Engineering Applications of Artificial Intelligence 53 (2016) 86–104

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Artificial intelligence in service-oriented software design

Guillermo Rodríguez *, Álvaro Soria, Marcelo Campo

ISISTAN Research Institute (CONICET-UNICEN), Campus Universitario, Paraje Arroyo Seco. Also CONICET – Argentina, B7001BB Tandil, Buenos Aires, Argentina

ARTICLE INFO

Article history:
Received 29 April 2015
Received in revised form
10 February 2016
Accepted 28 March 2016
Available online 19 April 2016

Keywords:
Artificial Intelligence
Service-oriented design
Web services discovery
Web service composition
Web service development

ABSTRACT

Service-Oriented Architecture (SOA) has gained considerable popularity for the development of distributed enterprise-wide applications within the software industry. The SOA paradigm promotes the reusability and integrability of software in heterogeneous environments by means of open standards. Most software companies capitalize on SOA by discovering and composing services already accessible over the Internet, whereas other organizations need internal control of applications and develop new services with quality-attribute properties tailored to their particular environment. Therefore, based on architectural and business requirements, developers can elaborate different alternatives within a SOA framework to design their software applications. Each of these alternatives will imply trade-offs among quality attributes, such as performance, dependability and availability, among others. In this context, Artificial Intelligence (AI) can assist developers in dealing with service-oriented design with the positive impact on scalability and management of generic quality attributes. In this paper, we offer a detailed, conceptualized and synthesized analysis of AI research works that have aimed at discovering, composing, or developing services. We also identify open research issues and challenges in the aforementioned research areas. The results of the characterization of 69 contemporary approaches and potential research directions for the areas are also shown. It is concluded that AI has aimed at exploiting the semantic resources and achieving quality-attribute properties so as to produce flexible and adaptive-to-change service discovery, composition, and development.

1. Introduction

Service-Oriented Architecture (SOA) has been chosen by the software industry for building distributed and enterprise-wide software applications, which should be both high-quality and adaptable to market changes (Erickson and Siau, 2008). Moreover, SOA creates opportunities to improve agility and speed in aligning business needs with information technology infrastructure (Hassanzadeh et al., 2011). In this context, SOA promotes service reuse to rapidly build applications by assembling already-implemented and Internet-accessible software pieces, called services.

Both the amount of Web Services available on the Internet and the need to satisfy multiple user requirements and Quality-of-Service (QoS) concerns create a niche for the application of Artificial Intelligence (AI). The main advantage of the use of AI stems from its efficient performance in dynamic, distributed, non-deterministic and uncertain environments; reducing the search space. Further, AI seems to be promising when high scalability and management of generic quality attributes are required, and when verification and validation of service composition (deadlock freedom and safety properties) are demanded.

Those services can be accessed, matched and integrated by discovery and composition applications. Service discovery is the process of locating existing services based on the description of their functional and non-functional properties. In general, a system for discovering services needs a service description language, a service selection means (i.e., matchmakers) and discovery architecture (e.g., decentralized P2P). Service composition combines roles and functionality to aggregate multiple services into a single composite service that can be used in further compositions; composing services involves complex issues that exceed human capabilities to address this process completely manually. The discovery and composition of services have been addressed by several approaches to facilitate the outsourcing of functionality in SOA-based applications (Dustdar and Schreiner, 2005; Rao and Su, 2005). However, the services resulting from discovery and composition might fail to fulfill required QoS concerns or business goals, leading developers to build new services by using development guidelines. Far from being a random process, the development of service alternatives is generally driven by a set of guidelines, incorporating knowledge about recommendable

* Corresponding author.
E-mail addresses: grodri@exa.unicen.edu.ar (G. Rodríguez), asoria@exa.unicen.edu.ar (Á. Soria), mcampo@exa.unicen.edu.ar (M. Campo).

http://dx.doi.org/10.1016/j.engappai.2016.03.009
0952-1976/

design practices, in order to select the most suitable among the numerous alternatives available, even for small search spaces.

In the light of the above, the use of AI arises as a suitable strategy to explore possible solutions for discovering, composing and developing services. Choosing a suitable alternative can be both error-prone and time-consuming (Tekinerdogan and Aksit, 2002); therefore, developers need to be assisted by AI in selecting among effective design alternatives that are also suitable in terms of quality-attribute properties. Considerable advances in AI research and the popularity of AI Planning and Evolutionary Computing in software development have made it possible to widen the research field towards assisting developers in discovering, composing and developing services that satisfy certain QoS concerns, among other constraints.

In this paper, we provide a detailed, conceptualized and synthesized analysis of 69 significant research works that describe AI-based approaches aimed at discovering, composing, or developing services. We also identify open research issues and challenges in the aforementioned research area. Our results can be used by both researchers and practitioners, who can analyze current approaches to identify potential niches for further research or analyze strengths and weaknesses of current approaches and select the most suitable for their professional contexts, respectively.

This article is organized as follows: Section 2 describes our research methodology; Section 3 reports on approaches for discovering services; Section 4 discusses works that have attempted to compose services; Section 5 presents works that have addressed automated or semi-automated development of services; Section 6 discusses future trends and open issues in discovering, composing and developing services. Final remarks and conclusions are stated in Section 7.

2. Research methodology

The goal of this research is to identify major works on the use of AI in discovering, composing and developing Web Services, and thereafter, to classify these works so as to discover gaps, critical issues and opportunities for further study and research. We executed our survey according to Kitchenham's well-established guidelines for systematic literature reviews in software engineering, aiming to achieve objective, unbiased and reproducible results (Kitchenham and Charters, 2007). We present the research questions (RQ) that guided our review process:

- *RQ1:* What evidence is there that AI contributes to achieve better results in comparison with other approaches for discovering, composing or developing services?
- *RQ2*: What are the main characteristics, common features, and differences among the available AI-based approaches for discovering, composing or developing services?
- *RQ3*: Should we expect more accurate results when discovering, composing or developing services by using Planning, or by using Genetic Algorithms?
- *RQ4*: What are the research challenges, needs and future trends in the areas of discovering, composing or developing services?

Our survey procedure starts with a keyword-based search using the following databases:

- Google Scholar
- ACM Digital Library
- IEEE Explore
- Science Direct
- Springer Link

The publication search was firstly conducted in terms of a structured combination of related keywords. Any "Web Service composition", "Web Service discovery", "service-matchmaking", "semantic service composition", "semantic service discovery", "Web Service development", or "Web Service realization" related problems involving the concepts of "automatic", "intelligent", "service-oriented computing", "service-oriented architecture", were covered by highlighting the support of AI techniques to achieve optimal results. After a first quick check, the qualified publications were acquired through cross-referencing. The delimitation of the publications is listed as follows:

- Only publications concerning Web Service discovery, Web Service composition, and Web Service development problems are considered.
- Only publications involving the use of AI-based approaches are considered.
- Only peer-reviewed journal/conference papers written in English are considered.

Our research spans over twelve years (2002 to 2014) and we reviewed a total of 69 AI-based approaches for discovering (29 approaches), composing (34 approaches) and developing services (6 approaches). Fig. 1 depicts the distribution of the research works by year of publication. The number of research works that use the support of AI for discovering, composing and developing services has increased from 2007 to 2012, since the SOA paradigm has gained important popularity in software industry. It is worth noting that the application of AI in service-oriented software development research has been increasing since 2007 and this trend seems to increase even more. The most popular AI techniques surveyed are AI Planning, Evolutionary Algorithms, Markov Chains, and Clustering, among others. Our study of the surveyed works has revealed that more research in this field is required to overcome current limitations of the application of such AI techniques to solve open issues in service discovering, composing or development.

2.1. Classification schema

To evaluate the works reviewed and make a comparative description among them, we have identified common criteria in terms of strengths and weaknesses regarding relevant issues in the field of service discovery, service composition and service development. First of all, we read all the analyzed research works. Then, we searched for common features within the whole set of articles; if more than half of the articles contained a feature, it was selected

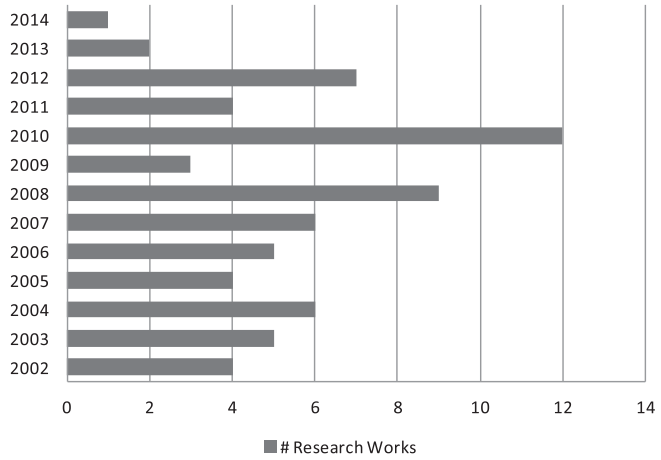

Fig. 1. Distribution of research works by year of publication.

as a common feature. Along this line, we repeated the same procedure for each of the subsets of articles: service discovery, service composition and service development. For all the cases, the features emerged from the articles by considering the domain knowledge in each article and the values that each feature could take. Next, we provide the common criteria:

- *AI Technique*. This feature represents the AI technique or set of AI techniques (e.g. AI Planning) used by the approach, together with parameters and main operators. This criterion attempts to detect the benefits of the technique, as well as its limitations to deal with certain issues.
- *Language*. This criterion refers to the underlying programming language of each research approach, such as Java, or a model for notation such as Meta-object Facility (MOF).
- *Modeling Language*. This feature describes the specification model to describe domain knowledge such as BPEL, and models to specify Semantic Web issues such as OWL-S, among others.
- *Platform and Deployment Environment*. This criterion deals with the set of tools that are part of the development environment, such as Eclipse or Visual Studio. Furthermore, this criterion addresses deployment environment (i.e. Web or desktop applications), architectural style (e.g., CORBA or ESB), and operating systems (Windows or Linux), amongst others.
- *Exhaustive Test*. This criterion evaluates whether the approach was exhaustively validated with a considerable number of case studies.
- *Quality-attribute properties*. This criterion describes the Quality of Service (QoS), meaning both functional and non-functional aspects of a Web Service (Ngan and Kanagasabai, 2013). We mainly included non-functional features maximized by each approach (Lo et al., 2015). A complete list is presented as follows: availability, accessibility, scalability, efficiency, flexibility, integrity, interoperability, maintainability, portability, security, performance, reliability, reusability, robustness, testability, and usability.

2.2. Specific criteria for discovering services

Complementing the aforementioned criteria, we outline the specific criteria to evaluate service discovery works as follows:

- *Mediation Support*. This criterion addresses the strategies and tools to support the process of discovering services. In other words, this criterion focuses on the matchmaking process between queries and service descriptions. The possible values are none, syntactic, semantic, context-aware and ontology-support.
- *Query Specification*. This feature describes the available mechanisms and manners to specify a query for service discovery. The possible values for this criterion are: keywords, natural language, constraints (e.g., performance greater than 0.75), template (e.g., WSDL template) and meta-model (queries based on service properties).
- *Discoverable Service Information*. This criterion refers to the kind of information that can be discovered throughout the registries. The values can be *service* descriptions, or *operation* descriptions.

2.3. Specific criteria for composing services

We outline the specific criteria to evaluate service composition works as follows:

- *Optimization of Composition*. This criterion deals with the environment in which the optimization takes place, such as network, runtime, compilation time and design time.
- *Semantic Web*. This criterion refers to the use of Semantic Web technologies to represent the user requirements for each service in service composition, such as OWL-S.

2.4. Specific criteria for service development assistance

This sub-section outlines the specific criteria to evaluate service development works:

- *Design Patterns*. This criterion describes design patterns used by the approaches (e.g., Mediator, Proxy, Broker or Visitor).
- *Development output*. This feature represents the format in which the development is presented by the approach; for example, a UML (Unified Model Language) class diagram, a piece of code, or a workflow, amongst others.

2.5. Comparison

In order to compare and contrast the approaches surveyed, we have developed an evaluation model that assigns a value to each criterion according to the weight of that attribute. Our model assumes that the values given for the criteria create an incremental scale that allows the comparison of the research works. For example, considering the criterion “Optimization of composition” of composition approaches, “design time” represents a higher value than “runtime”, since an optimization level at design time provides developers with a valuable insight to assess the application design at early stages.

We have selected some criteria for service discovery, composition and development and assigned a numerical value to them. In order to make the different criterion scales clear each value ranges from 0 to 2, to 3 or to 4, depending on the number of possible values that could be taken by each criterion. Finally, the evaluation points for each approach are calculated by summing up all numerical values assigned to each criterion (rightmost column of Tables 1–3). It is worth mentioning that rows of Tables 1–3 are ordered descending by the evaluation points.

For service discovery, composition and development, the possible values to exhaustive test are 0 to indicate that the research work did not perform exhaustive tests and 1 otherwise. The criterion “quality-attribute properties” may take the value 1, only if more than one non-functional requirement has been maximized and 0 otherwise.

As for service discovery, we have selected the following criteria to make comparisons: exhaustive test, quality-attribute properties, mediation support, query specification and discoverable service information. Mediation support could take 0 for “none”, 1 for “syntactic”, 2 for “semantic/ontology support” and 3 for “semantic and context-aware”. Query specification may take 1 for “keyword”, 2 for “natural language”, 3 for “templates” and 4 for “meta-models”. Finally, “discoverable service information” could take 1 if the support is at operation level or at service level and 2 if the support is at service and operation level.

