Web Service Composition and Legacy Systems: A Survey

Mohammed Said
Central Lab of Agriculture Expert Systems (CLAES)
Giza, Egypt

Osama Ismail
Faculty of Computers and Information,
Cairo University
Cairo, Egypt

Hesham Hassan
Faculty of Computers and Information,
Cairo University
Cairo, Egypt

ABSTRACT

Service Oriented Architecture (SOA) has gained considerable interest in recent years, mostly due to the advent of standards based Web services that simplify interoperability, loose coupling and reuse. One of the basic business motivations for implementing SOA today is achieving business agility, as SOA can help businesses respond more quickly and cost effectively to the dynamic and continues changes in market conditions. It can also simplify interconnection to the existing legacy systems as well as reconfiguring loosely coupled business services in a simple, fast and low cost manner. For SOA to succeed in that, it is a key issue to provide a Web service composition approach to facilitate business innovation and adapt IT to today's fast changing markets.
In this paper, we present a survey of some existing proposals about service composition approaches and provide an overview of the strategies for the modernization of the legacy system using SOA.

Keywords

Services, Web Service Composition, Service Oriented Architecture (SOA), Legacy Systems.

1. INTRODUCTION

A service is a person or an organization performing some work for another person or organization. Service-Oriented Architecture (SOA) is defined by the organization for the advancement of structured information standards (OASIS) as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations" [1].
Web services can be considered applications that are wide spread and self directed. They can be found, connected and interacting together through the web. The maximum value of using web services lies into gathering them in great applications whose business roles can be used and presented through services. Service consumers can build powerful applications and huge systems by taking the benefits of web services through standard interfaces.
In 2007, the organization of Gartner Group stated that about [2] 50% of very important operational applications had been built using SOA. This percentage has been grown by 2010 up to 80%. In addition, according to Forrester Research report [3] in 2009, it was stated that about 75% of information technology administrators working at Global 2000 organizations planned to use SOA. And only less than 1% has been reported negative reactions on experiencing SOA. All of the above make the implementation of SOA very considerable goal for the decision makers of information technology field. In fact, the concepts of SOA will rarely change overtime, but the implementation technologies will probably vary [4].

This paper introduces some main SOA concepts and its open issues focusing on service oriented composition approaches and its relationships with the evolution of legacy systems.

The remainder of the paper is organized as follows. Section 2 explains general idea about web service. Section 3 introduces the different approaches used to compose service oriented and the main problems that face the composition process and then explanation to legacy system evolution and strategies for the modernization of the legacy system using SOA are exposed in section 4. Finally, the concluding remarks as well as SOA challenges and open issues are drawn in section 5.

2. WEB SERVICES

Web services are software components that communicate using standards-based Web technologies. Since they are based on open standards such as HTTP and XML-based protocols including SOAP and WSDL, Web services can be considered hardware, programming language and operating system independent.
Services can be described using specific service description languages such as Web Services Definition Language (WSDL) and Business Process Execution Language (BPEL). Also they are published and discovered according to predefined protocols like Simple Object Access Protocol (SOAP), and combined using an engine that organizes the interactions among collaborating services. WSDL is an XML-based language which defines the interface displayed by web service in order to be invoked by other services. WSDL provides a function-centric description of web services containing inputs, outputs, and exception handling. BPEL is an XML-based language supporting process oriented service composition [5] [6].
SOAP is considered a platform- and language-independent communication protocol that defines an XML-based format for web services to exchange information over HTTP by using remote procedure calls.
According to the W3C (World Wide Web Consortium) a Web services is "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards"[7].

3. SERVICE-ORIENTED COMPOSITION APPROACHES

Using an approach, that applies standard programming languages to link components of a Web service, overcomes the variety of middleware platforms that are being used. So there is a great need for developing Service Composition Middleware in order to make service composition in means of abstractions and infrastructure. A service composition middleware needs full and well description of the web service features which are functionality, interfaces and protocols it supports. Web service components are considered system- and vendor-specific. This can be clear when using Workflow Management Systems (WfMS) which are highly flexible and generic but they need the web service components to be familiar with WFMS API. Therefore web service components require more additional development effort. [8]

The Main Problems related to web services composition:

- how to specify them in a formal and expressive enough language,
- how to compose them (automatically),
- how to discover them through the web,
- And how to ensure their correctness.

