A Survey on Issues and Challenges of Web Service Development, Composition, Discovery

SHAH Tejas R.
M.Sc. (I.T.) Programme,
Veer Narmad South Gujarat University, Surat
trshah@vnsgu.ac.in

PATEL S. V.
Sarvajanik College of Engineering & Technology, Surat
Gujarat Technological University, Gujarat.
patelsv@gmail.com

Abstract

In SOA (Service Oriented Architecture) based application implementation, Web Service (WS) development is one of the outstanding and upcoming area. For web service discovery and composition, developed and refined approaches have been established. It is extremely advantageous to give applicable, appropriate web service to the consumer based on their preferences, constraints, context with high precision, accuracy and at the right time. Many researchers and practitioners have observed issues and challenges in web service development specific to web service discovery and composition separately but nowhere both these issues and challenges have been dealt together. This paper provides an insight of unresolved issues and challenges investigated by researchers in the field of web service development specifically in composing and discovering appropriate web services.

Keywords: Service Oriented Architecture, Web service, Web Service Development, Web Service Discovery, Web Service Composition

1. Introduction

The Service-Oriented Architecture (SOA) has appeared as a significant distributed computing paradigm to support rapid, low-cost development of distributed applications in heterogeneous environments. SOA establishes an architectural model that allows services to be published, discovered, and consumed by applications or other services, the goal of which is to realize loosely coupled, platform-independent profound area of distributed computing [1]. A Web service is essentially a semantically well-defined abstraction of a set of computational or physical activities involving a number of resources, intended to fulfill a consumer requirement. A more formal definition provided by IBM is that Web services are “a new breed of Web application, and they are self contained, self-describing, modular applications that can be published, located and invoked across the Web”. The service discovery and composition are the key challenges for SOA and web services development. By composing multiple atomic or composite services into more capable and powerful enterprise applications, the true capacity of SOA can be achieved. The service composition and discovery enormously reduces the cost and risks of building new business applications by reusing the Web service.
The aim of this paper is to survey the existing techniques, methods, research prototypes, and standards on service discovery and composition. Although Web services composition has been heavily investigated, several issues related to dependability, ubiquity, personalization, among others, still need to be addressed. These challenges are pertinent to several innovative computing paradigm such as SOA, cloud computing, social computing, and Internet of Things. If these problems are not taken care of, it might hamper the adoption of the method successfully and may have negative consequences like missed schedule and overrun budget for Service Oriented applications. The study reveals that there is a wide possibility for developing new approaches and techniques in web service development to determine problems observed in various composition and discovery approaches.

The remaining part of the paper is organized as follows. In Section II, some of the important approaches and prototypes of web services composition and discovery have been reviewed. Section III examines the survey of issues and challenges of web service development, composition and discovery. Section IV provides the summary of applicability of various problems and challenges to web service development. The concluding section offer remarks on the paper.

2. Review of Web Service Composition and Discovery Approaches

1.1 Web Service Composition Approaches

The interaction between web services can be done with 2 standardized ways: Service Orchestration and Service Choreography in service composition. According to composition time, service composition could be categorized into static or dynamic. In a static composition, the aggregation of services takes place at design time. In dynamic composition service components determined and replaced during runtime. Dynamic Web services composition requires the execution system to support automatic discovery, selection, and binding of service components. The manual service composition uses BPEL (Business Process Execution Language) to compose services with human involvement. Automatic service composition typically exploits the semantic web and AI planning techniques.

The web services composition and their classification survey have been published in [2,3,4,5,6]. The number of available web services composition approaches based on simple classification such as manual vs. automated and static vs. dynamic has been reviewed in [2]. In [3], the authors discuss composition methods according to the level of automation, and [4,5,6] focuses on different standards of web service composition. The article of sheng et al. in [7] gives the exhaustive idea and analysis of existing major techniques, research prototypes, and standards on Web services composition. This work also identifies open and unresolved research challenges in this area.

In this section we present some of web service composition languages, platforms, approaches, prototypes and strategies based on automatic and semi-automatic composition view. Table 1 highlights the key points of semi-automated and automated service composition views.