With regard to service composition, the criteria selected to compare are exhaustive test, quality-attribute properties, optimization of composition and Semantic Web. The criterion “optimization of composition” may take 1 for “network level”, 2 for “runtime”, 3 for “compilation time” and 4 for “design time”. The last criterion may take 0 for “none”, 1 for “ontologies” and 2 for enriched ontologies, such as “ALE (Attributive Language and full Existential qualification) ontologies”, “semantic annotations”, “distributed operational semantics” and “knowledge-based operators”.

As for service development, the criteria used for comparison are exhaustive test, quality-attribute properties and development output. The latter may take 0 for “none”, 1 for “diagrams”, 2 for “specification documents” such as WSDL-based documents, and 3 for “code”.

Table 1
Characterization of Web Service discovery systems.

Research work	AI technique	Language	Model language	Platform and deployment environment	Exhaustive test	Quality-attribute properties	Mediation Support	Query Specification	Discoverable Service Information	Sum of Points
Rong and Liu (2010)	**Matching**	**N/A**	**OWL-S, WSDL-S, WSML, DAML-S**	**N/A**	**No**	**Precision, Reliability**	**Syntactic, semantic, context-aware**	**Keywords, meta-model**	**Service, operation**	**10**
Shao et al. (2007)	**Collaborative Filtering**	**Java**	**Custom Model**	**Eclipse, Axis 1.4**	**Yes**	**Availability, Responsiveness**	**Semantic**	**Keyword, meta-model**	**Service, operation**	**10**
Suraci et al. (2007)	**Matching**	**SLP, XML, UPnP**	**OWL-S**	**N/A**	**No**	**Precision, Effectiveness**	**Semantic, context-aware**	**Keywords, meta-model**	**Operation**	**9**
Kuck (2007)	**Information Retrieval**	**N/A**	**WSDL, tModel**	**UDDI API v. 3**	**No**	**Reliability, Accuracy**	**Semantic, context-aware**	**Meta-model**	**Operation**	**9**
Manikrao and Prabhakar (2005)	**Collaborative Filtering**	**N/A**	**DAML-S**	**UDDI API**	**No**	**Precision, Availability**	**Semantic**	**Meta-model**	**Service**	**9**
Xiao et al. (2010)	Ontologies	Java	OWL	Eclipse, OWL API, RDF parser	Yes	Precision, Reliability	Semantic, context-aware	Keywords	Service, operation	8
Chan et al. (2012)	Collaborative Filtering	Flex (front-end), Java (back-end)	VSM (Vector Space Model)	Remote Object and BlazeDS framework (Web Platform)	Yes	Usability, Precision, Reliability	Semantic, context-aware	Natural language	Operation	8
Li et al. (2006)	Ontologies	Java	WSDL-S	Radiant, Lumina, Saros, Eclipse	No	Precision, Reliability	Ontology support	Template, meta-model	Operation	8
Sivashanmugam et al. (2003)	Ontologies	N/A	DAML-S, tModels (technology models)	UDDI API v. 1	No	Scalability and Robustness	Semantic	Template, meta-model	Operation	8
Liang (2006)	Service Mining	N/A	OWL-S, BPEL4WS	Web Services interaction mining architecture (WSIM)	No	Precision, Availability	Semantic	Keywords, meta-model	Service	8
Kawamura et al. (2005)	Ontologies	Java	WSSP (Web Service Semantic Profile)	Eclipse, Voice of the Customer (VOC), UDDI API	Yes	Accuracy	Semantic	Template, meta-model, keyword, constraint	Operation	8
Cardoso and Sheth (2003)	Ontologies	HTML	DAML-S	METEOR-S, Web platform	No	Cost-effectiveness, Reliability	Ontology support	Template, meta-model, natural language	Operation	8
Paolucci et al. (2002)	Ontologies	N/A	DAML-S, tModels	UDDI API v. 1	No	Interoperability	Semantic	Template, meta-model, keyword, constraint	Operation	7
Sangers et al. (2013)	Ontologies	Java	WSMO	WordNet, Eclipse	Yes	Precision, Reliability	Semantic	Keywords	Service, operation	7
Ma (2008)	Clustering Semantic	N/A	OWL-S	N/A	Yes	Precision, Availability	Semantic	Natural language, Keyword	Service	7
Kokash et al. (2007)	Collaborative Filtering	Java	Implicit Culture	Eclipse, Axis	Yes	Precision, Availability	Semantic, matching	Natural language	Operation	7
Mistry et al. (2012)	Ontologies	Java	OWL-S	JUDDI server, JENA, Protégé, Eclipse	Yes	Precision, Learnability	Semantic	Keywords	Service, operation	7
Dong et al. (2004)	Information Retrieval	HTML	WSDL	Web platform	Yes	Trust-worthiness	Syntactic	Template, composition	Service, operation	6
Mecar et al. (2005)	Ontologies	Java	DAML-S	Eclipse, DAML JessKB	No	Precision, Reliability	Semantic	Keywords	Service, operation	6
Balke and Wagner (2003)	Collaborative Filtering	N/A	OWL-S, WSML, DAML-S	N/A	No	Precision, Reliability, Accuracy	Semantic	Keywords	Service, operation	6
Crasso et al. (2008)	Information Retrieval	Java	WSDL	Eclipse	Yes	Flexibility, Precision	Syntactic	Keyword	Service, operation	6
Roman et al. (2005)	Ontologies	MOF	WSMO	SDK-cluster	No	Cost-effectiveness, Scalability and Robustness	Semantic	Natural language	Service	6
Pathak et al. (2005)	Ontologies	Java	OWL-S	JESS engine	No	Interoperability	Semantic	Keywords, constraints	Service, operation	5
Martin et al. (2007)	Ontologies	N/A	OWL-S	UDDI, WSDL, SOAP	No	Interoperability	Semantic	Natural language	Service	5

Table 1 (*continued*)

Research work	AI technique	Language	Model language	Platform and deployment environment	Exhaustive test	Quality-attribute properties	Mediation Support	Query Specification	Discoverable Service Information	Sum of Points
Birukou et al. (2007)	Information Retrieval	Java	XML	Enterprise Java Beans (EJB)	Yes	Robustness	Syntactic	Keyword	Operation	4
Sreenath et al. (2003)	Collaborative Filtering	N/A	WSDL, DAML-S	UDDI API	Yes	Interoperability, Trustworthiness	Semantic	N/A	N/A	4
Zhuge and Liu (2004)	Information Retrieval	Grid Operation Language (GOL)	tModel	Web platform	No	Effectiveness	Syntactic	Keyword	Service	3
Wang and Stroulia (2003)	Information Retrieval	N/A	WSDL	N/A	Yes	Precision	None	Keywords, natural language	Service	3
Stroulia and Wang (2005)	Information Retrieval	N/A	WSDL, DAML-S	WordNet	No	Precision	None	Keywords, natural language	Service	3

In accordance with our research questions, in the following sections we summarize, categorize and evaluate the 69 approaches reviewed.

3. Approaches to discover services

Discovering services that fulfill the functional requirements of the client through common service registries is a burdensome and daunting task (Garofalakis et al., 2006) that requires developers to choose from a plethora of services, a typical scenario in massively distributed environments as the Web (Crasso et al., 2010).

3.1. Web service discovery

A challenging issue for service discovery application systems, which have to locate existing Web Services, is to represent and discover information related to those services. The documents written in WSDL (Web Service Description Language) are the ones used to describe and represent a Web Service.

Some research has been conducted in the context of service discovery and matchmaking. Rambold et al. (2009) have reported on autonomic service discovering approaches by performing a comparison with eight prime criteria. A crucial remark stated by the authors is that autonomic service discovery is an essential prerequisite for an autonomic service composition later on and therefore helps to tackle the high complexity of current service infrastructures. Ngan and Kanagasabai (2013) provided an extensive review of semantic Web Service discovery. This review concluded that a synergy of service discovery and semantic technologies facilitates rich and formal representations of services and agent interactions, as well as specialization and generalization of service needs.

Shvaiko and Euzenat (2005) systematically discussed approaches and systems developed in schema and ontology matching domains. Most of the schema/ontology matching systems exploit only syntactic and external techniques following the input interpretation classification. Syntactic techniques interpret the input in function of its sole structure following some clearly stated algorithm, whereas external techniques exploit auxiliary resources of a domain and common knowledge in order to interpret the input. The input interpretation classification is based on the granularity of match, i.e., element- or structure-level, and then on how the techniques generally interpret the input information.

Additionally, terminological and structural techniques are also exploited by the schema/ontology matching systems. Terminological techniques can be string-based or based on the interpretation of these terms as linguistic objects, whereas structural techniques are split into two types of methods: those which consider the internal structure of entities such as attributes and their types, and those which consider the relation of entities with other entities. These techniques are performed by following the kind of input classification, which is based on the kind of input used by elementary matching techniques.

Finally, only one schema/ontology matching system uses semantic techniques following both input interpretation classification and the kind of input classification. Semantic techniques use some formal semantics to interpret the input and justify their results. Nonetheless, the category of semantic techniques requires further research.

Bellur et al. (2008) provided a taxonomy of semantic matchmaking algorithms by considering functional and non-functional requirements of Web Services. The authors have described Greedy algorithms, bipartite matching, heterogeneous ontologies, algorithms based on description logics, and algorithms based on ranked instance retrieval. Dong et al. discuss dimensions used to

Table 2
Characterization of Web Service composition systems.

Research Work	AI Technique	Language	Model Language	Platform and Deployment Environment	Exhaustive test	Quality-attribute Properties	Optimization of Composition	Semantic Web	Sum of Points
Lècuè et al. (2008)	**Planning**	**Golog**	**DL**	**Eclipse, Prolog**	**Yes**	**Performance, Expressiveness**	**Runtime, design time**	***ALE* ontology**	**8**
Sirin et al. (2005)	**HTN-Planning**	**Semantic Web Rule Language (SWRL), Golog, Java**	**OWL, SHOP2**	**SHOP2, Eclipse**	**No**	**Performance, Trustworthiness**	**Design time**	**OWL-S**	**7**
Fajiang et al. (2014)	**Genetic Algorithm (binary tournament, crossover, mutation)**	**Java**	**WS-BPEL**	**Eclipse (Windows), OWL API**	**Yes**	**Availability, Performance, Cost-effectiveness**	**Design time**	**OWL-S**	**7**
Tang et al. (2011)	**Propositional Logic**	**Java**	**Horn model, Petri nets, SAWSDL, OWL**	**JENA**	**No**	**Performance, Cost-effectiveness**	**Design time**	**Semantic annotations**	**7**
Zhou et al. (2010)	Planning	Java	OWLS-XPLAN	Eclipse	Yes	Scalability, Interoperability, Cost-effectiveness	Runtime	OWL-S	6
Fajiang et al. (2010)	Genetic Algorithm (crossover, mutation)+CBR	N/A	WS-BPEL	N/A	No	Feasibility, Flexibility	Design time	Ontology	6
Tang and Ai (2010)	Genetic Algorithm (crossover, optimizers)+ Constraint Optimization	Visual C#	Custom model	.Net	Yes	Reliability, Precision	Runtime	Knowledge-based crossover operator	6
Santofimia et al. (2008)	Multi-Agent System (MAS)	Java	Belief-Desire-Intention model (BDI)	Jadex platform	No	Usability, Adaptability	Design time	Ontology	6
Li et al. (2010)	Planning	N/A	BPEL	JBOSS	No	Performance	Design time	OWL-S	6
Liu et al. (2006)	Genetic Algorithm (selection, crossover)+Particle Swarm Optimization	N/A	none	N/A	No	Feasibility, Performance	Design time	None	5
Kuzu and Cicekli (2012)	Goal Decomposition and RPG (Relaxed Planning Graph)	Java	PDDL PDDXML	Eclipse, WSIF, WSDL2Java, Apache Kandula Project	No	Responsiveness (High), Sacalability (low)	Runtime	OWL-S	5
Klusch et al. (2005)	FastForward Planning and HTN-Planning	PDDL, C++, Java	OWL	Eclipse, OWLS2PDDL	No	Availability	Design time	OWL-S	5
Yang et al. (2011)	Genetic Algorithm (crossover: *cut & splice*, mutation)	Java	Custom model	Eclipse (Windows)	Yes	Scalability	Design time	None	5
Ponnekanti and Fox (2002)	Propositional Logic	Java, Prolog	Horn model	JESS	No	Performance, Expresiveness	Design time	None	5
Wagner et al. (2011)	Functional Clustering + Planning	N/A	OWL-S	Keikaku algorithm tool, IBM UDDI	Yes	Reliability, Flexibility, Cost-effectiveness	Runtime	Ontology	5
Kun et al. (2009)	MDP + HTN-Planning	N/A	OWL-S, MDP models	N/A	No	Flexibility, Usability, Availability, Reliability, Performance, Cost-effectiveness	Design time	None	5
Bertoli et al. (2010)	Planning	N/A	DAML-S, OWL-S, BPEL	N/A	Yes	Reachability	Compilation and run time	None	4
Zhang et al. (2010)	Swarm Intelligence+Genetic Algorithm	Visual Basic	workflow	.Net (Windows Vista)	Yes	Scalability, Performance	Runtime	None	4
Canfora et al. (2005)	Genetic Algorithm	Java	BPEL	Eclipse (Windows 3 GHz Intel Pentium)	Yes	Reliability, Availability	Runtime	None	4
Wang et al. (2010)	Reinforcement Learning+MDP	N/A	MDP model	Q-Learning algorithm	Yes	Adaptability, Performance	Runtime	None	4
Ai and Tang (2008)	Genetic Algorithm (crossover, mutation)	Visual C#	YAWL	.Net	Yes	Scalability, Extensibility	Runtime	None	4
Hatzi et al. (2012)	Planning (PORSCE II, VLEPO)	PDDL, Java	OWL	Eclipse	Yes	Performance, Availability	Runtime	OWL-S	4
Liu et al. (2010)	Genetic Algorithm (selection, crossover, mutation)+Ant Colony	N/A	BPEL	N/A	Yes	Reliability, Availability, Cost-Effectiveness, Performance	Runtime	None	4
Aggarwal et al.	Constraint Optimization	N/A	BPEL4WS, OWL-S	LINDO Solver,	No	Performance, Precision	Runtime	Ontology	4