There are varied approaches to compose a service (see Figure 1), and they are described in details in the following subsections.

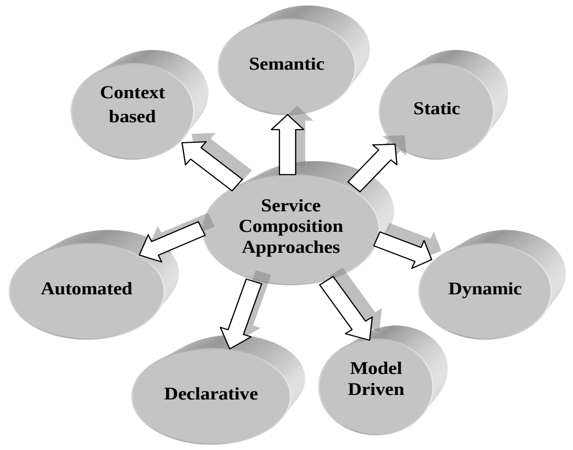

Figure 1: Services Composition approaches

3.1 Static Service Composition

Static Service Composition is done by choosing the needed components, linking them, and at last compiling and deploying these components. This approach is used when business partners and service components does not or only rarely change. There are two main approaches for the static service composition: [9]

1. Web service Orchestration: It depends on merging available services by adding a central controller, which is called the orchestrator that is responsible for firing and combining the single sub activities. Among the orchestration languages (e.g. BPML [10] and BPEL [5]).
2. Web service choreography: In this approach, the overall activity is achieved by the composition of peer-to-peer interactions among services that are working together.

3.2 Dynamic Service Composition

Dynamic composition makes the service environment highly flexible and dynamic. New services become available daily and the number of service providers is regularly increasing.

Achieving to the customer requirements and keeping with the changes done to the environment with the minimum involvement of user are considered the sign of the ideal service processes [9].

There is a big challenge problem which is how to compose services automatically. The services can be combined to perform a specific task that the existing services can not accomplish. According to the previous words, the dynamic composition of services is so important and useful but the automation process of it is still under research. The main problem in the automation process is the big gap between the concepts used by people and the computer interpretation to data. [8]

Web services' dynamic composition needs two important things; the location of services depending on their capabilities, and detecting which of the previous located services can be used to formalize service composition matching [11]. This difficulty can be controlled using semantic web technologies, for example OWL-S which is ontology, within The Ontology Web Language (OWL) based framework of the Semantic Web, for describing Semantic Web Services.

OWL-S (previously known as DAML-S) (see Figure 2); is a service ontology that enables automatic service discovery, invocation, composition, interoperation, and execution monitoring [12]. OWL-S forms services using three way ontology:

- Service profile: describes what the service requires from users and what it gives.
- Service model: illustrates the workflow of the service.
- Service grounding: gives tutorial on how to use the service.

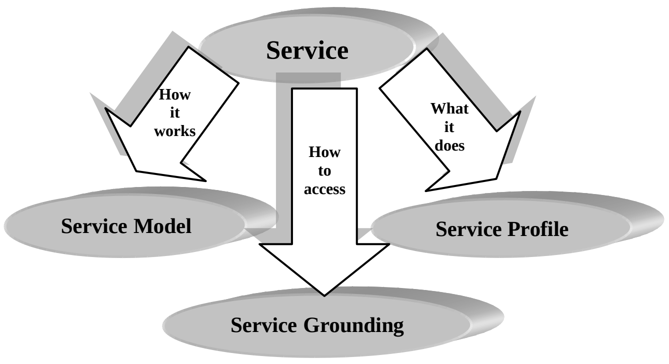

Figure 2: OWL-S Service

3.3 Model Driven Service Composition

This approach [8] is based on dynamic service composition as it assists the management and development of dynamic service compositions. The framework of this approach contains a service composition manager (SCM) which helps the user in developing, executing and managing service compositions and also contains service composition repository that maintains composition elements and rules [13]. The process of service composition development is subdivided into four phases: service definition, scheduling, construction and execution.

Firstly, SCM collect the requirements of the user and deliver them to the definer to determine preliminary composition.

Then, service definer invokes the composition engine to require the rule repository. If a specific rule is already found while composition, the repository of composed elements sends the made actions back to the service definer. Secondly, the service scheduler provides some possible compositions alternatives and allows the user to select one of them. The selected alternative is then passed to the service constructor which its mission is generating the suitable software for service execution. Finally, the service executor follows up the execution process of the service.