<table>
<thead>
<tr>
<th>Category</th>
<th>Approach/ Prototype</th>
<th>Key Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composition</td>
<td>Self-Serv [9]</td>
<td>compositing web accessible information and business services: dynamic and peer-to-peer provisioning of composite Web services, three types of services: elementary services and composite services, and service communities</td>
</tr>
<tr>
<td>WISE [10]</td>
<td>Workflow based Internet Services: software platform for process based business-to-business electronic commerce. three service layers: database services, process services, and interface services, specify processes via a process definition tool named StructWare</td>
<td></td>
</tr>
<tr>
<td>ServiceGlobe [11]</td>
<td>Lightweight, distributed, and extensible service platform, present new techniques for Web service execution and deployment in dynamic environments, two approaches: dynamic service selection and the dispatcher service</td>
<td></td>
</tr>
<tr>
<td>Berardi et al. [12]</td>
<td>Web services are portrayed by Roman Model, exporting behavioral features by using finite transition systems, synthesized in a “just-in-time”</td>
<td></td>
</tr>
<tr>
<td>SOA4ALL [13]</td>
<td>Service Oriented Architectures for All: Provide the power, elasticity and ease for a wider uptake of SOA, automated service discovery, mediation and composition of RESTful and SOAP-based services</td>
<td></td>
</tr>
<tr>
<td>Fujii et al. [15]</td>
<td>A semantic-based, context-aware composition of request from a natural language. Component service Model with semantics (CosMoS), component runtime environment (CoRE), and semantic graph-based services composition (SeGSeC)</td>
<td></td>
</tr>
<tr>
<td>SeSCo [16]</td>
<td>Seamless Services Composition Services composition mechanism in pervasive environments. Utilizes an event-oriented middleware platform, called Pervasive Information Communities Organization (PICO)</td>
<td></td>
</tr>
<tr>
<td>AO4BPEL [17]</td>
<td>Aspect-Oriented for BPEL: Extension of BPEL with aspect-oriented workflow features and provides cross-cutting concern such as logging, auditing, security, and dynamic adaptation of web services composition at runtime</td>
<td></td>
</tr>
<tr>
<td>SCENE [18]</td>
<td>Services Composition Execution Environment: Service Centric System Engineering (SeCSE) project, methods, tools and a platform for service engineering. Extension of BPEL language with rule engine drools</td>
<td></td>
</tr>
<tr>
<td>Skogan et al. [19]</td>
<td>Uses UML Activity diagrams to model service compositions. The UML diagrams are then used to generate executable BPEL processes using XSLT transformations</td>
<td></td>
</tr>
<tr>
<td>Rao et al. [20]</td>
<td>Mixed initiative framework for semantic web service discovery(WSD) and composition, but allowing user intervention in key decisions, reasoning on OWL ontologies with GraphPlan algorithm</td>
<td></td>
</tr>
</tbody>
</table>
| Semi-Zeng et al. [21] | The formulation of service composition as a goal-directed planning problem that takes three inputs: a set of domain
### Automated Service Composition