Table 2 (*continued*)

Research Work	AI Technique	Language	Model Language	Platform and Deployment Environment	Exhaustive test	Quality-attribute Properties	Optimization of Composition	Semantic Web	Sum of Points
(2010)				METEOR-S					
Hassine et al. (2006)	Constraint Optimization	N/A	OWL-S	N/A	No	Performance, Interoperability, Usability	Runtime	Ontology	4
Doshi et al. (2006)	Markov Decision Process (MDP)+ Bayesian Learning	Java	BPEL4WS, MDP model	BPELWS4J API, IBM Websphere	Yes	Robustness, Adaptability	Runtime	None	4
Narayanan and McIlraith (2002)	Propositional Logic	N/A	DAML-S, DAML+OIL	KarmaSIM	No	Reachability	Runtime	Distributed Operational semantics	4
Tan et al. (2009)	Propositional Logic	N/A	BPEL, Petri nets	N/A	No	Reachability	Design time	None	4
Zou et al. (2012)	Planning	PDDL, Java	WSDL, OWL	Intel Pentium dual core processor 2.4 GHz	Yes	Decentralization Concurrency, and Contingency	Runtime	None	4
Chan and Lyu (2008)	Propositional Logic	PSL	Petri nets, BPEL	N/A	Yes	Reliability	Runtime	None	3
Paik et al. (2012)	HTN-Planning	Java	UML, OWL-S	NASC System	Yes	Scalability	Runtime	None	3
Oh et al. (2008)	Planning	PDDL, Java	WSDL	Eclipse	Yes	Effectiveness, Scalabilty, Robustness	Network level	None	3
Falou et al. (2008)	Planning (Tree Search, Graph Plan)	PDDL	XML	N/A	No	Usability	Runtime	None	2
McDermott (2002)	Estimated-regression planning	PDDL, Prolog	Custom model	UNPOP Planner	No	Cost-effectiveness	N/A	None	0

Table 3
Characterization of approaches to develop services.

Research Work	AI Technique	Language	Model Language	Platform and Deployment Environment	Exhaustive test	Quality-attribute Properties	Design Patterns	DevelopmentOutput	Sum of Points
Elgedawy (2009)	**Concept Substitutability Graph (CSG)**	**N/A**	**G+model, Sequence Mediation Procedure**	**N/A**	**No**	**Business agility, Responsiveness**	**Adapter**	**Code**	**4**
Elgedawy (2013)	**Concepts Substitutability Enhanced Graph (CSEG)**	**N/A**	**G+model, context matching,Sequence Mediation Procedure**	**N/A**	**Yes**	**Interoperability**	**Conversation Patterns, Adapter**	**Code**	**4**
Bhakti et al. (2010)	**CBR**	**Java**	**UML**	**Eclipse, Axis2, SOAP, MySQL**	**No**	**Robutness, Dependability, Availability**	**Broker**	**UML class and sequence diagrams**	**4**
Arnold et al. (2008)	Search-based Graph Isomorphism	Java	Custom models	IBM Rational Software Architect, Websphere	Yes	Performance, Availability and Security	Deployment Patterns	UML class diagram	3
Mannava and Ramesh (2012)	CBR	Java,.Net	WSDL, Feature-Oriented Programming, UML	Eclipse,.Net	No	Modifiability and Transparency	Visitor, Service Injection, Composite	WSDL documents	3
Bhakti and Abdullah (2011)	CBR	Java	UML	Eclipse, Axis2, SOAP, MySQL	No	Robutness, Dependability, Availability	Broker, CBR Pattern	None	1

analyze technical features of semantic Web Service (SWS) matchmakers such as the markup languages used for describing the semantics of Web Services, the SWS discovery mechanisms, the SWS discovery architecture, the approaches used for SWS matching and the extent of matching for service selection, the semantic parts of SWS for service matching and selection, and the platforms or collections for testing the performance of SWS matchmakers (Dong et al., 2013). Klusch (2012) presents an overview and selects results of the international contest on semantic service selection (S3) which has shown a significant improvement in precision from 2007 to 2010. This gain stems from the use of better matching filters, ontology caching strategies and hybrid semantic matchmakers. Platenius et al. (2013) conducted a systematic literature survey of service matching approaches which consider fuzzy matching. The authors considered different aspects of the services' specifications that are matched: ontologies, input/outputs, pre-/post-conditions, protocols, and quality of service (QoS). Also, the authors highlighted that current challenges in the field are related with the combination of different matching approaches and the generation of more expressive matching results.

In the following sub-section, we describe different approaches that use the AI support to discover Web Services, such IR-based approaches, semantic-aware approaches and context-aware approaches.

3.2. IR-based approaches

The use of Information Retrieval (IR) has gained considerable attention in the field of discovering, since IR provides a rich catalog of techniques to process natural language present in service descriptions. Dong et al. (2004) and Korfhage (1997) have suggested removing stop-words and pull out stems from Web Service documents. Regarding the former, the authors developed an approach called *Woogle* that combines multiple vector spaces with clustering techniques and assesses the similarity between two Web Service descriptions, by separately assessing the similarity between each part of these descriptions, and then comparing the individual results. With regard to the latter, the authors proposed to assess the retrieval effectiveness of the approaches by using IR measures such as Recall, Precision and R-Precision.

Wang et al. have proposed to combine a Vector Space Model (VSM) method with a structural-matching heuristics (Wang and Stroulia, 2003). This heuristics consists of two phases: firstly, a textual WSDL description is translated into a vector; secondly, most similar WSDL files to the translated WSDL description are retrieved by comparing XML syntax. In line with the suggestions provided in Stroulia et al.'s survey (Stroulia and Wang, 2005), Wang et al. plan as future work to incorporate *WordNet Lexical Database* for enhancing semantic distance calculus when exploiting WSDL semantic description. Along this line, Zhuge et al. complement syntactic exact matching techniques by connecting terms that semantically include other terms, even different from a syntactical viewpoint by utilizing flexible matching (Zhuge and Liu, 2004). Birukou et al. have combined VSM with past information of Web Services used in a community of developers (Birukou et al., 2007). To gather this information, the authors propose analyzing the query descriptions made by the community, the retrieved list of WSDL candidates for each query, and finally, the successfully invoked Web Services.

Crasso et al. (2008) proposed heuristics for bridging different WSDL message styles by mining relevant terms from data-type definitions. The approach consists in a combination of query-by-example and VSM along with a classification system, and allows developers to state a query using their preferred programming language by specifying the functional interface of the desired service. In this context, the approach looks for services relevant to the example only within a sub-space generated by the classifier. The approach outperforms aforementioned approaches such as the ones described in (Wang and Stroulia, 2003) and (Dong et al., 2004). However, a considerable limitation of the mentioned UDDI-based approaches in this section is the lack of semantic description of Web Services and its analysis so as to enrich the discovery process by increasing service matching.

3.3. Semantic-aware approaches

In order to tackle the limitation of the aforementioned AI techniques, enrichment of the discovery process with ontologies has been addressed; these tools attempt to disambiguate the retrieval documents by supporting semantic matching. This is crucial for software agents that attempt to discover services automatically. Three main efforts have defined a meta-model for describing Web Services: OWL-S (Martin et al., 2007), WSMO (Roman et al., 2005) and WSDL-S (Sivashanmugam et al., 2003). The first provides a framework for describing both the functions and advertisements (actually published in UDDI) for Web Services by using OWL (Ontology Web Language) to allow for adding annotations to operations and quality-attribute properties. The second is a conceptual model for Web Services which comprises ontologies, Web Service goals and mediators. The ontologies allow publishers to add annotations to the interfaces, quality-attribute properties, pre- and post-conditions, as well as effects and assumptions of service operations. The goals model allows discoverers to annotate functional requirements and quality-attribute properties. The mediator model can be used to define mediators responsible for aligning different ontologies. Finally, WSDL-S incorporates semantic descriptions into current Web Service standards; the approach also uses standard extensibility elements to refer from WSDL documents to external ontologies. The semantic information specified in WSDL-S consists of definitions of preconditions, inputs, outputs and effects of Web Service operations.

The aforementioned approaches have gained popularity to the extent that other authors develop discovery systems based on them. It is worth clarifying that the use of semantic descriptions requires to manage shared and distributed ontologies, and to annotate Web Services. Paolucci et al. have described a matchmaking algorithm for Web Service descriptions written in OWL-S, which infers the logical relations between the inputs and outputs of a request with the inputs and outputs of published semantic services (Paolucci et al., 2002). Improving this idea, Kawamura et al. have proposed Matchmaker to incorporate IR-based techniques, semantic annotations and constraint declarations into UDDI (Kawamura et al., 2005). To reduce the search space, the authors decided to use filters, namely namespaces, text, I/O type and constraint. Cardoso et al. have proposed an algorithm to exploit syntactic and semantic information (Cardoso and Sheth, 2003). The approach enriches WSDL documents with semantic descriptions to increase matching precision, acting as a mediator between different ontologies. A further improved implementation of the approach of Sivashanmugam et al. was proposed by (Li et al., 2006), by which users can find highly suitable and partner services efficiently and effectively; the approach also allows partner services to be bound at design-time, deployment-time or execution-time. Moreover, the proposed approach eases the adaptation of the Web Service environments which change dynamically. Along this line, Sangers et al. propose a semantic Web Service discovery framework by making use of natural language processing techniques (Sangers et al., 2013). The framework allows for searching through a set of semantic Web Services in order to find a match with a keyword-based user query. Techniques such as part-of-speech tagging, lemmatization, and word sense disambiguation are used for matching keywords with semantic Web Service descriptions.

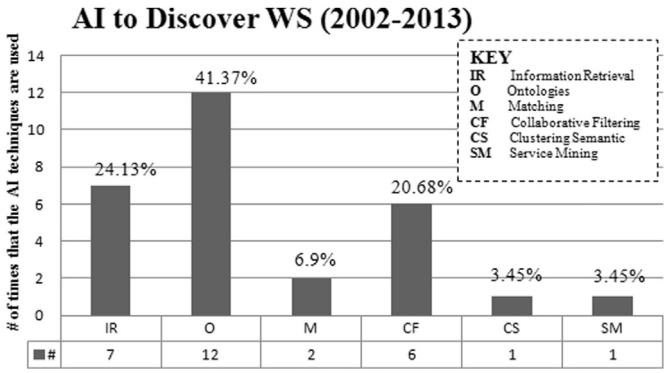

Fig. 2. Distribution of research works of AI in WS discovery.

Pathak et al. describe a framework for ontology-based flexible discovery of semantic Web Services modeling functional and non-functional properties (Pathak et al., 2005). The proposed approach relies on user-supplied, context-specific mappings from user ontology to relevant domain ontologies used to specify Web Services; moreover, the matchmaking engine is aware of the relevant domain ontologies and Web Services. Mecar et al. deal with implementation issues of semantic matchmaking of Web Services using intelligent middle agents (Mecar et al., 2005). The matchmaking algorithm is based on the DAML-S (DARPA Agent Markup Language for Services) ontology, which contains required semantic information for discovery and comparison, as well as execution and monitoring of Web Services in order to find the most appropriate service that meets user preferences. Along this line, Mistry et al. propose a new architecture of SOA that incorporates a new adaptive technique called social learning; the technique improves service provider's domain ontology through service consumer's concept contributions and, thus, eventually makes the service more discoverable (Mistry et al., 2012). The provider's ontology is updated over time through learning from the Internet, by means of social concept contributions of service consumers; this learning (social learning) is conducted by introducing new ontology matching and merging algorithms so as to meet the exact requirement. Nevertheless, the main limitation of the AI techniques applied to the aforementioned works is related to the use of contexts. Disregarding contexts as a knowledge entity may impair the performance of discovery approaches.

3.4. Context-aware approaches

In the light of the above, the use of context-aware computing arises as a promising instrument to explore. According to Suraci et al., context-aware service discovery may be defined as the use of context information to retrieve the most relevant services for the user, which requires an association between the service consumer, service provider and context provider (Suraci et al. 2007). Rong et al. stated that context can be explicitly provided by the user during the matchmaking process or implicitly collected by the system in an automatic or semi-automatic way (Rong and Liu, 2010). Subsequently, the authors have proposed to divide context-aware approaches to discover Web Services into four categories: personal-profile oriented context, usage-history oriented context, process-oriented context, and other contexts.

Personal-profile oriented context has been dealt with in several works aiming to build user profiles with specific user attributes to personalize the Web Service discovery process (Balke and Wagner, 2003; Kuck and Gnasa, 2007). Along this line, Xiao et al. (2010) have utilized the relations among context values to infer users' needs, and then dynamically generated service searching criteria based on those needs to discover and recommend services.

