphere platform, BEA Aqua logic, Windows Communication Foundation, SAP Net Weaver, JVP Jupiter (Shenet al., 2013).

A manifesto for service-oriented architecture has six core values which are listed as follows (Ordonez et al., 2014):

- *Business value is given more importance than technical strategy.*
- *Strategic goals are given more importance than project-specific benefits.*
- *Intrinsic inter-operability is given more importance than custom integration.*
- *Shared services are given more importance than specific-purpose implementations.*
- *Flexibility is given more importance than optimization.*
- *Evolutionary refinement is given more importance than pursuit of initial perfection.*

2.2 Standardized Service Contract

Services adhere to standard communications agreements, as defined collectively by one or more service-description documents within a given set of services.

Service reference autonomy (an aspect of loose coupling)

The relationship between services is minimized to the level that they are only aware of their existence.

Service location transparency (an aspect of loose coupling)

Services can be called from anywhere within the network that it is located no matter where it is present.

2.3 Service Longevity

Services should be designed to be long lived. Where possible services should avoid forcing consumers to change if they do not require new features, if you call a service today you should be able to call the same service tomorrow (Casati et al., 2000).

2.4 Service Abstraction

The services act as black boxes, that is their inner logic is hidden from the consumers.

2.5 Service autonomy

Services are independent and control the functionality they encapsulate, from a Design-time and a run-time perspective (Ordonez et al., 2014).

2.6 Service Statelessness

Services are stateless, that is either return the requested value or give an exception hence minimizing resource use.

2.7 Service Granularity

A principle to ensure services have an adequate size and scope. The functionality provided by the service to the user must be relevant.

2.8 Service Normalization

Services are decomposed or consolidated (normalized) to minimize redundancy. In some, this may not be done, These are the cases where performance optimization, access, and aggregation are required.

2.9 Service Composability

Services can be used to compose other services.

2.10 Service Discovery

Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted.

2.11 Service Reusability

Logic is divided into various services, to promote reuse of code.

2.12 Service Encapsulation

Many services which were not initially planned under SOA, may get encapsulated or become a part of SOA.

2.13 Patterns

Each SOA building block can play any of the three roles:

2.13.1 Service Provider

It creates a web service and provides its information to the service registry. Each provider debates upon a lot of hows and whys like which service to expose, which to give more importance: security or easy availability, what price to offer the service for and many more. The provider also has to decide what category the service should be listed in for a given broker service (Shrivastava et al., 2015) and what sort of trading partner agreements are required to use the service (Ordonez et al., 2014).

2.13.2 Service Broker, Service Registry or Service Repository

Its main functionality is to make the information regarding the web service available to any potential requester. Whoever implements the broker decides the scope of the broker. Public brokers are available anywhere and everywhere but private brokers are only available to a limited amount of public. UDDI was an early, no longer actively supported attempt to provide Web services discovery (Shenet al., 2013).

2.13.3 Service Requester/Consumer

It locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its web services. Whichever service the service-consumers need, they have to take it into the brokers, bind it with respective service and then use it. They can access multiple services if the service provides multiple services (Syuet al., 2014)

The service consumer–provider relationship is governed by a standardized service contract (Syuet al., 2014) which has a business part, a functional part and a technical part.

Shrivastava et.al they focuses their work on recognizing the faulty services and their replacement depending on QoS constraints instead of modifying the whole composition, reconfiguration of the faulty region which will reduce the overheads of computations and service distractions in service delivery (Ordonez et al., 2014). The problem solved by using two approaches. First thing generating an optional path of services from its predecessor services to the completion of service process. After that establish an alternative service path from the start to the end of a service process (Casati et al., 2000).

Casati etal., proposed a service composition based on QoS consideration that supported management, specification and performance of E – services, sampled as processes and operated by a service process engine. Each node has its own rules for service selection, means some QoS parameters are correlated with each node(6).

Zhou et al., proposes a QoS aware of service composition approach which expands the options limit for service selection by allowing services of various granularity to be available for the selection. This approach is based on mixed integer linear programming concepts and optimized the user defined objectives to meet QoS constraints. They based on optimizing the selection approach by maximizing and minimizing the linear objective function; constrained in nature (Shrivastava et al., 2015).

Zhang et al. proposes an approach for identifying the faulty service within a service composition. They also proposed the approach of diagnosing and recovering the fault through reconfiguration using Dependency Matrix. This approach is based on the concept of Predicate on Probe (PoP). This dependency matrix is built on the basis of process workflow structures. This matrix is built without any historical knowledge of prior executions. This approach is a polynomial time algorithm(Shrivastava et al., 2015).