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRS-III [22]</td>
<td>The proposed model complements the WSMO orchestration in IRS-III, a framework for Semantic Web Services based on WSMO specification. A tool based on the above model that supports a user-guided interactive composition approach.</td>
</tr>
<tr>
<td>SWORD [24]</td>
<td>Set of tools that allows developers to quickly compose existing web services to realize new composite Web services. No WSDL and OWL-S but ER model to specify web services.</td>
</tr>
<tr>
<td>McIlraith et al. [25]</td>
<td>Address automated composition and its execution for the semantic Web. Adapt and extend GOLOG a logic programming language built on top of the situation calculus, to enable processes generic, customizable and usable in the context of the Web.</td>
</tr>
<tr>
<td>McDermott et al. [26]</td>
<td>Planning Domain Definition Language (PDDL) is extended to formalize Web services, and Estimated-regression (a planning technique) planners are applied to implement automated services composition.</td>
</tr>
<tr>
<td>ASTRO [27]</td>
<td>A services composition framework, based on the concept of “Planning as Model Checking”. ASTRO toolset, consisting of WS-gen, WS-mon, WS-console and WS-animator, to support automated services composition, monitoring and execution.</td>
</tr>
<tr>
<td>OWLS-Xplan [28]</td>
<td>An artificial intelligence planner called Xplan is applied to generate the services composition plan, which takes as input the PDDL descriptions of OWL-S services and a planning query.</td>
</tr>
<tr>
<td>Majithia et al. [29]</td>
<td>Automatically construct a service composition schema from a high-level objective, abstract workflow generator, matchmaking algorithm.</td>
</tr>
<tr>
<td>Brogi and Corfini [31]</td>
<td>Service Aggregation Matchmaking (SAM), To discover service compositions, or create when no suitable composition for a given request is found. The model used to represent services in Consume-Produce-Read (CPR) Nets.</td>
</tr>
<tr>
<td>Jonathan Lee et al. [32]</td>
<td>SOAP and non-SOAP, OSGi services WS compositions, Extending BPEL engine.</td>
</tr>
<tr>
<td>PORSCE II [33]</td>
<td>An integrated system that performs automatic semantic web service composition exploiting AI techniques, specifically planning.</td>
</tr>
</tbody>
</table>
Approaches to the automated Web service composition problem have been astonishingly diverse and offer different interpretations of what should be addressed in a composition approach. They also vary on the degree of automation involved in the process ranging from semi-automated to fully automated composition process.

1.2 Web Service Discovery Approaches

The Web Service discovery is the process of finding the most suitable and appropriate service that satisfies the customer’s request with his preferences, constraints. A number of approaches, methods, prototypes have been proposed in service discovery. According to Mohebbi [35], there are three different views for discovering services, namely architectural view, automation view, and matchmaking view. This section focus on the matchmaking view which is further divided into semantic-based, syntax-based and context-aware. The authors in [36] reviewed web service discovery with two categories: Web Service discovery based on decentralized approach and centralized approach. Dong et al. in [37] concise six technical aspects from the past literature and analyzed the typical semantic Web Services matchmakers.

This section is not considering the keyword based searching, as it offers limited, irrelevant discovery in the world of semantic web. Although the plenty of approaches found, only few of them are included with two major categories: semantic based service discovery and context-aware service discovery. Table 2 highlights the key points of semantic based (ontology based) and context-aware service composition views.