Usage history provides a context by identifying users' consumption pattern to be employed for predicting users' future behavior. This last category has branched into two lines of research: personal usage history and group usage history. According to Kokash et al., the personal usage history profile is built by keeping system log information for each user-system interaction (Kokash et al., 2007), while group usage history profile building presupposes that most Web Service discovery systems offer recommendations to groups of users to simplify the matchmaking process (Chan and Lyu, 2008; Manikrao and Prabhakar, 2005; Shao et al., 2007; Sreenath and Singh, 2004). Collaborative Filtering (CF) has been a widely used AI technique for recommending services in the context of service discovery. In sum, six research works have used CF: (Balke and Wagner, 2003), (Kokash et al., 2007), (Chan and Lyu, 2008), (Manikrao and Prabhakar, 2005), (Shao et al., 2007) and (Sreenath and Singh, 2004).

Process-oriented context is created by the information gathered during the current discovery process; this kind of context aims to study the effectiveness of the candidate Web Services and then to optimize the discovery. For example, (Suraci et al., 2007) has proposed a context-aware semantic service discovery architecture using a context-aware filtering process in order to suit the users' preferences (Liang et al., 2006; Ma et al., 2008).

3.5. Discussion

As we mentioned in Section 2, we employed the six general criteria, together with the three specific criteria to analyze relevant and contemporary discovery systems throughout this section, identifying how AI techniques have facilitated the service discovery task. Fig. 2 depicts the distribution of the most relevant research works, between 2002 and 2013, that have applied AI in the Web Service discovery process. The most popular techniques used by the approaches are Ontologies, Information Retrieval and Collaborative Filtering.

On the one hand, the reported reasons for using ontology are its ability to carry out machine-interpretable descriptions, WSDL-extension support and providers' trustworthiness assurance. On the other hand, the widespread use of IR-based approaches stems from the fact that there is rich background inherited from IR; however, these approaches fail to support non-functional descriptions and depend on provider use of self-explanatory names and comments. Table 1 shows the results in terms of the criteria to characterize the reviewed Web Service discovery systems. The characterization results show that some features are more popular than others depending on the environment. For instance, IR techniques (column 2) are effective in keyword-based searching environments (column 9); however, due to the heterogeneity of the Internet, different keywords are commonly used for advertising services that offer the same functionality. This occurs because different services are built by diverse development teams, who might not share the same programming conventions. The inconsistency between keywords and in interfaces of already-available services and queries may be the cause of many problems related to the retrieval effectiveness of discovery approaches. Moreover, for example, a common situation is that the name of an output of a service operation fails to be meaningful enough. Thus, it is necessary to link this name to a concept so that the discoverer can deduce what a service provides. For these reasons, most of the discovery systems (12 out of 29 approaches, column 1) use ontologies as AI support, model the domain by means of OWL-S (column 4), in combination with query specifications based on keywords and natural language, and follow a semantic matchmaking process. Along this line, context-aware approaches are suitable for situations in which service registries are full of numerous services, and it is required to know user data, such as

location, type of terminal in use, or control policies (for example, row 18 of Table 1). These data are highly necessary for the discoverer to meet user requirements in terms of QoS, performance, cost, accessibility, among others.

From a quality-attribute viewpoint, we can conclude that there are crucial criteria that should be considered when designing service discovery systems. First, the systems should be standards-compliant and should support reuse. Second, depending on the contexts, the semantic formalism should be rich enough to encode service descriptions, requests and goals. Third, by means of AI support, the systems should require minimal user involvement. Fourth, if the context of the systems should deal with a large number of services and a large numbers of users at the same time, the scalability becomes essential, particularly in the matchmaking phase. Fifth, the systems should continue working without performance loss under faults or network changes. Finally, the use of WSMO has been a suitable tool to address issues related to heterogeneous contexts such as different platforms, data formats, and ontologies, among others.

It would be rational to expect that approaches that are mediated by means of ontologies and semantic annotations rely on semantic model languages, such as WSDL-S and OWL-S together with its variants. After analyzing the research works, we cannot to provide evidence of acceptable satisfaction of quality attributes such as scalability, robustness and testability. Information Retrieval and Collaborative Filtering techniques exploit semantic annotations within WSDL documents to provide developers with more precise and accurate Web Services that satisfy challenging functional and non-functional requirements. It is also worth noting that most of the programming platforms are based on Java; from an architectural viewpoint, the aforementioned approaches are centralized, decentralized P2P or hybrid (Klush, 2008).

It is worth mentioning that there are a number of fuzzy techniques for service discovery (Platenius et al., 2013); however, they are out of the scope of this research. There are situations in which no service exactly satisfies the request; thus, approximate matching is necessary to deal with a certain amount of fuzziness.

After summing up, the following research works come out on the top five of the most valuable works: Rough et al. (2010), Shao et al. (2007), Suraci et al. (2007), Kuck and Gnasa (2007) and Manikrao and Prabhakar (2005). Most of these works use context-aware mediation, support specifying queries by using meta-models and discover information at operation level and at service level.

4. Approaches to compose services

The main purpose of Web Services is to achieve interoperability among distributed and heterogeneous applications to combine the functionality of several Web Services (Alonso et al., 2004). Therefore, flexible composition of Web Services to fulfill the given challenging requirements is one of the most important objectives in this research field. According to (Klush, 2008), service discovery and service composition are in the context of service coordination, which aims at the coherent and efficient discovery, composition, negotiation, and execution of semantic Web Services in a given environment and application context. Service discovery is the process of locating existing Web Services based on the description of their functional and non-functional semantics, whereas service composition is the act of taking several compatible services, and bundling them together to meet the needs of a given customer.

Generally, to obtain a required composite Web Service, the user has to specify an abstract workflow consisted of single services that are bound at runtime. In particular, the mainstream approach to composition is to have a single entity responsible for manually scripting such workflows (orchestration and choreography) between WSDL services of different business partners (Klush, 2008). Nonetheless, neither WSDL nor BPEL (Business Process Execution Language) or any other workflow languages such as UML have formal semantics which would allow for an automated logic-based composition. Thus, AI emerges as a suitable support for addressing automatic service composition.

4.1. Web service composition

Along Web Service composition, five phases can be identified: service description, service matchmaking, service classification, service combination, and service selection (Syu et al., 2012). Currently, service description uses WSDL as the most widely-accepted and de-facto standard, yet WSDL lacks formal semantics and metadata of service; thus, the Semantic Web community has provided service specification ontologies, such as Semantic Markup for Web Services (OWL-S), Web Service Modeling Ontology (WSMO) and Semantic Annotations for WSDL (SAWSDL), to accurately and semantically describe services (Loutas et al., 2012). Service matchmaking detects functional similarity or compatibility among service's information, such as functional and non-functional properties. Service classification supports both service combination and service selection. In the case of service combination, service classification allows for clustering and treating services that have identical functionality as a single unit; while in service selection, classification can aggregate services providing needed or required functionality, but posing different non-functional properties, and then the service selector can choose the most appropriate.

A Web Service may have numerous alternative development solutions, all of which have the same functionality, but may have different QoS properties. Thus, a significant research problem in Web Service composition is how to select a Web Service alternative for each of the Web Services so that the composite Web Service delivers the optimal overall performance. There may be incompatibilities between Web Services at the time of development; these incompatibilities may be due to dependency constraints and/or conflict constraints. The former occur when the implementation of a certain Web Service demands the implementation of another particular Web Service; whereas the latter occur when the implementation of a certain Web Service excludes the possibility of including a set of alternatives in the Web Service composition (Tang and Ai, 2010).

In order to enhance the research carried out by (Dustdar and Schreiner, 2005; Rao et al., 2005; Peer, 2005; Küster et al., 2005; Klush, 2008; Strunk, 2010; Bartalos and Bieliková, 2011), we provide readers with approaches that exploit AI to assist developers in service composition.

Dutsdar et al. discuss an urgent need for service composition, along with the required technologies to perform service composition, and present numerous composition strategies based on existing composition platforms and frameworks (Dustdar and Schreiner, 2005). Rao et al. (2005) present an overview of research efforts on automatic Web Service composition, both from the workflow and AI Planning research community. Peer exclusively addresses the use of AI Planning in dynamic and incomplete-information Web Service composition contexts (Peer, 2005). Küster et al. (2005) classify applications of service compositions, especially automatic service compositions. The classification is organized into three categories such as fulfilling conditions, generating multiple effects and overcoming lack of knowledge. Klush (2008) reports on the use of AI Planning in service discovery and composition in the context of semantic Web Services. As for service discovery, he classifies research works related to service description languages, service selection means and discovery architectures. With regard to service composition, he has classified

service composition planners at different levels such as functional level and process level, and at different conditions under uncertainty, such as static and dynamic service composition. Strunk summarizes, classifies and evaluates research efforts on QoS-aware service composition; firstly, the author indicates that the tasks required by the composite service and their interactions, the control and data flow are identified. Then, an appropriate concrete service is selected and bound to the task (Strunk, 2010). Bartalos et al. survey automatic Web Service composition approaches regarding not only issues related to semantics of services, but also formalization of pre/post-conditions of the automatic dynamic composition problem (Bartalos and Bieliková, 2011). In the following sub-sections, we review widely used AI techniques, namely Planning, Genetic Algorithms, Information Retrieval, and Collaborative Filtering, among others.

4.2. AI planning approaches

The aim of AI Planning is to find a series of actions that allow any entity, in this case a service composition problem, to mutate from an initial state towards a final state by achieving a predetermined goal. OWL-S, which facilitates modeling and specifying Web Service composition problems, is particularly suitable for the application of Planning techniques (Stavropoulos et al., 2013).

According to Küster et al., a typical situation when composing services could be the chaining of services. This occurs when some preconditions of a required service are not fulfilled; then, a chaining of services takes place, in which a combination of services can fulfill the required preconditions of the initial service. There exist four approaches to deal with service chaining, such as graph search, forward chaining, backward chaining and estimated regression planning. Graph search proposes to build a graph to represent all services available. Forward chaining starts with the available knowledge about the world to find services whose precondition can be met, inferring additional knowledge until the requested effects are fulfilled. Unlike forward chaining, backward chaining starts with those services generating the requested effects instead of those whose preconditions are grantable. Estimated regression planning arises as a strategy to improve the performance of the latter heuristics (Küster et al., 2005).

Bertoli et al. propose a planning approach via model checking, in which the planning domain is represented by finite states with asynchronous communication primitives (Bertoli et al., 2010). The approach may work in non-deterministic, partially-observed, and asynchronous domains; the planning domain is generated automatically from the WS-BPEL specification of the component services by considering reachability requirements. Then, the generated plans are automatically translated back into executable WS-BPEL processes that implement the composite service so it can be run on standard execution engines. This approach covers relevant service composition scenarios and overcomes limitations of current approaches in effectiveness and expressiveness. The relevant scenarios are described in terms of real-life contexts, such as scenarios where components feature loops and where the composition requirement can be reached after they traverse such loops a finite number of times; symmetric scenarios where services play different roles (P&S, Producer and Shipper scenario), among other scenarios. The current limitations of the approach are stated in terms of expressiveness and effectiveness. For this reason, the research work of Bertoli et al. aims to address the problem of composing stateful Web Services according to complex behavioral requirements. Finally, Bertoli et al. have tested scalability of their approach over the number of asynchronous component services in the context of symmetric scenarios; however, the authors pose that techniques for further improving the scalability have to be investigated.

Another planning approach is introduced by McDermott who represents atomic services as state transition operators and employs estimated regression planning with heuristics to compose Web Services by extending current standards such as PDDL, UNPOP planner and HSP. PDDL was extended with a polymorphic type system, whereas UNPOP and HSP are planners that were extended to handle simple Web Service composition problems. Despite of the encouraging results in precision and cost-effectiveness, there is still room for improvement as regards scalability using this approach (McDermott, 2002). Oh et al. explore an AI Planning framework for automatic and flexible composition of Web Services to fulfill challenging requirements, such as achieving interoperability between distributed and heterogeneous applications, and the publication, location and execution of loosely coupled software components as integral parts of distributed applications (Oh et al., 2008). The polynomial-time heuristic is based on Graph Plan and aims to minimize the number of actions (i.e. Web Services); nonetheless, the approach fails to incorporate semantic information into the Web Service specifications. By using the general approach of AI Planning, El Falou et al. propose to model services as actions, and business processes as planning to connect Web Services (El Falou et al., 2008). To carry out the research, the authors use the Planning Domain Definition Language (PDDL) to describe the planning domain and two AI Planning algorithms: Tree Search and Graph Plan. The advantage of this approach is allowing for the creation and elimination of objects when executing actions. This advantage contributes to meeting new and more expressive requests, in which goals may contain objects that have been generated by the plan. Klusch et al. (2005) use semantic descriptions of Web Services in OWL-S to derive planning domains and problems, and then invoke a planning module, called XPlan, to generate composite services. This system is compliant with an XML dialect of PDDL, yet semantic information provided by domain ontologies is not utilized; therefore, the planning module requires exact matching between service inputs and outputs. To address this issue, Hatzi et al. facilitate the composition by adding semantic information to WSDLs by means of ontologies (Hatzi et al., 2012). The Web Services are described by using OWL-S and translated into planning by PDDL using the tool PORSCE II and executed by VLEPO. The approach achieves high quality, by considering the accuracy metric, and avoids expert's intervention; thus, even non-experienced users can compose services.