The traditional Quality of Service (QoS) analysis model cannot be directly applied on big data service composition. If we take the big data services into consideration of service composition problem So, Dong Li et al. proposed an expanded edition of QoS-based analysis model that uses the linear regression model with a set weight of QoS

using AHP analysis that improved the service selection algorithm based on backtracking method and validate its effectiveness (Zhang et al., 2012).

2.14 Resource & Service

Resource and service What's more are both two ideal ideas. Those suppliers deal with their services internally and uncover services to clients. A service, once bought eventually customer perusing An client Furthermore guaranteed Toward providers, provide an agreement between these gatherings. Those suppliers would answerable for mapping those services with Furthermore configurations (a supplier might delay those setup until the client begins to utilize them on attain An All the more utilization of their resources (Ordonez et al, 2014).

3. Service Composition Architecture

Service composition is the putting together of a number of simple services to make a more complex one. For example, a "product sale" web service could be composed of simpler "product selection", "shopping cart review", "payment method selection", "credit card payment", and "invoice payment" services. Service composition provides a key benefit such as the Ability to develop new function combinations rapidly. For example, if it is decided that the product sale service should cater for a new method of payment – "Internet cash" – this can be done by developing a new "Internet cash payment" service, and including it in the composition. So far, this seems to be little different from other software modularization techniques, from machine-code subroutines through to Java objects. Indeed, in an SOA that does not include messaging; service composition will be implemented by some such technique. But in many SOAs composition is implemented by services sending messages to invoke other services, and this technique gives much greater flexibility(Hudaib et al, 2016).

3.1 Two Styles of Composition are often Distinguished:

- Orchestration, in which one of the services schedules and directs the others. If the above example was designed as an orchestration, there would be a direction service that would invoke in sequence the product selection, shopping cart review, payment method selection, and, depending on the selection result, credit card payment or invoice payment services.
- Choreography, in which the composed services interact and cooperate without the aid of a directing service. If the above example was designed as a choreography, there would be no directing service: the product selection service would invoke the shopping cart review service, the shopping cart review service would invoke the payment method selection service, and the payment method selection service would invoke the credit card payment or invoice payment service.

3.1.1 Service Discovery

When a program uses a software service, the identity of that service can be explicitly given in the program code. For example, where services are implemented as Java objects, their methods can be invoked by name by user programs. Where messaging is used, the destinations of the messages can be explicitly named at programming time. This is called hard-wiring of service connections (Shen, et al, 2013).

Hard-wiring is a simple approach, but it has limitations. A different and much more flexible approach is service discovery. In this approach, the identity of the target service is not known at programming time, but is discovered at run time. The user program finds target services that meet its requirements, and chooses one of them (Gabrel et al, 2015).

3.2 The Benefits of Service Discovery are

Ability to optimize performance, functionality, and cost – by selecting component services by these criteria

Easier introduction of system upgrades – an upgraded service can be made available for selection in parallel with the one that it replaces, which can then be withdrawn

- Asset Wrapping

The IT assets of an enterprise can often be considered as actors that perform services. A CPU performs an information processing service; a file store performs an information storage service; and so on. This includes software as well as hardware assets) (Hudaib & Fakhouri, 2016). A database management system performs a database management service; an accounts package performs a financial information processing service. An important feature of SOA is the recognition that these assets perform services, and the development of software façades that provide access to these assets and have interfaces that are in the same form as the interfaces to other software services of the enterprise. This is called asset wrapping. From a component-based software engineering point of view, the assets and the façade are components that are assembled to form a software service. The software

services formed in this way can be used in service composition, have registry entries, and be dynamically discovered, in the same way as other services (Gabrel et al., 2015).

When an enterprise adopts an SOA, asset wrapping is typically applied to existing application software packages. This provides a significant benefit:

Ability to integrate existing assets – which means that the value of an enterprise's existing assets is preserved, the cost of developing or acquiring replacements is avoided, and there is a smooth migration path from the old architecture to the new one With the advent of SOA, some application vendors have begun to offer versions of their products in which the product capabilities are exposed as services. The acquisition of such a version is clearly a convenient way for an enterprise to achieve the "wrapping" of an application asset.