<table>
<thead>
<tr>
<th>Category</th>
<th>Approach/Prototype</th>
<th>Key Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic based Discovery approach (Use of Ontology)</td>
<td>Fenza et al. [38]</td>
<td>A hybrid non-logic-based approach that combines an agent-based paradigm and a fuzzy model for SWS matchmaking in the centralized environment</td>
</tr>
<tr>
<td></td>
<td>Paliwal [39]</td>
<td>Non-explicit semantic service description, Clustering Approach of offline UDDI for functional level service categorization</td>
</tr>
<tr>
<td></td>
<td>SWSDOM[40]</td>
<td>Framework of Semantic Web Service Discovery based on Ontology Mapping (SWSDOM), Obtaining similarity by aggregating linguistic similarity, structural similarity and instance similarity.</td>
</tr>
<tr>
<td></td>
<td>Wang et al.[41]</td>
<td>Pre-processes user preferences’ cluster, User receives demanded services quickly and enhancing performance of unmixed semantic UDDI</td>
</tr>
<tr>
<td></td>
<td>Yousefipour et al.[42]</td>
<td>QoS(Quality of Service) aware framework based on broker, Service monitoring by agent and report collection of QoS real time values of WS</td>
</tr>
<tr>
<td></td>
<td>Sajib et al. [43]</td>
<td>Improving provider’s domain ontology with social learning approach, Discovery of more relevant services with consumer’s concept contributions</td>
</tr>
</tbody>
</table>
### Semantic Based Discovery Approach (Use of Ontology)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Themis-S.[44]</td>
<td>Themis-S, prototype of an ontology based natural language service discovery engine, Using WordNet as general purpose ontology, outperforming syntactic information retrieval models</td>
</tr>
<tr>
<td>SAWSDL–iMatcher [45]</td>
<td>Designed a hybrid and adaptive service matchmaker, Enabling user-customizable SAWSDL services retrieval. Supports multiple matching strategies for service discovery on the basis of service description</td>
</tr>
<tr>
<td>OPOSSUM [46]</td>
<td>A search engine, Object–Procedure–Semantics Unified Matching (OPOSSUM), for semantically retrieving and indexing Web services from the Internet. A crawler- retrieval of SWS by comparing the I/O parameters and generating an index</td>
</tr>
<tr>
<td>Calado et al. [48]</td>
<td>Semantic web service discovery with similarity metrics, Direct and indirect matching</td>
</tr>
<tr>
<td>Hatzi et al. [49] WESS</td>
<td>Web Service Search Engine, A Specialized Search Engine for Web Service Discovery- WESS, targeted to discover and retrieve web service descriptions-directed crawling</td>
</tr>
<tr>
<td>QoS-based Discovery [50]</td>
<td>QoS-based Discovery and Ranking of Web Services, optimal matching dimensionless matrix to evaluate the best candidate services</td>
</tr>
<tr>
<td>Stefan et al.[51]</td>
<td>Conceptual situation spaces aligned to established semantic web service standards. Situation characteristics as members in geometrical vector spaces, Applied to E-learning and E-business domain</td>
</tr>
<tr>
<td>Xiao et al. [52]</td>
<td>Context modelling approach which can dynamically handle various context types and values, Based on relation among context, suggestion of service from user</td>
</tr>
<tr>
<td>Liu et al.[53]</td>
<td>Fuzzy rough set theory based context-aware dynamic service matchmaking, Users’ satisfaction in pervasive environment</td>
</tr>
<tr>
<td>Toninelli et al.[54]</td>
<td>User-centric semantic service discovery with middleware AIDAS, Context-awareness based on metadata of user,device,service profile to provide tailored views on services of interest.</td>
</tr>
</tbody>
</table>

It can be analysed from the above table that semantic based service discovery minimizes the manual intervention and automatically discover web services dynamically. We can also enhance the web service development process by using ontology and annotations. Context-aware discovery provides key insights to help consumer in the ubiquitous and pervasive environment.

### 3. Issues and Challenges of Web Service Development, Composition and Discovery

#### 3.1 Common Issues and Challenges of Web Service Development

The service oriented software development is an emerging and fast-growing research area. Many researchers have contributed to solve the issues present is web service development, this area still
needs continuous attention. This session discusses the common issues and challenges which are found in web service development environment.

3.1.1 Lack of appropriate RE method

The Service Oriented Computing paradigm is lack of appropriate and widely acceptable RE (Requirement Engineering) techniques. The inherent gap between specification and service oriented description of the method has affected the other SOC activities like service discovery and composition [55][56]. Currently services are described by description languages like Web Service Description Language (WSDL), but service orientation needs its own RE activities, since a service described in terms of operations and bindings may not be enough to specify the desired goals and domain assumptions of stakeholders [57].

3.1.2 Refinement of Specifications after Service Discovery

The service discovery can be easily done with the help of UDDI, but Service Oriented Software Development (SOSD) should comprise of automated dynamic service discovery with high level language support which can also focus on inclusion service refinement.

To improve the completeness of requirement specifications, iterative discovery process is required [58], which can accommodate modified specifications. From the consumer’s side, specification refinement progress leads towards iterative discovery based on changed specifications.

3.1.3 Innovation and Creativity in Service Specification

To give required services to the consumers, SOSD requires new ideas, innovations in discovery and management of services. The service provider has to promptly update the service according to the market demand and competition. The service provider can conduct brainstorming techniques and workshops to make tangential reference for creative thinking in service development [59].

3.1.4 Customer Acceptance on Service Change

The consumer’s requirements appear progressively while using service in practice. It is a challenging task to manage the evolved services because shallow changes are localized but deep changes on services may have cascading effect on the other enterprise services or business partner services [60]. There should be redesign and redeployment of the services when consumer’s specifications will evolve over time. The customer acceptance is highly preferable before deploying the service to its final stage.