In the research conducted by Lècuè et al., semantics and causal laws on the Web Service composition problem are considered. Utilizing Situation Calculus (SC) allows for adding expressiveness to service parameters, adapting *Golog* by means of Description Logics (DL) (Lécué et al., 2008). The approach attempts to address how to effectively retrieve a conditional composition of services by means of its causal links and laws (through AI Planning). To this end, the authors suggest adding explicit DL reasoning between input and output parameters of services (causal links) to the situation calculus, and combining them with reasoning on causal relationships between their preconditions and effects (causal laws). By combining causal links and causal laws, it is possible to reduce search space, reduce the number of alternative plans and increase performance; these advantages stem from the fact that Lècuè et al.'s approach is more restrictive than approaches that consider only causal links or only causal laws. Moreover, considering conditional plans is obviously flexible and adapted to the available plethora of Web Services. Li et al. introduce the separation model to isolate the representation of Web Service type definition from the definition of instances, in order to improve the organization of Web Services (Li et al., 2010). In this context, the AFlow system is introduced by combining planning and service matchmaking, which yields favorable reductions in the planning domain. This reduction of the planning domain scope is achieved

by abstracting service types from lots of service instances, enhancing the performance and efficiency of the planner. The AI Planner receives user requirements and builds abstract workflows, whereas the Service Matchmaker selects the most suitable instances from each abstract workflow.

Sirin et al. present SHOP2, a Hierarchical Task Network (HTN) planning approach for automated composition of OWL-S Web Services by executing Web-provided information during the planning process (Sirin et al., 2004). Since HTN planning is similar to the concept of decomposition process in OWL-S ontologies, the backward-chaining approach translates OWL-S service descriptions into the SHOP2 domain, and executes resulting plans on the Web. However, this approach presents two limitations: one is that mental states of agents involved are not modeled; and the second is that it fails to generate conditional plans. The first limitation is the lack of modeling of agents' mental states. The authors assume that all effects in planning are physical; in complex situations, there may be other changes, such as in the mental states of the agents involved in the planning process, but they are out of the scope of the research. The second limitation is the lack of conditional plans; these plans are important to mitigate the constraint on information change during planning. Zou et al. propose a framework of service composition in multi-cloud based environments. The importance of multi-cloud based environments stems from the fact that cloud computing is evolving as a widely used computing platform where many different Web Services are published and available in cloud data centers. The approach combines planning and combinatorial optimization by exploiting OWL-S XPlan to generate composition plans, achieving a superior trade-off between response time and quality. This quality is measured in term of the collection of the most suitable clouds for service conversion and planning (Zou et al., 2010). The approach is a practical solution for deployment in multi-cloud Web Service provision environments and can effectively and efficiently find a desirable cloud combination for Web Service composition. In line with highly dynamic environments, Kuzu et al. present *SimPlanner*, a dynamic planning approach to solve current issues in automated service composition. These issues are partial observability of the environment, non-deterministic effects of Web Services, service execution failures and compensation mechanisms (Kuzu and Cicekli, 2012). The approach proposes to address these issues by providing a real application of a dynamic recovery system, not only for service failure, but also for the situations where the user cannot provide input for services. Also, the approach addresses the issue in which the semantic Web Service descriptions and the relations between them are discovered automatically. *SimPlanner* is an anytime planner and domain independent for critical time operation that addresses the limitation of HTN-based approaches regarding insufficient knowledge due to domain dependence. In this context, availability must be checked repeatedly; the planner keeps track of the current state of the workflows and, in case of errors in the composition, the Web Service composer informs the planner about the situation and the planner provides alternatives. The connection between semantic and syntactic Web Service descriptions is carried out by OWL-S. During the Web Service composition, user interaction is needed to validate the plans found; in case users fail to provide the tool with the required input, *SimPlanner* tries to find a Web Service that supplies that information needed. Paik et al. (2012) deal with small, restricted parts of fully automated composition, to enable nested multilevel composition for achieving scalability, by means of workflow orchestration. However, the Web Service environment is highly complex and sometimes it is not feasible to generate compositions automatically. Although planning has been used for orchestration, Zou et al. propose to generate a distributed choreography plan based on automated planning, since planning is more suitable for constructing the composition plan from the perspective of a single party in orchestration. The approach starts from a repository of related WSDL-based documents and user-defined contingencies, and represents them in PDDL; then, a master plan that provides a global view of the choreography is built, and it is further decentralized and localized in each peer using a dependency graph (Zou et al., 2012). The main encouraging results are: (1) a method to generate distributed plans for Web Service choreography is presented, and (2) efficiency and fully automation by means of AI Planning and dependency graph analysis are achieved, and (3) assurance of correct collaboration of multiple peers is demonstrated. Nonetheless, the quality optimization of the plan introducing non-functional properties is mentioned as future work.

We think that the widespread use of AI Planning may stem from the fact that it is a suitable technique to deal with dynamic composition in contexts with incomplete information; however, the technique may be enhanced with semantic information for approximating the optimal composite services when exact solutions are not found. Moreover, AI Planning fails to be suitable for a choreography-based Web Service composition, since it involves decentralized control, concurrent workflows, and contingency.

4.3. Evolutionary approaches

In order to address the aforementioned issues of AI Planning, evolutionary approaches arise as feasible tools to explore. From a computational point of view, the Web Service selection problem is a typical constrained combinatorial optimization problem. Thus, Genetic Algorithms (GAs) have resulted effective and efficient tools for solving the problem. Canfora et al. propose a QoS-aware evolutionary approach to bind a set of concrete services to abstract services within an orchestration. To address constraints, the approach uses either a static or a dynamic fitness function based on penalty; however, the performance of the Genetic Algorithm is sensitive to the increase of search space. The authors conclude that, depending on the size of the concrete service sets, the GA outperforms the Integer Programming algorithm (Canfora et al., 2005). Aligned with service dependencies and conflicts, Ai et al. also use a penalty-based GA to tackle that the overall QoS of the composite Web Service is optimal, along with the constraints on inter-service dependencies and conflicts. Programmed in Visual C# 2005, the algorithm uses crossover and mutation operators achieving high-quality solutions (Ai and Tang, 2008). The main advantages of the approach are scalability (minor changes have to be made) and extensibility (more non-functional properties can be added). Fanjiang et al. use GA and Case-Based Reasoning (CBR) to deal with dynamic service composition. The approach supports flexible service workflow according to users' requirements by composing services (Fanjiang et al., 2010). The GA is used to support the service composition, while CBR is used to reuse workflow structure and services. Besides, the GA uses WS-BPEL descriptions and operators such as crossover and mutation; the fitness function takes into account satisfaction of user requirements, workflow feasibility and Quality of Services (QoS) values. In a later work, Fajiang et al. propose a GA to address semantic-based composition, in which functional and non-functional requirements are considered; this approach disregards human intervention and is special for cases in which multiple composition concerns must be addressed simultaneously, namely overall functionality, internally workable dataflow, non-functional transaction, and QoS (response time, cost and availability, among others) (Fanjiang and Syu, 2014). The operators used in the algorithm are binary tournament (selection), crossover and mutation; the goal is optimized by composing four independent fitness functions ordered by priority for scoring chromosomes. However, the approach is yet to be tested (i) with large-scale real-world service repository and (ii)

exhaustively and extensively considering all possible scenarios.

Liu et al. propose a hybrid approach to tackle the lack of feasibility in Web Service composition based on GA and Particle Swarm Optimization (PSO); for this reason the work belongs to the category Swarm Intelligence. The GA is responsible for searching throughout the problem space, while PSO enhances local searchability (Liu et al., 2007). The approach uses feedback information to balance both algorithms and is independent of knowledge representation. In line with hybrid approaches, Liu et al. address QoS-aware Web service composition by means of a GA improved with ant colony algorithms, suiting real-time requirements. The approach proved to be more efficient than classical ant colony approaches and demonstrated a better convergence (Liu et al., 2010); for this reason the work also belongs to the category Swarm Intelligence. The operators used are selection, crossover and mutation, along with a static fitness function. Despite the benefits of the approach, scalability has not been tested. Tang et al. propose a GA plus a local optimizer, in which each individual represents a Web Service selection plan (Tang and Ai, 2010). Because of this optimizer, this work belongs to the category Constraint Optimization (CO). The optimizer is used twice: at the beginning to improve fitness values, increasing QoS and reducing constraint violations, and at the end of each generation to improve the individuals in the population. However, this optimizer evidences high execution time and instability. Unlike Ai et al.'s, this approach uses a knowledge-based crossover operator. Jiang et al. propose a variable length chromosome GA for QoS-aware service composition among multiple paths. The operators of the GA are mutation and crossover; the latter is carried out by the cut-and-splice technique based on service parameter matching (Jiang et al., 2011). The approach presents considerable scalability for abstract service construction in a changing context, and supports important workflow structures and re-composition; additionally, new paths can be added at will. Nevertheless, the approach fails to support gene exchange inside *And/Or* workflow patterns, and the parameter matching is unsuitable for universal service composition problems. Another considerable drawback is the decrease in performance caused by the fitness function conducted at the beginning of the GA.

To sum up, the aforementioned works reported that GAs represent a more scalable option and are more suitable to handle generic QoS attributes than AI Planning and Integer Programming approaches. Furthermore, evolutionary approaches provide faster composition when re-planning takes place, since current QoS deviates from the estimated one, leading to constraint violations; nonetheless, there are issues that jeopardize the application of GA to compose Web Services. Firstly, it is not possible to define high-priority goals within the fitness function. Secondly, the size of the input directly impacts the performance of the algorithm. Finally, GA may become instable when the constraints' density is considerably high.

4.4. Other AI approaches

Aiming to address the aforementioned issues, other AI techniques have been applied to compose Web Services. For instance, propositional logic was used in the context of SWORD, a developer toolkit for building composite Web Services using rule-based plan generation (Ponnekanti and Fox, 2002). In SWORD, by following a forward-chaining heuristic, a service is modeled by its pre- and post-conditions, which are specified in a world model that consists of entities and relationships among entities. A Web Service is represented in the form of a Horn clause denoting that the post-conditions are achieved if the pre-conditions are true. An important limitation is that SWORD can sometimes generate uncertain results if a precondition cannot determine a unique post-condition. The use of Petri nets has been proposed for service-composition verification and validation (Narayanan and McIlraith, 2002). The authors propose to specify Web Services by using DAML-S, allowing for semantic markup of services. Then, the specifications are translated into Petri nets to leverage their ability for both offline analysis tasks (i.e., Web Service composition) and online execution tasks (i.e., deadlock determination, resource satisfaction and quantitative performance analysis). In line with service-composition verification, Chan et al. introduce an automated composition algorithm with Petri net verification. The algorithm composes Web Services as a choreography based on WSDL specifications; then, this composition is verified to be deadlock free by modeling the Web Service interaction as a Petri net (Chan and Lyu, 2008).

In combination with logical inference of Horn clauses, Petri nets have been chosen to model the rule set, and its structural analysis techniques are used to obtain the composite service. In this context, every basic service is represented by a situation calculus formula whose operational semantics is provided using Petri nets. The existence of the composite service can be determined by checking the reachability of Petri nets. Petri nets concentrate on the conceptual level, making reasonable simplification on some nontrivial details, for example, the partial compatibility issue. This contribution to the automation in service composition led us to consider Petri nets as an AI technique. Compared with other existing Web Service automatic composition approaches, Petri nets do not only consider the input/output type-compatibility of services, but also take the behavioral-constraint compatibility into account.

Petri nets are also used in Tan et al.'s approach (Tan et al., 2009), in which a method is proposed to analyze compatibility between any two services specified in BPEL, by converting them into Colored Petri nets (CPNs). The approach uses a set of formalisms derived from CPNs, such as the Service Workflow Net (SWF-net), defined to describe services, compositions and mediators. Then, the feasibility of mediation is checked by a state-space method by means of the Communicating Reachability Graph (CRG); nonetheless, the approach leaves room for further research regarding cost of the reachability method and full automation of the mediator generation, which depends on services' specific properties. Besides using Petri nets, Tang et al. (2011) propose to use Horn clauses to perform logical inference; the Web Service composition problem is translated into Horn clauses by exploring dependency relations among services. Petri nets model both the rule set, which is gathered based on hyper-graph theory for composition, and its structural analysis for obtaining the service composed. Moreover, the approach considers behavioral-constraint compatibility among services, by using Semantic Annotations for WSDL (SAWSDL) to specify Web Services. Despite the advantages of using Petri nets, there are a number of issues to address, such as the inability to map workflow patterns. These patterns usually involve multiple instances, complex synchronizations or non-local withdrawals onto high-level Petri nets (van der Aaslt, 2005).

In this context, Wagner et al. (2011) propose a Functional Clustering-based approach to enhance reliability, QoS awareness, compliance with QoS constraints and flexibility in Web Service composition. The approach identifies clusters of services with similar functionality by means of functional alignment and QoS aggregation, in which attributes of each cluster, such as price or reliability, are determined. After clustering, the planning phase takes place by the *Keikaku* algorithm, a regression planning tool that performs iterative depth-first search; for this reason, this approach also belongs to the category Planning in Table 2 The advantage of this algorithm is that reliability and utility of computed workflows are significantly increased; however, the

approach has not been tested with different selection algorithms to show how they impact on the results. Aggarwal et al. (2004) propose METEOR-S, an approach that allows designers to bind Web Services to an abstract process by service templates, based on business goals and process constraints. To model the process flow, a BPWS4J API is utilized to parse the abstract BPEL. By means of ontologies, this approach uses Semantic Web concepts, such as data semantics, functional semantics and QoS semantics, to represent the users' requirements. All criteria that affect the solution of the services are represented as constraints or objectives, which are treated with LINDO (Integer Linear Programming Solver); for this reason, this approach belongs to the category CO in Table 2. The promising points of this work are high flexibility and improvements in global optimization. Belonging to the same category, Hassine et al. (2006) propose formalization of Web Service composition as a constraint-optimization problem compatible with any WSDL to tackle Web Service composition. The approach deals with dynamic, distributed and uncertain contexts with incomplete information; moreover, the approach increases user intervention to find optimal solutions at runtime. However, the authors have started to work on extending the approach to a Multi-Agent System (MAS) in order to be more effective in realistic environments. In the work of Santofimia et al., a MAS is proposed to automatically compose services, as a constituent of middleware architecture, providing transparency from the users' viewpoint (Santofimia et al., 2008). This is an important issue since users generally find themselves involved in the composition, by selecting or deciding what services to compose and how.