3.4 Declarative Service Composition

In this approach, the services are composed temporarily to get the user requirements. Two phases are included in the declarative approach: first one, it makes a start point to work with containing primary situation and the final required goal then it creates set of suitable common plans. Second phase picks one possible plan, finds out proper services and sets their workflow.

This approach [8] contains three layers; a conversation layer, a functionality layer and a database management layer. The conversation layer mission is to state the order of exchanging messages via conversation protocol. The functionality layer contains two components, raw application and a filter that makes analysis on the input information of an operation inside the raw application.

3.5 Automated Web Services Composition

Ontology based service composition is another name to the automated composition approach. Ontology is a collection of Web services that share the same domain of interest. To organize Web services into ontologies, DAML-S (DARPA Agent Markup Language for Web services) is used to support mechanisms for this job. [8]

Web service environment can be characterized by three features [14]: exploratory, volatile and dynamic. Exploratory implies that when a certain service is needed, it is called during the runtime. And volatile means that the service can be invoked at certain time but it is not available on any other time. But dynamic represents coverage of the web service changes even over time.

3.6 Context based Web Service Discovery and Composition

The context manager uses some explanation that has information of service providers, devices, and networks. Both service providers and consumers are accessing the context manager. It is linked to an interaction enabling platform via an adaptive channel [15]. This interaction passes requests to the platform which retrieves the services matching user requirements and performs the composition based on the adaptation rules. [8]

The word "context" is defined as "the kind of information that makes information services aware of their current context". There are four components that manage context information [8]:

- Web Service: have full control over the context information. They decide how the information influences their execution and their replies.
- Context plug-ins: are programmed in Java and installed at each local host. Each plug-in is associated with one context type.
- Context services: are associated with one context type and must be available over the Internet.
- Clients: Those who use the services.

3.7 Semantic Web Service Composition

The semantic web can give services description at the process level, and also provides functional information, forming the preconditions and post conditions of the process in order to evaluate the growth of the domain. Semantic web depends on ontologies to structure the domain concepts that are shared between the services [16] [17] [18].

Therefore, semantic web can result in practical and powerful applications that depend on annotations and inference engines to help into composing, discovering, executing, and interoperating web services automatically. [8]

Composition of web services needs a semantic description for services for easy interaction among them. WSDL does not provide any semantic description to Web services. (see Figure 3) developed by [19] presents WSDL features in white ovals and those added by semantic descriptions in grey filled ellipses. Also this figure shows the directions and multiplicity information to describe the relations between the entities.

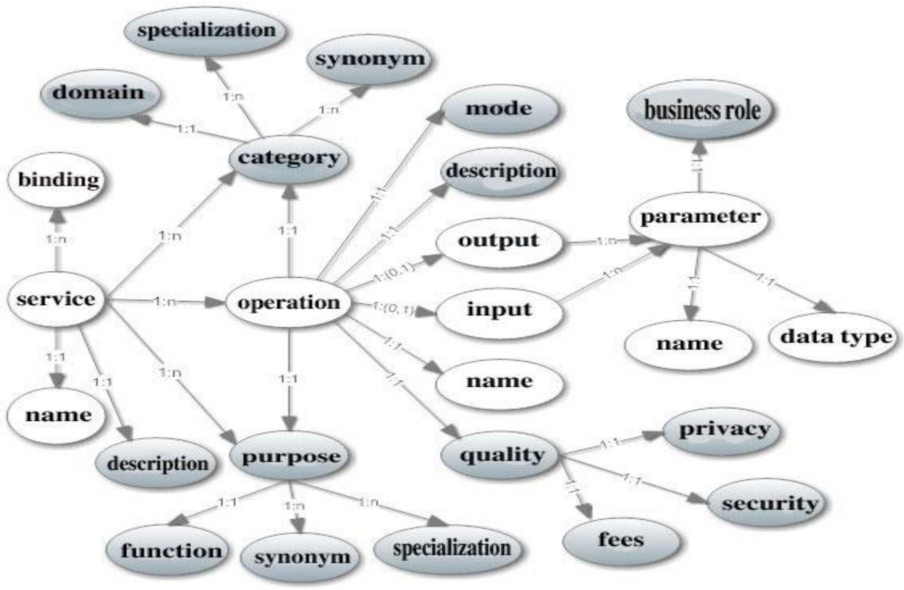

Figure 3: Ontology based description of web service [19]

4. LEGACY SYSTEM EVOLUTION

According to the large amount of information contained within companies, this increases the complexity of the legacy systems which store this information. Moving to SOA platform can help in handling this increase.