3.1.5 Clustering of Services as per Requirement

The web services can be grouped together according to their respective domain and area. The services can be assigned to a desired category with the help of clustering process. This process will reduce the searching and discovery time and increases the domain knowledge of the stakeholder. The SOSD requires a excellent knowledge management strategy for clustering of services [61].

3.1.6 Identifying Business Process at Service Design

Business processes of the organization can be easily converted into services. Identification process of services by their business goals and intentions is a challenging task. Majority of the current
service description methods fail in describing business processes in detail [62]. The developed web services should conform to the business requirements.

3.1.7 **Inadequate Web Service Management**

The WS architecture should have the proper monitoring and management of the existing and published web services. The architecture should provide the optimized and configured environment where services must configure themselves automatically (Self-configuring), can be easily discover, diagnose and malfunctions and anomalies in the system [63]. It should support the policy based corrections (Self-healing) and also offer the self protecting and self optimizing feature. The inadequate management of web services poses new issues of adaptability and interoperability.

3.1.8 **Service Governance**

The web services are designed and used by different business enterprises whose business processes are framed across organizational boundaries. They should manage web services as per the standards and compliances. Service governance is a challenge issue, as services must meet the functional and QoS objectives with given context of the business unit. [63]

3.1.9 **No Universal Registry**

There was no consensus regarding ownership of the root UDDI registries. Public registries that were run by IBM, Microsoft, SAP have been no longer available currently [64]. It is difficult to check the performance, scalability and statistical gathering of data due to non-existence of universal registry for web services.

3.1.10 **UDDI: Storing Limited Attributes**

UDDI is not innately design to store and publish QoS requirements of the web services. In addition UDDI allows search facility on limited attributes of the service like Service name, key Reference [65]. This problem makes it critical to store run-time parameters like performance, availability, throughput of the web services.

3.1.11 **WSDL: Not Designed for NFR**

WSDL is inherently designed to give functional aspects of web service like Service Type, Ports, Port type and binding parameters [66, 67]. It is not designed for considering Non functional aspects of the WS, which makes it difficult to store Quality of Service (QoS) parameters like availability, reliability, throughput, price etc.

3.1.12 **Adequate Level of QoS**

Different set of QoS metrics can be applied at run-time as well as design-time. The configuration of web service can change during run time. Besides, applications built on web services should provide features like required performance, reduced costs, faster response time to consumer; effective resource usage and throughput to service providers [68]. By inserting QoS properties at adequate level at run time supports the dynamism of application and satisfy the provider and consumer of service.

3.2 **Issues and Challenges of Web Service Composition**
Moreover, the rapid rise of new computing paradigms such as cloud computing, social computing, and Web of Things also presents compounded challenges in this area [69]. In [7] Sheng et al. discovered the open issues and presented the overview on prototypes, open platform for service compositions. In this section, we identify several challenges on services composition.

### 3.2.1 Composition Correctness

The verification of the compositional correctness is missing in most of the approaches of service composition. In order to verify the correctness, compositional specifications have to be modelled into mathematical formalisms like pi-calculus, process algebra such that the composition process can include deadlock detection and consistency feature. The easier way is to include compositional correctness features in the composition algorithm itself [70,71].

### 3.2.2 Transactional Support

Due to the innate autonomy and divergent capabilities of web services, maintaining the transaction support is challenging task. The service composition requires an advanced transactional management for reliable, consistent and recoverable composition process. Very few approaches like WebTransact [72] appear to be better than others since transactional model is supported [70,71].

### 3.2.3 Dependable Service Composition

The reliable and dependable services composition remains a significant challenge [73,74]. The service developers should able to check the soundness and completeness of compositions to identify the problems at early stages of composition specifically for mission critical applications. Web services from different domain and organization may have varying constraints and features. The composition process should include domain knowledge, constraints, preferences, and policies to solve domain dependency [75].