Another considerable issue in Web Service composition is to deal with non-deterministic behavior of services, which is an inherent stochastic nature of Web Services in dynamic environments. To tackle this issue, Markov Decision Process (MDP) has been used by (Doshi et al., 2004). MDP optimally guides a stateful workflow towards its goal, by using Bayesian learning to learn probabilities within the model, generating robust and adaptive workflows. The approach uses the BPWS4J API and runs in IBM's Websphere application suite; however, scalability remains to be tested. Kun et al. (2009) propose to combine HTN Planning with MDP, since HTN Planning fails to consider the choice of decompositions available to a problem, which can lead to a variety of valid solutions. Thus, the approach uses the decomposition model proposed by SHOP2, yielding more flexibility, among other non-functional properties of Web services. However, exploring a re-planning mechanism is still a challenging issue to be addressed when plan execution fails.

Zhang et al. (2010) use Ant Colony Optimization (ACO) and QoS-based dynamic service composition. The Swarm Intelligence (SI) approach uses a multi-objective optimal-path selection modeling for Web Service composition. The ACO algorithm uses a technique named "the evaporation of pheromones" to find Pareto optimality; this algorithm yields favorable results in efficiency, performance and scalability; despite its lack of QoS-aware dynamic Web Service composition. Wang et al. (2010) propose a Reinforcement Learning (RL) approach in combination with MDP to achieve optimal Web Service composition at runtime, by means of the Q-Learning algorithm. MDP is used to perform composition, whereas RL is used to perform optimization. This approach requires no prior knowledge about QoS and allows the composition process to be adaptive to environment; however, a main limitation is that the approach is yet to be exhaustively tested for scalability.

4.5. Discussion

We have reviewed numerous approaches and frameworks that have been developed in order to provide widely usable Web Service composition platforms. The analysis of relevant and

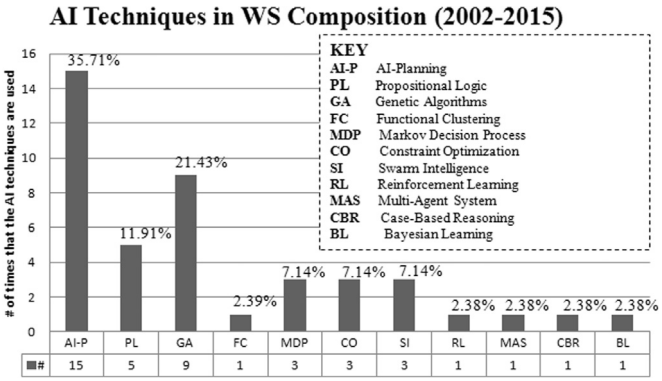

Fig. 3. Distribution of research works of AI in WS Composition.

contemporary Web service composition systems was carried out applying 6 general and 2 specific criteria; this exploration evidenced how AI techniques have contributed to facilitate service composition. Fig. 3 depicts the distribution of the most relevant research works between 2002 and 2015 that have applied AI in the Web Service composition. As shown in Table 2 and Fig. 3, planning techniques (36.11% of the research works) and evolutionary algorithms (22.22% of the research works) have been widely used to build executable workflows of composed Web Services that satisfy users' requirements, especially non-functional requirements such as availability, robustness, performance, and adaptability, among others (column 6 of Table 2).

Most of the approaches described in Table 2 are modeled by means of BPEL (column 7), since they are in charge of building workflows, and use Java-based tools and inference engines (column 4). Furthermore, semantic tools are utilized through ontologies modeled using OWL-S. Thus, as expected, most approaches address runtime composition optimizations. A limitation widely detected in the aforementioned approaches is the lack of independence of programming platforms and vendors. Moreover, few research works are focused on robustness and observability of devices and services (column 6). Consequently, we consider that it is necessary to conduct further research to develop reliable and robust Web Service compositions.

It is worth highlighting that some AI techniques are more appropriate than others depending on the composition environment. For instance, AI Planning is suitable for dynamic Web Service composition with incomplete information. Along this line, constraint-optimization techniques are useful for dynamic, distributed and uncertain environments, especially when user intervention at runtime is required to find optimal solutions. In case of high-scalability requirements and management of generic quality attributes, evolutionary techniques seem to be promising strategies. Instead, Markov techniques are suitable for environments in which designers have to address non-deterministic behavior of Web Services along service composition. In environments where both input/output compatibility and exceptional paths (e. g. faults) are considered, the use of Petri nets is highly recommended. Finally, functional clustering techniques are suggested for environments in which reliability, QoS-awareness and flexibility are highly required. We can provide a complete list of quality attributes in service composition addressed by reviewed research works: efficiency, effectiveness, expressiveness, scalability, accuracy, response time, availability, cost, extensibility, suitability, reliability, utility, flexibility, robustness, adaptability, performance, convergence, execution time, (in) stability.

It is also noticeable that the works of Lecue et al. (2008), Sirin et al. (2007), Fajiang et al. (2014) and Tang et al. (2011) come out on the top four of the most valuable research works related to AI-based service composition. These works propose optimization of

service composition at design time, which is valuable for developers to effieciently assess alternative solutions. Further, these works propose ontologies and OWL-S to support Semantic Web and satisfy several quality-attribute properties simultaneously.

5. Approaches to service development assistance

It is well known that SOA allows developers to rapidly build applications by assembling already-implemented and Internet-accessible services, which allows software organizations to hasten development of distributed applications and their consequent time-to-market (Papazoglou and Heuvel, 2006). However, this approach to develop SOA applications is unsuitable for particular organizations with high-priority and critical demands of internal control, security, flexibility, confidentiality and data integrity of their services, since the development of core functionality may be jeopardized by either uncertainty or changing environment. For this reason, we consider that assisting software developers in building services is a crucial issue when discovering or composing services fail to achieve critical quality-attribute properties. Given that automated software generation is a demanding task, an exhaustive review of the latest literature has evidenced that few approaches aim at applying AI techniques to provide automated development assistance during the software design phase. Next, we shed some light on the use of design patterns and Case-Based Reasoning (CBR) to explore service design alternatives.

5.1. Pattern-driven approaches

At this stage, we have reviewed relevant catalogs of SOA design patterns in the literature, such as Erl (2008), Bell (2010) and Daigneau et al. (2011). Having these catalogs in mind, we have reviewed the research work by detecting and classifying the design patterns. Arnold et al. (2008) propose the use of deployment patterns that represent the structure and constraints of composite solutions, including non-functional properties. The realization is performed by Search-Based Graph Isomorphism (SBGI) and patterns expressed in a formal object-relationship based modeling language, under the IBM Rational Software Architect platform. As for limitations, this work could be furthered to address interactive pattern realization and reverse pattern discovery, among others.

Elgedawy (2009) advise the application of adapters to match requests in the context of service conversations, i.e., interaction among Web Services, yielding improvements in business agility and responsiveness (Elgedawy, 2009). The approach uses Concept Substitutability Graph (CSG) to capture conversation semantics represented by ontologies. CSG consists of segments, where each segment gathers the substitution semantics between application domain concepts with respect to a given application domain operation. Also, the G+model is used to capture conversation patterns, conversation context matching, and Sequence Mediation Procedure (SMP) in message exchanges; this procedure is used to match different state sequences generated from the conversation patterns to be matched. Mannava et al. combine Visitor pattern, on the server side, and CBR pattern (Ramirez and Cheng, 2010), on the client side, for service invocation and Web Service composition (Mannava and Ramesh, 2012). In a context of feature-oriented programming and aspect-oriented programming, the authors propose the use of the Service Injection pattern to introduce new services at runtime by maintaining transparency to users. The approach is able to handle service requests from clients and injects new services into a server's code as feature module. Although the approach is suitable for distributed environments, it has not been tested yet. An improved work presented in Elgedawy et al. introduces Concept Substitutability Enhanced Graph (CSEG), which

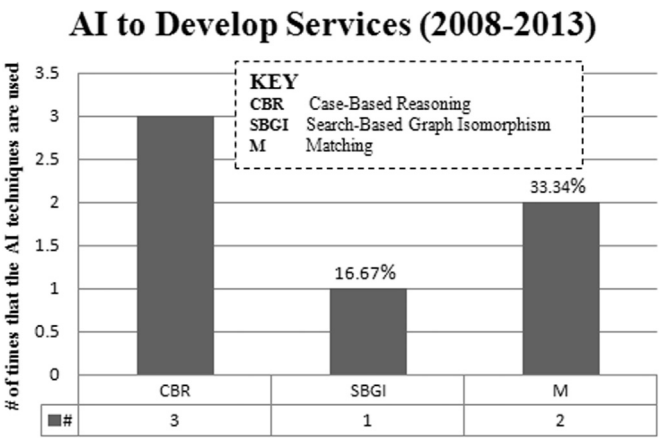

Fig. 4. Distribution of AI-based research works to develop services.

is able to capture the aggregated concept substitution semantics of application domain concepts in a context-aware manner (Elgedawy, 2013). Unlike CSG, CSEG captures both bilateral as well as aggregated conditional substitution semantics of application domain concepts. The approach generates the conversation patterns from the services G+model, and then, matches these patterns using context matching and SMP to find the operation mappings, which determine the structure of the adapter required. After that, converters between different operations, using the concepts substitution semantics captured in the CSEG, are generated. For the sake of simplicity, CSG and CSEG belong to category "Matching" as shown in Fig. 4.

5.2. Case-based reasoning approaches

Widespread use of CBR may be attributed to the resemblance between this technique and human problem solving strategies construed on previous experiences. Furthermore, on the one hand, using CBR reduces the amount of input needed, since CBR searches the current case knowledge base for solutions rather than working out solutions from a rule base which can contain irrelevant rules. On the other hand, a CBR-based approach expands its knowledge base by adding new cases to its repertoire, which makes this technique both time and cost effective. However, CBR presents certain limitations, namely an inefficient method to index and access cases, poor and slow retrieval techniques, avoidable redundancy of similar cases in the knowledge base, and insufficient attempts to adapt solutions to the needs of a current problem (Mansouri and Hamdi-Cherif, 2011). Bhakti et al. (2010) propose SOA autonomic computing by means of CBR along the adaptation, learning and planning phases of self-organizing computing. The authors use a similarity metric called Heterogeneous Euclidean overlap metric, whereas for modeling the SOA meta-model the authors apply the Unified Model Language (UML). As a result, the research attempts to shed some light on improving the usability, adaptability, and robustness of traditional SOA; however, the approach remains to be tested for scalability. Along this line, given that a SOA is capable of changing its structure and functionality autonomously with little human intervention, Bhakti et al. propose the use of CBR to address unpredictable events which could cause unavailability of services in cases of crashes or other network issues (Bhakti and Abdullah, 2011).

5.3. Discussion

As mentioned in Section 2, we applied 6 general and 2 specific criteria to analyze relevant and contemporary approaches to develop services throughout this section, identifying how AI

techniques have contributed to facilitate the development process. Fig. 4 depicts the distribution of the most relevant research works that have applied AI to assist developers in the service development process. Table 3 describes the results after characterizing the approaches for developing services presented in Section 6. One the one hand, we observe that some approaches use CBR (50% as shown in Fig. 4) in order to fully achieve the essence of autonomic computing, such as robustness, dependability and availability, among others (column 7 of Table 3). In autonomic computing, a system needs to exhibit four aspects of self-management: self-configuration, self-optimization, self-healing, and self-protection, some of which are met by means of CBR. We conclude that CBR is useful when it is necessary to find previous solutions to current problems with similar conditions; however, it is advisable to apply CBR in delimited and well-defined application domains, such as SOA, J2EE, among others, in order to leverage a contextualized case base with more retrievable cases.

On the other hand, we noticed that the use of design patterns (column 8 of Table 3) to materialize SOA applications is suitable for situations in which functional and non-functional requirements remain both and equal along the materialization process. Nonetheless, little attention has been paid to assisting developers in developing quality-attribute driven services. Furthermore, none of the approaches has been exhaustively tested with a significant number of case studies. Although some approaches deal with automated realization of code (rows 1 and 2 of Table 3), they are focused only on specific quality attributes and design patterns, and fail to provide customized development alternative solutions. Table 3 also shows the research works that have obtained the highest scores in the comparison procedure: Elgedaway et al. (Elgedaway et al., 2009, 2013). These high scores can be mainly explained by the development outputs provided by the approaches' system applications, which are in the form of UML class diagrams or code.

6. Future trends and open issues

In the previous sections, we summarized the most relevant approaches for discovering, composing and developing services from an AI viewpoint. We identified the main characteristics and common features between these approaches. So far, we have addressed the first three research questions stated in Section 2. The present section addresses the fourth and final research questions by identifying open issues and research challenges in the areas of service discovery, service composition and service development.

6.1. Service discovery

A future trend in service discovery research is the study of protocols for mobile environments, opening new challenges concerning mobility, decentralized P2P architectures and heterogeneity. Mobility is an issue that needs to be further researched, since discovery approaches have to cope with limited resources and computational capabilities of mobile devices, Scalability of discovery platforms is an issue in this environment, both because of the vast number of mobile devices available as service providers/consumers, and for the number of services to be indexed by a registry. A service discovery protocol should be available to service providers/consumers at any time, being sensitive to current environmental conditions; moreover, the protocol should be conversant with the availability and quality of the services that are indexed, and adapt their content. Consequently, as the number of devices increases, monitoring mechanisms require further exploration to provide widespread security and trust.