Keeping the debugging of and various modifications done to the legacy system along several years is very important to the process of transformation. SOA has many features that make the transformation of legacy systems is very desirable in today's world. These features include loose coupling, abstraction of underlying logic, agility, flexibility, reusability, autonomy, statelessness, discoverability and reduced costs. Providing a data bridge between incompatible technologies is considered an important advantage to using SOA with legacy systems.

There are four strategies for the modernization of the legacy system using SOA; replacement, wrapping, redevelopment, and migration [20]. They are discussed in the upcoming sections.

4.1 Replacement Strategy

Replacement strategy main idea is to rebuild the whole legacy system from scratch. There are two advantages for using replacement strategy; first if the legacy system uses technologies not recently being used so they are hard to preserve them, second if the business rules in the system application is realized.

To meet precisely the organization’s needs, new building of the application is the perfect option. But this process is very money and time consuming. Replacement strategy uses two strategies; big-bang strategy or incrementally. Incremental strategy is used only when the legacy system has clear structure. [20]

4.2 Wrapping Strategy

Wrapping strategy helps legacy component being easily reached by components use other software through providing new SOA interface like WSDL. Wrapping woks mainly on the interface of the legacy system neglecting the difficulty of the internal parts, so it is considered black-box transformation technique. [20]

Wrapping can be considered good, fast and cheap solution if the code of the target legacy system has great value and is written in qualified manner.

This strategy keeps the main features of the integrated legacy applications so the problems exist in these applications remain the same even after modernization and this is the major problem of wrapping. Using white-box tools in this type of legacy systems will be more helpful within transformation to get more details about the internals studying.

4.3 Redevelopment Strategy

Redevelopment means reengineering. Reengineering is considered the modification and analysis of an application to be presented in some different and new view. Reverse engineering, redesigning, restructuring, and re-implementing are considered some activities of the reengineering process. There are three main issues in service-oriented reengineering: service identification, service packaging, and service deployment. Identification of services from a legacy system is a complex mission. It is one of the main activities in the modeling of a service-oriented solution, and therefore errors made during identification can flow down through detailed design and implementation activities that may require multiple iterations, especially in building composite applications. Service packaging is a detailed description of a service that is available to be delivered to customers. And service deployment refers to service selection and service composition to satisfy functional and quality of service (Qos) requirements. [20]

Software reengineering is a very important part in the transformation process to the service-oriented environment. Reengineering can be applied to only legacy systems which have the following features:

1. The legacy system needs to be transformed to a distributed environment and can be converted and divided as a Web Service.
2. The legacy system functionality is reusable and consistent and has worth business logic.
3. The legacy system is hard to be maintained as a whole, and it is easier to maintain only some of its components.

4. The legacy system functionality has more benefit if represented as separate services.
5. Target components need to run on different platforms or vendor products.
6. The legacy system components should be implied step by step without affecting the service consumer.

4.4 Migration Strategy

Migration strategy is some close to wrapping and redevelopment strategies in the identifying, decoupling, and extracting of the legacy system code. And it likes reengineering strategy in making new interfaces matching with SOA structure. Therefore, migration strategy includes features of both redevelopment and wrapping strategies aiming at producing system that has enhanced compatible SOA design. Usually migration techniques are too similar to wrapping and redevelopment techniques. So, migration refers to transforming the whole legacy system to the new environment. [20]

4.5 Comparison of Modernization Techniques

In table 1 we have gathered some different research techniques that handle different modernization strategies that are explained in previous section. These techniques are described and compared according to the following criteria. [20]

- **Legacy System Type**: The kind of system to which the technique applies.
- **Degree of Complexity**: Time/cost complexity of the method (or NA, if not reported).
- **Analysis Depth**: The strategy used to analyze the legacy system to understand its concepts and locate the important functions to be exposed as part of SOA architecture. The analysis could be shallow or deep depending on the strategy used. Minimal dependency on the existing legacy system components in achieving SOA architecture increases flexibility.
- **Process Adaptability**: How well the process adapts to the legacy system to minimize the amount of the required modifications.
- **Tool Support**: To what degree is the process automated, and if a tool is proposed or implemented.