### 3.2.4 Adaptable and Autonomous Service Composition

The today’s web environment demands more adaptable and flexible service composition approach. Autonomous services composition is a promising research effort to increase self-configuring, self-optimizing, self-healing, and self-adapting features in composition [7,76]. The Web services from different enterprises are often incompatible and require service mediation to handle adapt services efficiently [77].

### 3.2.5 Service Composition in Pervasive Environment

The profusion of ubiquitous, interconnected computing devices made web services available to mobile users [78,79]. Composing services across several mobile devices in such an environment presents fresh challenges like addressing context awareness, heterogeneity and contingencies of devices, personalization requirement of users. The ubiquitous and mobile devices running in pervasive environment are usually resource constraint, special considerations are necessary for the efficiency and performance of composite services.

### 3.2.6 Composing SOAP, non-SOAP and non-web services

To invoke and compose a combination of SOAP, non-SOAP, and non-web services into a composite process is difficult one. There are two major exertions: one is how to invoke and
compose heterogeneous web services with diverse protocols and content types, and how to integrate non-web services in composition process. This issue can be resolved with extended BPEL engine with adapters [32].

3.2.7 Dynamicity in AI Planning based Composition

The generated AI plan produces the static composition schema without exploring the run-time composition. It is challenging to explore how dynamic composition at run-time via re-planning can be applied to planning techniques that support most of the features, such as planning as model checking [80].

3.2.8 Business-driven Automated Composition

The business process can be mapped to the web service. SOA applications should form the business logic at the abstraction level. The integration and composition process should be carried out in such a manner that automatically includes the business requirement and processes [63].

3.2.9 Non-uniformity between Services

The web services are of different formats for exchanging data over the web. In addition, domain specific concepts lead to the non-uniformity in publishing data to the registries [81,82]. However to make all the services uniform is not feasible, heterogeneity and non-uniformity may cause immense problem while scavenging web services.

3.3 Issues and Challenges of Web Service Discovery

There are plenty of sub approaches invented for syntactical, semantic and context aware web service discovery. Finding most suitable and relevant web services has obstacles in the path of discovery process. Some of the issues are resolved with the objective of enhancing the process, but new challenges are evolving as increase of web services and usage is countless. This session review some the critical issues and challenges concern to web service discovery.

3.3.1 Efficiency of Matchmaking Model

The matchmaking model works as the coordinating agent to match services against all published web services in the registry. The user’s request and matchmaking model are highly interrelated and dependent and requires the design of matchmaking algorithm. The matchmaking system ought to support input-output through the repository and enable features of service browsing, modification and cancellation [83]. The service matchmaker shall automatically detect the language and type of parameters for describing the request and finds an appropriate service matching strategy. The SAWSDL-iMatcher [84] allows user to select matchmaking strategy from request parameters.

3.3.2 Formal Request Language for WSD

The service requesters should describe their requirements in an unambiguous and machine-interpretable form. There should be the pre-processing of request. The semantic annotations are added in DARPA, Web Ontology Language or other semantic markup language. Apart from the advantages of semantic web service discovery, the user’s request mapping with the ontology demands development of formal request language [63]. A web service request language (WSRL) is presented in [85] for web-service interaction based on planning and constraint satisfaction.

3.3.3 Intimate Knowledge of Semantic Web Services and Ontology
Some of the web service discovery methods are describing the user request in languages like OWS-S, WSMO, or WSDL-S. The user should have the intimate knowledge of semantic web service description and implementation [86]. To specify a web request, the user should have the familiarity and knowledge of ontology [87]. The manual annotation process requires skills in ontology engineering which is a quite elevated requirement for regular consumers.

3.3.4 Handling Heterogeneity of Ontology

In open and dynamic environment, different ontologies are developed for sharing the knowledge. Moreover, service providers and requesters may have a different understanding of shared knowledge. The ontologies used for annotating the same services and request may surface this problem [88]. The ontology mapping techniques are required to provide coordination between different ontology [86]. However, Web Service Modelling Ontology (WSMO) is exception to provide mediator-centric solution to this problem.