Despite recent advances in decentralized P2P architecture, the scalability of semantic service discovery in real-time mobile ad-hoc network applications is a considerable open issue. From an architectural viewpoint, reviewed AI support should be broadly applied to decentralized architectures in order to address mobility issues and services may get fulfilled. Up to now, provided solutions have focused only on the expressivity of semantic service description, and the complexity of semantic matching means. As a consequence, to successfully identify service providers over the Internet, it is necessary to monitor the Web, for example, by Web crawling techniques so as to access services with the appropriate functional and non-functional properties.

As for heterogeneity, an environment that consists of heterogeneous devices supposes an agreement on the protocol via which services are advertised and discovered. For instance, Mokhtar et al. have proposed an automated approach to mapping facilities from the discovery source to a discovery target; however, extensibility mechanisms and adaptability strategies remain challenging (Mokhtar et al., 2010). Moreover, in a heterogeneous environment the probabilities of introducing malicious code are high, as well as the number of devices involved in an attack.

6.2. Service composition

Current service composition research also proposes a niche to enhance and enrich approaches concerning mobility, decentralized P2P architectures and heterogeneity.

As for mobility, self-adapting service composition must also deal with mobile devices with limited resources and computational capabilities; thus, it is necessary to explore strategies adaptable to topology changes within the environment to coordinate service composition by considering mobility patterns, platform battery lifetime, fault tolerance and reliability. Moreover, it is extremely daunting for current service composition approaches to be aware of the numerous devices and their failures; as a consequence, observability of what a service provides is still an issue to address by exploiting semantic information. Inference of machine-interpretable information about what a service can do and what it can provide also requires further research. Syntactic interpretation of service-based information lacks the reliability to perform this function properly because the meaning of underlying information is missing (Blake and Wei, 2010).

The advent of the decentralized paradigm yields as a result challenges to service composition. It is still complicated and time consuming to develop, test, and debug compositions of low-coupled services in service-oriented systems with current strategies and tools. Interoperability mechanisms are highly necessary due to the heterogeneity of devices in a decentralized environment. That is to say, service composition should be independent of programming languages, vendors, and operating systems, amongst others; in this context, the exploration of ontologies seems to be a promising line of research to achieve portability and interoperability.

With regard to heterogeneity, security, privacy and trust are still open issues for enabling service access in heterogeneous environments because services taking part in the composition can scarcely be aware of the sources of information they actually interact with (Issarny et al., 2011). The introduction of diverse agent technologies would help to constitute more complex systems in open and dynamic environments. Thus, Web Services would become more autonomous, and the exploration of mechanisms for composing Web Services enacted by autonomous agents will be imperative (Lee et al., 2012), providing significant evidence of the required support of AI techniques to deal with this service composition issues.

6.3. Service development

Although considerable efforts have focused mainly on facilitating service discovery, and the outsourcing and reuse of services in SOA-based applications, little attention has been paid to aiding developers in service development associated with business goals and quality-attribute properties. As stated in Section 5, some lines of research have attempted to shed light on developing services within the organizations by exploring different AI techniques to assist developers in building software pieces; however, quality-attribute properties of the service assemblies have been disregarded, which often leads to mismatches between the quality-attribute behavior prescribed by the architecture specification and the behavior resulting after its development (Diaz-Pace et al., 2012).

In this context, a future line of research should focus on developing services driven by quality-attribute properties within a software organization. Thus, the development of a service normally starts with some form of architectural description (e.g., public interface, main features to be provided, operating environment) and a set of quality-attribute properties, which are taken by the designer to produce a more concrete design model of the service. For example, developing a connection between services can be achieved by applying design patterns depending on the quality attributes defined by the SOA application. As this decision on the suitable candidate design is error-prone and time-consuming, developers need to be assisted in choosing from two or more alternative solutions. It is worth mentioning that these alternatives may be just as effective from a functional standpoint, but might still diverge from the intended architecture in terms of quality-attribute properties. In this light, we have conducted some research on the use of Case-Based Reasoning to assist software developers in choosing suitable object-oriented designs from a quality-attributes perspective to reify service-oriented architectures (Rodríguez et al., 2014).

7. Conclusions

Many different approaches have been proposed to create widely accepted and usable systems for discovering, composing and developing Web Services. In this paper, we have provided a detailed, conceptualized and synthesized analysis of 69 significant research works that presented AI-based approaches aimed at discovering, composing, or developing services in a loosely coupled way. In this context, the use of AI has shed light on both exploiting the semantic resources and achieving quality-attribute properties so as to produce flexible and adaptive-to-change service discovery, service composition and service development systems. Additionally, QoS-aware and semantic descriptions have been suggested to extend the current standards in order to enhance the SOA development process.

Furthermore, three aspects have been reported for being an important influence on future research directions: mobility, decentralized P2P architectures, and heterogeneity. As for mobility, we have observed that the trend in service discovery is the use of ontologies, which support increased reliability, cost-effectiveness, precision and accuracy when retrieving services; however, scalability of semantic service discovery in real-time mobile applications requires further research. Current service composition research should explore strategies adaptable to topology changes within the environment to coordinate service composition by considering mobility patterns, platform battery lifetime, fault tolerance and reliability.

Regarding decentralized P2P architectures and heterogeneity, despite the widespread use of ontologies, scalability of semantic service discovery in different networks, even in decentralized P2P networks, is an issue that needs to be addressed. Planning and evolutionary algorithms have been the most widely used AI techniques to automate the Web Service composition process in dynamic environments; however, issues related to Web Service compatibility and portability demand research to efficiently compose Web Services in heterogeneous and decentralized environments.

To cope with service development, various approaches have provided software developers with guidelines and design patterns to streamline the exploration of alternative solutions. Nevertheless, providing software organizations with semi-automated approaches, both to develop service-oriented applications driven by quality-attribute properties and to assess the fulfillment of functional and non-functional requirements, is a line of research that demands further exploration.

Finally, the current evolution of the Internet presents new possibilities and opens a new research area facing existing challenges in terms of other important issues such as security, trust, awareness and adaptability. That is to say, researchers need to incorporate these issues to the agenda of current limitations surveyed in this research work related to service discovery, composition and development.

Acknowledgments

We acknowledge the financial support provided by ANPCyT through grant PICT 2011 No. 0080. Also, we acknowledge the financial support provided by Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) through doctoral grant no. 4936.

References

Aggarwal, R., Verma, K., Miller, J., Milnor, W., 2004. Constraint driven web service composition in METEOR-S. In: Proceedings of the 2004 IEEE International Conference on Services Computing, pp. 23–30.

Ai, L., Tang, M., 2008. A penalty-based genetic algorithm for QoS-aware Web service composition with inter-service dependencies and conflicts. In: Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation.

Alonso, G., Casati, F., Kuno, H., Machiraju, V., 2004. Web Services. Concepts, Architectures and Applications. Springer-Verlag, Berlin, Heidelberg.

Arnold, W., Eilam, T., Kalantar, M., Konstantinou, A.V., Totok, A.A., 2008. Automatic realization of SOA deployment patterns in distributed environments. In: Proceedings of the Service-Oriented Computing. Springer, Berlin Heidelberg, pp. 162–179.

Balke, W.T., Wagner, M., 2003. Towards Personalized Selection of Web services. In WWW (Alternate Paper Tracks), pp. 20–24.

Bartalos, P., Bieliková, M., 2011. Automatic dynamic web service composition: a survey and problem formalization. Comput. Inform. 30, 793–827.

Bell, M., 2010. SOA Modeling Patterns for Service-Oriented Discovery and Analysis. John Wiley and Sons, Inc., United States.

Bellur, U., Gupta, A., Vadodaria, H., 2008. Semantic Matchmaking Algorithms. INTECH Open Access Publisher, Croatia, pp. 586–604.

Bertoli, P., Pistore, M., Traverso, P., 2010. Automated composition of Web services via planning in asynchronous domains. Artif. Intell. 174 (3–4), 316–361.

Bhakti, M.A.C., Abdullah, A.B., 2011. Autonomic computing approach in service oriented architecture. In: Proceedings of the 2011 IEEE Symposium on Computers & Informatics, Kuala Lumpur, pp. 231–236.

Bhakti, M.A.C., Abdullah, A.B., Jung, L.T., 2010. Autonomic, self-organizing service-Oriented Architecture in service ecosystem. In: Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies, Dubai, pp. 153–158.

Birukou, A., Blanzieri, E., D'Andrea, V., Giorgini, P., Kokash, N., 2007. Improving web service discovery with usage data. IEEE Softw. 24 (6), 47–54.

Blake, M.B., Wei, Y., 2010. Service-oriented computing and cloud computing: challenges and opportunities. IEEE Internet Comput. 14 (6), 72–75.

Canfora, G., Di Penta, M., Esposito, R., Villani, M.L., 2005. An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the ACM 2005 conference on Genetic and evolutionary computation, pp. 1069–1075.

Cardoso, J., Sheth, A., 2003. Semantic e-workflow composition. J. Intell. Inf. Syst. 21 (3), 191–225.

Chan, N.N., Gaaloul, W., Tata, S., 2012. A recommender system based on historical usage data for web service discovery. Serv. Oriented Comput. Appl. 6 (1), 51–63.