Table 1: Comparison of modernization techniques

Tech. Name	Modern. strategy	Legacy System Type	Degree of Complexity	Analysis Depth	Process Adaptability	Tool Support
Sneed [21][22]	Wrapping	Legacy programs	NA	Depends on business rules in the legacy code	Code stripping	Yes
Canfora et al. [23]	Wrapping	Interactive legacy system	NA	Use cases of legacy app.	NA	No
Chung et al. [24]	Redevelopment	Interactive legacy system	Moderate	Reverse software engineering & forward soft. eng	Reverse software engineering	Yes
Distante et al[25]	Redevelopment	Windows stand-alone application	Time consuming	Design recovery & forward design methods	Web transaction & navigation mode	Yes
Cuadrado et al. [26]	Redevelopment	Dependent	Dependent	Detailed description of legacy system	NA	Yes (Eclipse TPTP & Omondo UML)
Lewis et al. & Smith [27]	Migration	Program Independent	Depends on legacy system	Architecture reconstruction & detailed analysis of the target SOA	Legacy system characteristics, architecture, and code is gathered	Yes
Cetin et al. [28]	Migration	Program Independent	NA	Legacy system is analyzed	If change is needed, legacy components modified or replaced	Yes
Marchetto & Ricca [29]	Migration	Java Application	Moderate	UML Use Case diagram	The internal structure can be changed if needed	Yes

All the techniques have advantages and disadvantages. The wrapping approach presented by Canfora [23] is manual, and so it is the least preferred approach. It was hard to identify the degree of process adaptability for these techniques. And so it is difficult to evaluate the complexity of these approaches, since all techniques depend greatly on the size of the legacy system. Actually, while the benefits of the strategies are quite well understood, there is still no general technique that can be applied to solve all of the problems that a developer may face.

4.6 Choosing a Strategy

When choosing a strategy, there are many features should be considered. Table 2 summarizes an initial set of strengths and weaknesses for each strategy. Two or more modernization strategies can be mixed to achieve the required goal depending on the advantages and disadvantages of each strategy.

Table 2: Summary of modernization strategies

Strategy	Advantages	Disadvantages
Replacement	- Reduce maintenance - Improve business functions	- Time consuming - Expensive - Experienced resources needed
Wrapping	- Fast	- Inflexible - Difficult maintenance
Redevelopment	- Increase agility - Flexibility - Reduced cost	- Source code needed - Original requirements needed
Migration	- Stable environment - Tools availability	- Time consuming - Experienced resources needed - Source code needed

In fact, there is no perfect solution to the problem of modernizing a legacy system. The choice of strategy depends mainly on the goals for the SOA architecture, the available budget and resources and the time needed to complete the project.

5. CONCLUSION

5.1 Fundamental Results

After huge comprehensive study about Service-Oriented Architecture (SOA), Service composition faces increasing interest in making much research effort to support the existence of a global component market to enforce widespread reusable software.

Nowadays, using third-party services becomes the new business model. So there have been great needs to support for service usability from a design and an adoption point of view.

This includes areas of SOA governance, SOA adoption, and development processes and practices for service oriented systems development. SOA implementations within enterprise IT systems are used to access data that resides in legacy systems. So applications are capable of interacting with standard web services in a traditional request response pattern. In our paper we reviewed some topics related to SOA, such as use of semantics for service discovery and composition, in which there are significant efforts in the research community. If SOA is to be used in advanced ways, significant research topics need to be addressed in areas as design for context awareness, service usability, federation, automated governance and runtime monitoring and adaptation, dynamic service discovery and composition, real time applications, and multi organizational implementations. In our future work, we account for new research and for promising challenges hoping this document provides SOA researchers in many fields.

According to many research efforts, the reality is that SOA remains the best solution available for systems integration and modernization of legacy systems.

5.2 Open Issues

Recently, there are many potential research topics for SOA that need to be solved. Among those needs, Reusability of services and, maintenance and evolution in dynamic, heterogeneous systems. In the upcoming paragraphs we are handling the most popular research points related to those specific topics.