3.3.5 Irrelevant Service Retrieval

Although the keyword based syntactic service discovery gives less burden on service providers, it is notoriously prone to low precision and imperfect recall [89]. Many completely irrelevant items can include the query’s keywords, leading to low precision and results in irrelevant services. The approach should have the process based queries or deductive approach to offer relevant and appropriate service retrieval.

3.3.6 Imprecise Discovery Mechanism

If the similarity of two service descriptions is simply and syntactically measured by comparing their .wsdl documents (structure of the service: service name, operation name, input parameters, output parameters), the result of service discovery likely to be imprecise or even erroneous due to lacking semantic support. The syntax based service discovery faces this imprecise discovery problem [90].

3.3.7 Index-based Mechanism in Matchmaking

In real web environment, where heterogeneous and dynamics services are registered in numerous geographically dispersed and non-interrelated service registries, index-based web service discovery mechanism is preferable [37]. The Google™ find services automatically by using crawlers. However, very less attention is given in the area of index-based SWS discovery.

3.3.8 Design of Specialized Web Service Search Engine

The global UDDI registries are no longer available and discovery through conventional, general purpose search engines does not yield satisfactory results based on keyword search. Searching WS descriptions from typical search engines has many disadvantages like lack of annotation and tag, No provision of search parameter specification, lack of NFR search and vast amount of irrelevant results. For these reasons, specific purpose efficient search engine should be designed. The work presented in [49] concerns the design of search engine for web service descriptions, both semantic (OWL-S and SAWSDL) and non-semantic (WSDL).

3.3.9 Binding of Correct Web Service Version
The web service discovery algorithm must be able to handle multiple versions of the web services present in the registry. By adopting the incremental model, the basic attributes describing the service like port, port types and binding will not change. But the underlying web method implementations may change [82]. The matchmaking algorithm must be able to bind the correct and latest version of web service.

4. Summary and Analysis of the Survey

Every Web Service composition and discovery method is having its own strength and weakness and developed with objective of composing and giving right services at the consumer’s desk. After the analysis and observation of the survey, an applicability table (Table 3) is created which maps the challenges and issues to web service development and interlink it. Issues and challenges are categorized as per their applicability with phases of service development. The comment in the cell shows the reason for its inclusion. The symbols of the applicability are:

√ - Applicable, ⌐ - Partially Applicable, ≠ - Not Applicable

Table 3 Issues and Challenges in WS development, WS discovery, WS composition

<table>
<thead>
<tr>
<th>No.</th>
<th>Issues and Challenges</th>
<th>Web Service Development</th>
<th>Web Service Composition</th>
<th>Web Service Discovery</th>
<th>Inter dependency of Composition and Discovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lack of appropriate RE technique</td>
<td>√ Gap of specification of requirement and service oriented description</td>
<td>√ Composition as per the proper Requirement s</td>
<td>√ Discovery of WS from specified requirements</td>
<td>√ Impact on both the processes</td>
</tr>
<tr>
<td></td>
<td>Service Governance</td>
<td>√</td>
<td>√ Compliances required</td>
<td>√ Compliances required</td>
<td>≠</td>
</tr>
<tr>
<td>---</td>
<td>-------------------</td>
<td>---</td>
<td>------------------------</td>
<td>------------------------</td>
<td>---</td>
</tr>
<tr>
<td>8</td>
<td>No Universal Registry</td>
<td>√ Requires universal and public repositories</td>
<td>≠</td>
<td>√ Discovery from UDDI</td>
<td>√</td>
</tr>
<tr>
<td>9</td>
<td>UDDI: Storing limited attributes of service</td>
<td>√</td>
<td>√</td>
<td>√ Discovery with desired attributes</td>
<td>√</td>
</tr>
<tr>
<td>10</td>
<td>WSDL not designed for NFR</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>¬</td>
</tr>
<tr>
<td>11</td>
<td>Composition Correctness</td>
<td>¬</td>
<td>√ Specification with mathematical model</td>
<td>¬</td>
<td>¬</td>
</tr>
<tr>
<td>12</td>
<td>Transactional Support</td>
<td>√ Overall transaction effect</td>
<td>√ Interaction amongst services requires transaction support</td>
<td>¬</td>
<td>¬</td>
</tr>
<tr>
<td>13</td>
<td>Dependable Service Composition</td>
<td>¬</td>
<td>√ Dependency on services fetched from different domains and organizations</td>
<td>¬</td>
<td>¬</td>
</tr>
<tr>
<td>14</td>
<td>Adaptable and Autonomous Service Composition</td>
<td>√</td>
<td>√ Self-configuring, Self-optimizing, Self-healing</td>
<td>¬</td>
<td>¬ Adaptable for composition</td>
</tr>
<tr>
<td>15</td>
<td>Service Composition in Pervasive Environment</td>
<td>√ Web services application for pervasive system</td>
<td>√ Context-awareness, contingencies and heterogeneity support</td>
<td>¬</td>
<td>≠</td>
</tr>
<tr>
<td>16</td>
<td>Composing SOAP, non-SOAP and non-Web Services</td>
<td>¬</td>
<td>√ Composing combinations of SOAP, non-SOAP and non WS</td>
<td>≠</td>
<td>√</td>
</tr>
<tr>
<td>17</td>
<td>Dynamicity in AI planning based</td>
<td>¬</td>
<td>√ Run time re-planning</td>
<td>¬</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Business-driven automated composition</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√ Discovery and composition should be business aware</td>
</tr>
<tr>
<td>19</td>
<td>Index-based mechanism in matchmaking</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Service discovery through indexing</td>
</tr>
<tr>
<td>20</td>
<td>Adequate Level of QoS</td>
<td>√ QoS metrics implementation</td>
<td>√ Security, privacy, performance, throughput while composing</td>
<td>√ Security, privacy, performance, throughput while discovering</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Efficiency of matchmaking model</td>
<td>√ Makes WS development more fruitful</td>
<td>√</td>
<td>√ Matchmaking algorithm for discovery</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Formal Request Language for WSD</td>
<td>√ Use of request language in development</td>
<td>√</td>
<td>√ Accurate discovery and composition with service request language</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Intimate knowledge of Semantic WS and Ontology</td>
<td>ـ</td>
<td>√</td>
<td>√ Semantic based WS discovery</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Irrelevant Service Retrieval</td>
<td>√</td>
<td>ـ</td>
<td>～</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Handling Heterogeneity of Ontology</td>
<td>～</td>
<td>～</td>
<td>～ Applicable to semantic WSD</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Imprecise discovery mechanism</td>
<td>～</td>
<td≯</td>
<td>√ Syntax based discovery</td>
<td>～ Required syntax and semantic based WS discovery</td>
</tr>
<tr>
<td>27</td>
<td>Design of Specialized WS search engine</td>
<td>√ Search engine with UDDI, Crawling features</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Non-uniformity between services</td>
<td>√</td>
<td>√ Interoperability</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Binding of correct web service version</td>
<td>√</td>
<td>√ Updating version when existing WS</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>
5. Conclusion

After observing the existing trends in web services discovery and composition development, certain issues still remain unsolved. There is an increasing need to develop service discovery and composition methods after resolving the challenges and issues present in the world of web services development. This scrupulous survey would help academician and researchers to get a strong foothold on the realm of service composition and discovery and problems associated with it. The major analysed challenges like design of specialized search engine for WS discovery; business context and QoS aware discovery and composition; composing non-SOAP and non-web services with scalability and adaptability will enhance the service-based development benefiting service consumer and provider.

References


[22] Frashad, Denilsorn, Semantic Web Service Composition in IRS-III: The Structured Approach


[34] Amal Elgammal, Using UML to Model Web Services for Automatic Composition, IJSE, 2010
[64] Holger Lausen and Thomas Haselwanter, 2007, Finding Web Services, In the Proceedings of European Semantic Technology Conference(ESTC 07)
[69] Athman Bouguettaya, Quan Z. Sheng, Florian Daniel (Eds.), Advanced Web Services, Springer, 2013