Chan, P., Lyu, M., 2008. Dynamic web service composition: a new approach in building reliable web service. In: Proceedings of the IEEE International Conference on Advanced Information Networking and Applications, pp. 20–25.
Crasso, M., Zunino, A., Campo, M., 2008. Easy web service discovery: a query-by-example approach. Sci. Comput. Program. 71 (2), 144–164.
Crasso, M., Mateos, C., Zunino, A., Campo, M., 2010. EasySOC: making web service outsourcing easier. Inf. Sci. 259 (0), 452–473.
Daigneau, R., 2011. Service Design Patterns: fundamental design solutions for SOAP/WSDL and restful Web services. Addison-Wesley, United States.
Diaz-Pace, J.A., Soria, A., Rodriguez, G., Campo, M., 2012. Assisting conformance checks between architectural scenarios and implementation. Inf. Softw. Technol. 54, 448–466.
Dong, H., Hussain, F.K., Chang, E., 2013. Semantic web service matchmakers: state of the art and challenges. Concurr. Comput.: Pract. Exp. 25 (7), 961–988.
Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J., 2004. Similarity search for Web services. In: Proceedings of the Thirtieth international conference on Very large data bases, vol. 30, pp. 372–383.
Doshi, P., Goodwin, R., Akkiraju, R., Verma, K., 2004. Dynamic workflow composition using markov decision processes. In: Proceedings of the 2004 International Conference on Web services, pp. 576–582.
Dustdar, S., Schreiner, W., 2005. A survey on Web services composition. Int. J. Web Grid Serv. 1 (1), 1–30.
El Falou, M., Bouzid, M., Mouaddib Al, Vidal, T., 2008. Automated web service composition using extended representation of planning domain. In: Proceedings of the 2008 IEEE International Conference on Web services, Beijing, pp. 762–763.
Elgedawy, I., 2013. Web services Conversation Adaptation Using Conditional Substitution Semantics of Application Domain Concepts. Hindawi Publishing Corporation. ISRN Software Engineering, Cairo.
Elgedawy, I., 2009. Automatic generation for Web services conversations adapters. In: Proceedings of the 24th IEEE International Symposium on Computer and Information Sciences, pp. 616–621.
Erickson, J., Siau, K., 2008. Web service, service-oriented computing, and service-oriented architecture: separating hype from reality. J. Database Manag. 19 (3), 42–54.
Erl, T., 2008. SOA Design Patterns. Pearson Education, United States.
Fanjiang, Y.Y., Syu, Y., 2014. Semantic-based automatic service composition with functional and non-functional requirements in design time: a genetic algorithm approach. Inf. Softw. Technol. 56 (3), 352–373.
Fanjiang, Y.Y., Syu, Y., Wu, C.H., Kuo, J.Y., Ma, S.P., 2010. Genetic algorithm for QoS-aware dynamic Web services composition. In: Proceedings of the 2010 International Conference on Machine Learning and Cybernetics (ICMLC), vol. 6, pp. 3246–3251.
Garofalakis, J.D., Panagis, Y., Sakkopoulos, E., Tsakalidis, A.K., 2006. Contemporary web service discovery mechanisms. J. Web Eng. 5 (3), 265–290.
Hassanzadeh, A., Namdarian, L., Elahi, S.B., 2011. Developing a framework for evaluating service oriented architecture governance (SOAG). Knowl.-Based Syst. 24 (5), 716–730.
Hassine, A.B., Matsubara, S., Ishida, T., 2006. A constraint-based approach to horizontal web service composition. In The Semantic Web-ISWC. Springer Berlin Heidelberg, pp. 130–143.
Hatzi, O., Vrakas, D., Nikolaidou, M., Bassiliades, N., Anagnostopoulos, D., Vlahavas, L., 2012. An integrated approach to automated semantic web service composition through planning. IEEE Trans. Serv. Comput. 5 (3), 319–332.
Issarny, V., Georgantas, N., Hachem, S., Zarras, A., Vassiliadist, P., Autili, M., Hamida, A.B., 2011. Service-oriented middleware for the Future Internet: state of the art and research directions. J. Internet Serv. Appl. 2 (1), 23–45.
Jiang, H., Yang, X., Yin, K., Zhang, S., Cristoforo, J.A., 2011. Multi-path QoS-aware web service composition using variable length chromosome genetic algorithm. Inf. Technol. J. 10 (1), 113–119.
Küster, U., Stern, M., König-Ries, B., 2005. A classification of issues and approaches in automatic service composition. In Proc. 1st Intl. Workshop on Engineering Service Compositions (WESC'05), pp. 25–33.
Kawamura, T., Hasegawa, T., Ohsuga, A., Paolucci, M., Sycara, K., 2005. Web services lookup: a matchmaker experiment. IT Prof. 7 (2), 36–41.
Kitchenham, B.A., Charters, S., 2007 Guidelines for performing Systematic Literature Reviews in Software Engineering. In Technical report, Ver. 2.3 EBSE Technical Report. EBSE.
Klusch, M., 2012. Overview of the S3 Contest: Performance Evaluation of Semantic Service Matchmakers. Semantic Web Services. Springer, Berlin Heidelberg, pp. 17–34.
Klusch, M., Gerber, A., Schmidt, M., 2005. Semantic web service composition planning with owls-xplan. In: Proceedings of the AAAI Fall Symposium on Semantic Web and Agents, USA, AAAI Press.
Klush, M., 2008. Semantic Web service Coordination. CASCOM: Intelligent Service Coordination in the Semantic Web. Springer, Germany, pp. 59–104.
Kokash, N., Birukou, A., D'Andrea, V., 2007. Web service discovery based on past user experience. In Business Information Systems, Springer Berlin Heidelberg, pp. 95–107.
Korfhage, R.R., 1997. Information Retrieval and Storage. John Wiley &Sons, New York.
Kuck, J., Gnasa, M., 2007. Context-sensitive service discovery meets information retrieval. In: Proceedings of The Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops' 07, White Plains, New York, pp. 601–605.
Kun, C., Xu, J., Reiff-Marganiec, S., 2009. Markov-htn planning approach to enhance flexibility of automatic web service composition. In: Proceedings of the 2009 IEEE International Conference on Web services, Los Angeles, pp. 9–16.
Kuzu, M., Cicekli, N.K., 2012. Dynamic planning approach to automated web service composition. Appl. Intell. 36 (1), 1–28.
Lécué, F., Léger, A., Delteil, A., 2008 DL reasoning and AI planning for Web service composition. In: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology 1, Sydney, pp. 445–453.
Lee, J., Lee S-j, Chen, H.-M., Wu, C.-L., 2012. Composing web services enacted by autonomous agents through agent-centric contract net protocol. Inf. Softw. Technol. 54 (9), 951–967.
Li, K., Verma, K., Mulye, R., Rabbani, R., Miller, J.A., Sheth, A.P., 2006. Designing semantic web processes: The WSDL-s approach. In Semantic Web services, Processes and Applications, Springer US, pp. 161–193.
Li, X., Tang, X., Song, Z., Yuan, X., Chen, D., 2010. AFlow: An Automated Web services composition system based on the AI planning and workflow. In: Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, vol. 2, pp. 1067–1071.
Liang, Q.A., Chung, J.Y., Miller, S., Ouyang, Y., 2006. Service pattern discovery of web service mining in web service registry-repository. In: Proceedings of the 2006 IEEE International Conference on e-Business Engineering, Shanghai, pp. 286–293.
Liu, H., Zhong, F., Ouyang, B., Wu, J., 2010. An approach for QoS-aware web service composition based on improved genetic algorithm. In: Proceedings of the 2010 International Conference on Web Information Systems and Mining, Sanya, vol. 1, pp. 123–128.
Liu, J., Li, J., Liu, K., Wei, W., 2007. A hybrid genetic and particle swarm algorithm for service composition. In: Proceedings of the Sixth International Conference on Advanced Language Processing and Web Information Technology, Luoyang, Henan, pp. 564–567.
Lo, W., Yin, J., Li, Y., Wu, Z., 2015. Efficient web service QoS prediction using local neighborhood matrix factorization. Eng. Appl. Artif. Intell. 38, 14–23.
Loutas, N., Peristeras, V., Zeginis, D., Tarabanis, K., 2012. The semantic service search engine (S3E). J. Intell. Inf. Syst. 38 (3), 645–668.
Ma, J., Zhang, Y., He, J., 2008. Efficiently finding Web services using a clustering semantic. In: Proceedings of the 2008 International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation, Beijing, China.
Manikrao, U. Prabhakar, 2005. Dynamic Selection of Web services with Recommendation System. In: Proceedings of the 2005 International Conference on Next Generation Web services Practices, Korea, pp. 117–121.
Mannava, V., Ramesh, T., 2012. Composite design pattern for feature-oriented service injection and composition of wed services for distributed computing systems with service oriented architecture. Int. J. Web Semant. Technol. 3 (3), 73–84.
Mansouri, D., Hamdi-Cherif, A., 2011. Ontology-oriented case-based reasoning (CBR) approach for trainings adaptive delivery. In: Proceedings of the 15th WSEAS International Conference on Computers, Corfu Island, pp. 328–333.
Martin, D., Burstein, M., McDermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Srinivasan, N., 2007. Bringing semantics to Web services with OWL-S. World Wide Web 10 (3), 243–277.
McDermott, D.V., 2002. Estimated-Regression Planning for Interactions with Web services. AIPS, p. 2.
Mecar, I., Devlic, A., Trzec, K., 2005. Agent-oriented semantic discovery and matchmaking of Web services. In: Proceedings of the 8th International Conference on Telecommunications, ACM Press, Zagreb, pp. 45–50.
Mistry, S.K., Kamal, M.H., Mistry, D., 2012. Semantic discovery of web services through social learning. Proc. Technol. 3, 167–177.
Mokhtar, S.B., Raverdy, P.G., Urbieta, A.A., Cardoso, R.S., 2010. Interoperable semantic and syntactic service discovery for ambient computing environments. Innov. Appl. Ambient. Intell.: Adv. Smart Syst., 213.
Narayanan, S., McIlraith, S., 2002. Simulation, verification and automated composition of Web services. In: Proceedings of the 11th ACM International Conference on World Wide Web, New York, pp. 77–88.
Ngan, L.D., Kanagasabai, R., 2013. Semantic Web service discovery: state-of-the-art and research challenges. Personal. Ubiquitous Comput. 17 (8), 1741–1752.
Oh, S.C., Lee, D., Kumara, S.R.T., 2008. Effective web service composition in diverse and large-scale service networks. IEEE Trans. Serv. Comput. 1 (1), 15–32.
Paik, I., Chen, W., Huhns, M., 2012. A scalable architecture for automatic service composition. IEEE Trans. Serv. Comput. PP 99 1–1.
Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K., 2002. Semantic matching of Web services capabilities. In: Proceedings of the Semantic Web—ISWC 2002, LNCS 2342, Springer Berlin Heidelberg, pp. 333–347.
Papazoglou, M.P., Heuvel, W., 2006. Service-oriented design and development methodology. Int. J. Web Eng. Technol. 2, 412–442.
Pathak, J., Koul, N., Caragea, D., Honavar, V., 2005. A Framework for Semantic Web services Discovery. In: Proceedings of the 7th ACM International Workshop on Web Information and Data Management, IEEE Xplore, New York, pp. 603–607.
Peer, J., 2005. Web Service Composition as AI Planning: A Survey. University of St. Gallen, Switzerland.
Platenius, M.C., von Detten, M., Becker, S., Schafer, W., Engels, G., 2013. A survey of fuzzy service matching approaches in the context of on-the-fly computing. Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering, ACM, New York, pp. 143–152.
Ponnekanti, S.R., Fox, A., 2002. SWORD: A developer toolkit for Web service composition. In: Proceedings of the 11th World Wide Web Conference, Honolulu,

HI, USA.
Rambold, M., Kasinger, H., Lautenbacher, F., Bauer, B., 2009. Towards autonomic service discovery a survey and comparison. In IEEE International Conference on Services Computing, Bangalore, pp. 192–201.
Ramirez, A.J., Cheng, B.H.C., 2010. Design patterns for developing dynamically adaptive system. In Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, ACM, New York, USA, pp. 49–58.
Rao, J., Su, X., 2005. A survey of automated web service composition methods. Semant. Web Serv. Web Process. Compos., 43–54, Springer Berlin Heidelberg.
Rodríguez, G.H., Soria, A., Campo, M., 2014. From software architecture descriptions to object-oriented designs. In XLIII Jornadas Argentinas de Informática e Investigación Operativa (43JAIIO)-Doctoral Consortium (IJCAI), Buenos Aires.
Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Fensel, D., 2005. Web service modeling ontology. Appl. Ontol. 1 (1), 77–106.
Rong, W, Liu, K., 2010. A survey of context aware web service discovery: from user's perspective. In: Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), Nanjing, pp. 15–22.
Sangers, J., Frasincar, F., Hogenboom, F., Chepegin, V., 2013. Semantic Web service discovery using natural language processing techniques. Expert. Syst. Appl. 40 (11), 4660–4671.
Santofimia, M.J., Moya, F., Villanueva, F.J., Villa, D., Lopez, J.C., 2008. An agent-based approach towards automatic service composition in ambient intelligence. Artif. Intell. Rev. 29 (3–4), 265–276.
Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., Mei, H., 2007. Personalized QoS Prediction for Web services via Collaborative Filtering. In: Proceedings of the 2007 IEEE International Conference on Web services, USA, pp. 439–446.
Shvaiko, P., Euzenat, J., 2005. A survey of schema-based matching approaches. J. Data Semant. IV, 146–171, Springer Berlin Heidelberg.
Sirin, E., Parsia, B., Wu, D., Hendler, J., Nau, D., 2004. HTN planning for web service composition using SHOP2. Web Semant.: Sci., Serv. Agents World Wide Web 1 (4), 377–396.
Sivashanmugam, K., Verma, K., Sheth, A., Miller, J., 2003. Adding Semantics to Web service Standards. In: Proceedings of the 1st International Conference on Web services, Las Vegas, NV, pp. 395–401.
Sreenath, R.M., Singh, M.P., 2004. Agent-based service selection. Web Semant.: Sci., Serv. Agents World Wide Web 1 (3), 261–279.
Stavropoulos, T.G., Vrakas, D., Vlahavas, I., 2013. A survey of service composition in ambient intelligence environments. Artif. Intell. Rev. 40 (3), 247–270.
Stroulia, E., Wang, Y., 2005. Structural and semantic matching for assessing web-service similarity. Int. J. Coop. Inf. Syst. 14 (4), 407–437.
Strunk, A., 2010. Qos-aware service composition: A survey. In: Proceedings of the IEEE 8th European Conference on Web Services, Ayia Napa, pp. 67–74.
Suraci, V., Mignanti, S., Aiuto, A., 2007. Context-aware semantic service discovery. In: Proceedings of the 16th IEEE IST-Mobile and Wireless Communications Summit, Budapest, pp 1–5.
Syu, Y., Ma, S.P., Kuo, J.Y., FanJiang, Y.Y., 2012. A survey on automated service composition methods and related techniques. In: Proceedings of the IEEE Ninth International Conference on Services Computing, Honolulu, pp. 290–297.
Tan, W., Fan, Y., Zhou, M., 2009. A petri net-based method for compatibility analysis and composition of Web services in business process execution language. IEEE Trans. Autom. Sci. Eng. 6 (1), 94–106.
Tang, X., Jiang, C., Zhou, M., 2011. Automatic Web service composition based on Horn clauses and Petri nets. Expert. Syst. Appl. 38 (10), 13024–13031.
Tang, M., Ai, L., 2010. A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In: Proceedings of the 2010 IEEE Congress on Evolutionary Computation, Barcelona, pp. 1–8.
Tekinerdogan, B., Aksit, M., 2002. Synthesis-based software architecture design. In: Aksit, M. (ed) Software Architectures and Component Technology. Springer US volume 648 of The Springer International Series in Engineering and Computer Science, pp. 143–173.
van der Aaslt, W.M.P., 2005. YAWL: yet another workflow language. Inf. Syst. 30 (4), 245–275.
Wagner, F., Ishikawa, F., Honiden, S., 2011. Qos-aware automatic service composition by applying functional clustering. In: Proceedings of the 2011 IEEE International Conference on Web services (ICWS), Washington, pp. 89–96.
Wang, H., Zhou, X., Zhou, X., Li, W., 2010. Adaptive Service Composition Based on Reinforcement Learning. Service-Oriented Computing. Springer, Berlin Heidelberg, pp. 92–107.
Wang, Y., Stroulia, E., 2003. Flexible interface matching for web-service discovery. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003, pp. 147–156.
Xiao, H., Zou, Y., Ng, J., Nigul, L., 2010. An approach for context-aware service discovery and recommendation. In: Proceedings of the 2010 IEEE International Conference on Web services (ICWS), Miami, pp. 163–170.
Zhang, W., Chang, C.K., Feng, T., Jiang, H.Y., 2010. QoS-based dynamic web service composition with ant colony optimization. In: Proceedings of the IEEE 34th Annual Computer Software and Applications Conference, Seoul, pp. 493–502.
Zhuge, H., Liu, J., 2004. Flexible retrieval of Web services. J. Syst. Softw. 70 (1), 107–116.
Zou, G., Chen, Y., Xu, Y., Huang, R., Xiang, Y., 2012. Towards automated choreographing of Web services using planning. In: Proceedings of the AAAI Conference on Artificial Intelligence.
Zou G., Chen, Y., Yang, Y., Huang, R., Xu, Y., 2010. AI planning and combinatorial optimization for web service composition in cloud computing. In: Proceeding of the 2010 international conference on cloud computing and virtualization, Singapore, pp. 1–8.