SOA brought new chances to improve the development of reusable components. However, there are still many challenges that need to be overcome. Although some initial solutions have been proposed to make services more reusable, there are still many points need to be covered as follows: [30]

1. Standardize service specification languages.
2. Implement high performance service registry and service discovery.
3. Provide rigid contract negotiation tools.
4. Implement dynamic service binding.
5. Develop strategies where the service execution should be located.

Another point of view in SOA new research issues that require to be addressed is related to Service-oriented systems which are significantly different from traditional systems. These differences include:

- The diversity of service consumers and service providers.
- Shorter release cycles because of the capability of rapidly adapting to changing business needs.
- The potential to leverage legacy investments with potentially minimal change to existing systems.

What does maintenance and evolution look like in this dynamic, heterogeneous, and potentially distributed development and maintenance environment is a very important question. We have identified a set of research topics that we believe would help to find answers to this question that are shown in the following [4]:

1. *Tools, Techniques, and Environments to Support Maintenance Activities:* The complexity of the maintenance process in an SOA environment continues to increase so some considerations should be taken. Analysis activities for service providers have to consider possible set of users. Also analysis for service implementation code has to consider direct users of the service implementation code and users of the service interfaces too. In addition to the above, release cycles between services and consumers, services and infrastructure, and consumers and infrastructure ideally should be coordinated.
2. *Multilanguage System Analysis and Maintenance*: One of the benefits associated with SOA, is the independent platform. Although, it is represented using standard interfaces, any language can be used to write the implemented service. Despite this is a great advantage, it is not easy to handle the whole system without partitioning.
3. *Reengineering Processes for Migration to SOA Environments*: SOA enables existing legacy systems to represent their functionality as services, as it has characteristics of loose coupling, published interfaces, and standardized communication model. Feasibility, risk, and cost are three main factors should be physically analyzed when migration is done onto the legacy system although the great value of it. Also the legacy code identification and extraction of services are too important
4. *Transition Patterns for Service-Oriented Systems:* SOA enables systems to be modernized incrementally and this is one of the various advantages of it. The components of legacy system are being replaced incrementally with newer components using Web Services technology. To initially access the new services, service consumers have to be modified once only if the interfaces remain stable. The minimization of “throw-away” cost and effort to provide intermediary system states is main difficulty of incremental migration.

6. REFERENCES

[1] Organization for the Advancement of Structured Information Standards, " *Service-Oriented Architecture Reference Model Technical Committee*". A Reference Model for Service-Oriented Architecture. White Paper, Billerica, MA, 2006.

[2] Lewis, G. A., & Smith, D. B., "*Service-oriented architecture and its implications for software maintenance and evolution*", In Frontiers of Software Maintenance (FoSM) IEEE, pp. 1-10, 2008.

[3] Heffner, R., *"Survey Results Show SOA Governance Improves SOA Benefit Realization*", For Enterprise

Architecture Professionals, 2009.

[4] Lewis, G. A., Smith, D. B., Kontogiannis, K., "*A research agenda for service-oriented architecture (SOA): Maintenance and evolution of service-oriented systems*", Software Engineering Institute, Carnegie Mellon University, 2010.

[5] IBM Corporation. *"Business Process Execution Language for Web Services BPEL-4WS"* (Version 1.1), 2002. http://www.ibm.com/developerworks/library/ws-bpel

[6] Curbera, F., Khalaf, R., Mukhi, N., Tai, S., and Weerawarana, S., *"The next step in Web services*". Communications of the ACM, 46(10):29-34, 2003.

[7] Booth, David, Haas, Hugo, McCabe, Francis, Newcomer, Eric, Champion, Michele, Ferris, Chris, and Orchard, David. Web Services Architecture. (*W3C Working Group Note*), 2004.

[8] Dustdar, S., Wolfgang S., "*A survey on web services composition*" International Journal of Web and Grid Services 1.1, pp. 1-30, 2005

[9] Sun, H., Wang, X., Zhou, B., & Zou, P, *"Research and Implementation of Dynamic Web Services Composition"*. APPT, LNCS 2834, Springer-Verlag Berlin Heidelberg, pp. 457-466, 2003.

[10] Curbera, F., Leymann, F., Storey, T., Ferguson, D., & Weerawarana, S., "*Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*", Englewood Cliffs: Prentice Hall PTR, 2005.