- Virtualization

Virtualization can be used to enable programs that were written to use one asset to be executed with a different asset. For example, there are so-called "hypervisors" that can provide different operating system environments to programs running on a single CPU (Al-Sayyed, et al, 2012). But in the context of SOA it is more commonly used to create virtual assets that are functionally similar to the underlying assets. This can deliver two benefits (Kim et al, 2016):

Improved reliability – through redundant operation of the underlying assets, so that one can take over when another fails or is withdrawn for maintenance. Ability to scale operations to meet different demand levels – through dynamically increasing or reducing the number of underlying assets that support a real asset, as demand rises and falls. These benefits are particularly important when the principles of SOA are applied to enterprise infrastructure. While SOA is most commonly thought of as a way of architecting an enterprise's application software, it can also be used at the infrastructure level, to create a Service-Oriented Infrastructure (SOI). Taken to the limit, this can provide a form of grid computing. The use of virtual assets that are made available over the Internet has become known as cloud computing (Lee et al, 2017), (Hudaib, & Fakhouri, 2017).

3.3 Model-Driven Implementation

Model-driven implementation refers to the automatic realization of a system or application from an abstract model. Where the model starts at a high level of architectural abstraction, it is usually referred to as Model-Driven Architecture (MDA). SOA lends itself particularly well to model-driven implementation, because it is based on a high-level software module concept (the service) for which there are good definition and interface standards (Shen, et al, 2013).

3.4 Model-Driven Implementation Provides

The ability to develop new functions rapidly (an important form of agility). In SOA, model-driven implementation can be applied to service compositions as well as to software services.

4. Researches in Service Composition Architecture

Syuet et al. (2009) reviewed an automatic web service composition surveys. Their work focuses on already published surveys in AWSC, they their paper according to service combination-centric, service selection-centric, combination and selection hybrid, service discovery-centric and workflow description-centric. Portrayed introduceda toolset that permits developers with fast create existing web benefits with figure it out new, compositewebservices. Sword gives a particular data-point in the self-evident expressiveness-complexity-efficiency tradeoff that exists in web service composition. Ponnekanti et al. (2014) Present an instrument that takes under record WSMO description that will be a user-guided, intelligent media composition approach whereby Web services need aid ran across Also recommended of the clients as stated by the composition connection. The produced composition is arranged in IRS-III by our java API to dataflow coordination.

Fan etal. (2014) propose recommend another service creation technique In light of those format system in the service Scalable Network schema Furthermore manufacture those tcp/ip protocol service creation model. Song et al. (2011) propose the execution of the AI arranging subsystem in the AFLOW, supporting higher changing What's more mechanization about creation. Also we portray how an AI arranging framework (SHOP2) could make utilized for OWL-S Web service sort descriptions to AFLOW on naturally create Web benefits Also help fractional concurrency over OWL-S. Ordonez et al. (2017) Displays HAUTO, a framework equipped should create benefits naturally. HAUTO may be based On HTN (hierarchical errand networks) automated arranging and will be created from claiming three modules: An demand transforming module that transforms common dialect What's more connection data under An arranging instance, the composition module In light of HTN arranging and the execution nature's domain for focalized (Web What's more telecom) services. The mix of a arranging part gives two

fundamental functionalities: the likelihood of customizing those composition from claiming services utilizing the client setting data Also a middleware level that integrates those execution of services to execute telecom situations. Finally, An model over natural punctual cautioning service is exhibited Similarly as An experiment.

Pathak et al. (2012) proposed a new incremental approach to service composition, MoSCoE (Modeling Web Service Composition and Execution), based on the three steps of abstraction, composition and refinement. Abstraction refers to the high-level description of the service desired (goal) by the user, which drives the identification of an appropriate composition strategy. In the event that such a composition is not realizable, MoSCoE guides the user through successive refinements of the specification towards a realizable goal service that meets the user requirements.

Kona et al. (2010) Introduce framework utilizing USDL (Universal Service-Semantics depiction Language), An language for formally describing those semantics for web-services. USDL is dependent upon the Web metaphysics language (OWL) and employs Word Net Similarly as a basic foundation for Comprehension the significance about benefits. Raik.al (2013) Introduce thorough schema that used to define What's more help Exceedingly adaptable context-aware service-based business methods. Done our approach, every last one of adaptation-related tasks, from identifying an adjustment issue should discovering an answer Furthermore applying it will An running transform instance, would performed naturally at run time. We likewise demonstrate how the framework need been executed Furthermore connected to a real-world situation from those logistics area. A summary of researches and the methods that are used in service composition are presented in table 2.