[11] Mao, Z.M., Brewer, E.A., and Katz, R.H., *"Fault-tolerant, Scalable, Wide-Area Internet Service Composition*". Technical Report UCB/CSD-01-1129, University of California, Berkeley, 2001.

[12] Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D., McDermott, D., Sycara, K., "*DAML-S: Web Service Description for the Semantic Web"*. In Proceedings of the 1st International Semantic Web Conference (ISWC), volume 2342 of *Lecture Notes in Computer Science*, pages 348-363. Springer-Verlag, Berlin, 2002.

[13] Orriëns, B., Yang, J., & Papazoglou, M. P, *"Model Driven Service Composition"*, In Service-Oriented Computing-ICSOC 2003, Springer-Verlag. Berlin Heidelberg, pp.75-90, 2003.

[14] Akram, M. S., Medjahed, B., & Bouguettaya, A, *"Supporting Dynamic Changes in Web Service Environments"*, ICSOC, LNCS 2910, Springer-Verlag Berlin Heidelberg, pp. 319-334, 2003.

[15] Baresi, L., Bianchini, D., De Antonellis, V., Fugini, M. G., Pernici, B., & Plebani, P., "*Context-Aware Composition of E-services"*. TES, LNCS 2819, pp.28-41, Springer-Verlag Berlin Heidelberg, 2003.

[16] Berners-Lee, T., Hendler, J., & Lassila, O., "*The Semantic Web*". Scientific American, 284(5), pp. 34-43, 2001.

[17] World Wide Web Consortium W3C. "*Semantic Web*", 2001. http://www.w3.org/2001/sw/.

[18] McIlraith, S. A., Son, T. C., & Zeng, H., "*Semantic Web Services*". IEEE Intelligent Systems, 16(2), pp. 46-53, 2001.

[19] Su, S. Y., Meng, J., Krithivasan, R., Degwekar, S., & Helal, S., "Dynamic inter-enterprise workflow management in a constraint-based e-service infrastructure", *Electronic commerce research*, Kluwer Academic Publishers, *3*(1-2), pp. 9-24, 2003

[20] Almonaies, A. A., Cordy, J. R., & Dean, T. R., "*Legacy system evolution towards service-oriented architecture*" International Workshop on SOA Migration and Evolution, pp. 53-62, 2010.

[21] Sneed, H. M., *"Integrating legacy software into a service oriented architecture*". In: CSMR, pp.3-14, 2006.

[22] Sneed, H. M., "*Wrapping legacy software for reuse in a SOA"*. Technical report, 2005.

[23] Canfora, G., Fasolino, A. R., Frattolillo, G., & Tramontana, P., *"Migrating interactive legacy systems to web services"*. In: Software Maintenance and Reengineering (CSMR), Proceedings of the 10th European Conference, pp. 24-36, 2006.

[24] Chung, S., An, J. B. C., & Davalos, S., "*Service-oriented software reengineering: SoSR"*. In System Sciences, HICSS, Proceedings of the 40th Annual Hawaii International Conference IEEE, pp.172c, 2007.

[25] Distante, D., Tilley, S., & Canfora, G., "*Towards a holistic approach to redesigning legacy applications for the web with uwat"*, In Software Maintenance and Reengineering, Proceedings of the 10th European Conference, pp.295-299, 2006.

[26] Cuadrado, F., García, B., Dueas, J. C., & Parada, H. A., "*A case study on software evolution towards service-oriented architecture*". In Advanced Information Networking and Applications-Workshops (AINAW), 22nd International Conference, pp.1399-1404, 2008.

[27] Smith, D.B.: *"Migration of legacy assets to service-oriented architecture environments*", In Software Engineering-Companion (ICSE), 29th International Conference, pp. 174-175, 2007.

[28] Cetin, S., Ilker Altintas, N., Oguztuzun, H., Dogru, A. H., Tufekci, O., Suloglu, S. "*Legacy migration to service-oriented computing with mashups"*. In Software Engineering Advances (ICSEA), IEEE, pp. 21-21, 2007.

[29] Marchetto, A., Ricca, F.: From objects to services: *"toward a stepwise migration approach for java applications"*, International journal on software tools for technology transfer 11(6), pp.427-440, 2009.

[30] Zhu, H., "*Building reusable components with service-oriented architectures*", Information Reuse and Integration, Conf IEEE International Conference on. IEEE, pp. 96-101, 2008.