Table 2. A summary of researches and the methods that are used in service composition

Researchers	Proposed work	Method
Nahvi & Habibi (2016)	Improving Service-Oriented Architecture Processes in Process of Automatic Services Composition Using Memory and QF, QWV Factor	In the proposed framework, elements affecting management of service-oriented architecture processes are organized according to a logical procedure. This framework identifies processes of this style of architecture based on requirements in service-oriented architecture processes management and according to qualitative features in this area. In the proposed framework, in addition to using existing data in the problem area, existing structure and patterns in the area of software architecture are also utilize, and management processes in service-orientated architecture are improved based on propriety of available requirements. QWV are qualitative weighted dynamic features which indicate priority of users, and QF is quality factor of service at the time of implementation which is weighted in the framework.
Mafi et al (2010)	Service Composition in Service Oriented Product Line	An approach for explain the service composition process in one of the development phases of service oriented software product line (design or implementation). In this way we separated static with dynamic service selection and composition for developing a family of SOA application.
LEE et al (2008)	DYNAMIC SERVICE COMPOSITION: A DISCOVERY-BASED APPROACH	This work presents a discovery-based service composition framework to improve the static and dynamic integration of component services, with the following key features: • devise the notion of service availability especially for composition; • develop dynamic service composition (DSC) pattern to address the issues of service availability; and • extend the Contract Net Protocol (ECNP) to coordinate service discovery, composition and invocation based on the composite pattern.
Weigand, H et al (2008)	Rule-Based Service	In this paper, a declarative and rule-driven framework to dynamic service composition, labeled “FARAO”, is introduced, while its

	Composition and Service-Oriented Business Rule Management	ramifications are further explored and illustrated with a realistic case study. The "heart-and-soul" of FARAO constitutes business rules that prescribe the way in which services can actually be aggregated dynamically into processes. The business rules are fed into the engine in a service-oriented way, that is, by a principal requesting a service delivery in accordance with given policies and by the service manager accepting this request. The business rules are maintained and updated outside the operational services. Given the platform independence offered by SOC, this can be anywhere inside or outside the company. Our current research efforts concentrate on the implementation of the FARAO framework to experiment with rule-based service composition. A topic for future research is the mapping of our business rule representation to standard business rule languages, and to define a transformation from this language to the operational FARAO environment using a model-driven engineering approach.
Lewis, G. A., & Smith, D. B. (2008)	Service-Oriented Architecture and its Implications for Software Maintenance and Evolution	This paper will first present some basic concepts related to SOA, including high-level components of service-oriented systems. It will then present the four pillars of service-oriented systems development as a set of best practices for systems development in SOA environments. Finally, it will present the implications of SOA adoption for maintenance and evolution activities and some research challenges in this area.
Scholz, A (2009)	SOA – Service Oriented Architectures adapted for Embedded Networks	This paper discusses an embedded SOA (SOA) concept based on the definition of an embedded service (Service) term and the differences to traditional Web services based SOAs. The paper describes a middleware platform that supports the execution and development of embedded network applications by employing model based code generation and a pattern based service composition model. The advantages of the approach are showcased using an application from the building automation sector, focusing on the energy management of smart buildings
Pahl, C., & Zhu, Y. (2006)	A Semantical Framework for the Orchestration and Choreography of Web Services	This concept is currently carried further to address the composition of individual services through orchestration and choreography to services processes that communicate and interact with each other. We propose an ontology framework for these Web service processes that provides techniques for their description, matching, and composition. A description logicbased knowledge representation and reasoning framework provides the foundations. We will base this ontological framework on an operational model of service process behaviour and composition.
Liegl, P. (2007)	The strategic impact of service oriented architectures	In this paper the SOA approach will be reviewed critically and the different sections affected within an enterprise will be examined. Possible problems during the transition and use of SOA will be identified. Where already possible, solutions will be provided. This paper is based on current research conducted during my PhD studies.
Katsikogiannis, G	A policy-aware Service Oriented	In this paper, we extend the ETSI functional M2M communication 3-tier architecture with the addition of the service domain and

(2018)	Architecture for secure machine-to-machine communications	Service-Oriented Computing (SOC) capabilities. The proposed service domain removes procedures handled previously by the other domains like establishing the connection of a device to the network to increase abstraction, reducing the computational load of the 3-tier architecture domains, and providing additional security enforcement on the services handled by the interaction points. The proposed SeMMA incorporates a standardized horizontal M2M service domain with the required capabilities, facilitates the evolution of the appropriate security policies, and ensures the interoperability between the device, network, application and service domains.
Asghari, P. et al (2018)	Service composition approaches in IoT: A systematic review	This paper focuses on several service composition approaches that are applied in the IoT environment based on the Systematic Literature Review (SLR) method. The aim of this study is to analytically and statistically categorize and analyze the current research techniques on the service composition in the IoT (published between 2012 and 2017). A technical taxonomy is presented for the service composition approaches according to content of the existing studies that are selected with SLR method in this review with respect to functional and non-functional aspects in service composition approaches. The functional aspect emphasizes on verifying the behavior of service composition approach and the non-functional aspect considers the Quality of Service (QoS) in IoT environment. The approaches are compared with each other according to some technical aspects such as system correctness factors in functional properties approaches, and (QoS) factors, presented algorithms, and existing platforms in non-functional approaches.

5. Conclusion

The number of publicly accessible service composition techniques and approaches are growing, so we see a fast improvement in service composition technologies this survey presented based on different techniques like Automated based service composition, Process algebra base service composition, or Petri Nets based service composition. Most of service composition techniques are automated and web based service composition in regards to its privileges in the comparisons with other techniques to enhance the achievement of composition objectives. Our future work is to assess the possibility and additionally those points of interest about the feasibility as well as the advantages and disadvantages of different service composition approaches.

References

Alexis, A. A. P., Otavio, A. S., Carpinteiro, B. G. B., Dionisio, M. L. F., Maycon, L. P., & Bruno, T. K. (2017). Planning and Execution of Heterogeneous Service Compositions.International Conference on High Performance Computing & Simulation, 2017.

Alexis, V., Krishnamurthy, C. B., & Brinda, R. C. (2017). Performance Evaluation of Dynamic Composition & Dynamic Reconfiguration in SOA Applications. International Conference on Computer, Communication, and Signal Processing (ICCCSP-2017).

Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., & Shan, M. H. P. L. (2000). Adaptive and dynamic service composition in eflow. Technical Report.

Denilson, S., Farshad, H., John, D., Enrico, M., & Roberto, C. S. P. (2014). Interactive Composition of WSMO-based Semantic Web Services in IRS-III.

Kalamegam, P. (2017). Usage of CPN Models in Web Service Compositions. International Conference on Technical Advancements in Computers and Communications.

Kim, S., Kim, I., & Sung, H. (2016). Ontology-based Open API Composition Method for Automatic Mash-up Service Generation.

Lee, C., Wang, C. Y., Eunsam, K., & Sumi, H. (2017). Blueprint Flow: A Declarative Service Composition Framework for Cloud Applications, *5*, 2169-3536.

Opengroup (2018). Retrieved September 20, 2018 from http://www.opengroup.org/soa/source-book/soa/p4.htm

Ordonez, A., Corrales, J. C., & Falcarin, P. (2014). Automated composition of convergent services based in HTN planning. *Ingeniería e Investigación, 34*, 66-71.

Poonkavithai, K. (2017). Usage of CPN Models in Web Service Compositions-A Survey. International Conference on Technical Advancements in Computers and Communications.

Saurabh, S., Ashish, S., & Deepika, S. (2015). An Approach for QoS Based Fault Reconfiguration in Service Oriented Architecture. *Procedia Computer Science, 46*, 766-773. Science Direct, ELSEVIER.

Shen, W., & Jian, Y. (2013). A service composition method based on the template mechanism in the Service Scalable Network Framework. 8th International Conference on Computer Science & Education.

SOA Manifesto. (2018). Retrieved September 21, 2018 from http://www.soa-manifesto.org

Suman, R. B., & Ugrasen (2012). Broker Based Secure Web Service Composition Using Star Topology. International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Tang, B. X., & Wang, X. H. (2014). Designing a Self-adaptive and Context-aware Service Composition System. 978-1-4799-4811-6.

Tang, X. H., Song, Z. T., & Luo, X. F. (2011). Planning Engine based on SHOP2 for AFLOW. Seventh International Conference on Semantics, Knowledge and Grids.

Virginie, G., Maude, M., Kamil, M. & Cecile, M. (2015). QoS-aware Automatic Syntactic Service Composition problem: complexity and resolution.

Wafaa, R., Yousef, H., Abdel, S. S., & Nariman, A. (2017). GA-based Optimizer for Optimizing Security and Cost in Service Compositions. IEEE 14th International Conference on Services Computing, 2017.

Yang, S. Y., Yong, Y. F., Jong, Y. K. & Shang, P. M. (2014). A Review of the Automatic Web Service Composition Surveys. International Conference on Semantic Computing.

Zhang, J., Zhang, X. Q., Chang, Y. C., & Lin, K. J. (2012). The Implementation of A Dependency Matrix-based QoS Diagnosis Support in SOA Middleware. *ICST Transactions on eBusiness, 12*.

Zhang, Y. P., Jing, Z. H., & Zhang, Y. W. (2015). MR-IDPSO: A Novel Algorithm for Large-Scale Dynamic Service Composition. Vol. 20.

Zhou, B., Yin, K. T., Jiang, H. H., & Zhang, S. (2011). QoS-based Selection of Multi-Granularity Web Services for the Composition. *Journal of Software, 6*, 366-373.