2009 IEEE International Conference on Services Computing

Towards Autonomic Service Discovery – A Survey and Comparison

Michael Rambold, Holger Kasinger, Florian Lautenbacher and Bernhard Bauer
Department of Computer Science
University of Augsburg
86135 Augsburg, Germany
{*rambold, kasinger, lautenbacher, bauer*}*@informatik.uni-augsburg.de*

***Abstract*—Service-oriented architecture has become the standard paradigm for software component integration. However, with the permanently increasing amount of available services and dynamic changes, the complexity of such service infrastructures, their maintenance, and consequently the expenditures spent for their operation increase equally. To deal with these effects, an improvement of service composition and discovery becomes necessary, especially a higher degree of automation. Following the idea of Autonomic Computing, which similarly aims at automating processes and workflows to a high degree, service composition and discovery have to proceed autonomously, which will on the one hand side reduce human involvement to a minimum, but on the other side require certain capabilities on the part of these mechanisms. For these purposes, in this paper we define prime criteria that have to be fulfilled for an autonomic service discovery. Based on that we present a comprehensive survey on existing service discovery approaches and evaluate to which extent they already fulfill these criteria. As a result, the paper reveals that there already exist some approaches that support or even fulfill a couple of the proposed criteria, which principally enables autonomic properties, but what is missing is an holistic approach focusing explicitly on providing autonomic properties.**

I. Introduction

Service-oriented architecture (SOA) has gained a lot of attention over the last years and has become the de-facto standard for web application and software component integration. SOA offers a frame in which loosely coupled software services can be created, managed, integrated, and combined. The main goal of SOA is a software architecture aligned to business processes, which can easily react on dynamic requirement changes and new consumer, regulatory, or competitive demands. As the number of services in the Internet and company intranets is steadily increasing and as requirements and demands are permanently changing, the complexity of service composition increases equally.

Already shallow changes (where change effects are restricted to the clients of that service, cf. [1]) require an adaptation of invoking services, registry entries, and service orchestrations. These modifications become more complex the more services need to be orchestrated: each service description needs to be actualized in the registry, the invocation of the service by other services needs to be modified, and already completed service orchestrations (e. g. in WS-BPEL [2]) need to be changed again. Because these modifications are mostly performed manually, they cause increasing operational expenditures and henceforth lead to reduced profit and higher total cost of ownership (TCO) of the service infrastructure.

To tackle these effects, one of the most notable research challenges for service-oriented computing is an autonomic composition of services (cf. [3]). This in turn requires an enhanced and automated service discovery that demands minimal user involvement. In other words, following the idea of Autonomic Computing (AC) [4] (for a more detailed overview on AC see e. g. [5], [6]), service discovery in the future has to be autonomic itself in order to enable an autonomic service composition, i. e. *self-configuring* (e. g. fully automatic discovery based on an unambiguous service description without prior registration), *self-optimizing* (e. g. ranking the discovery results based on the actually provided quality of service and cost of service), *self-healing* (e. g. being capable to cope with faults and network changes during discovery as well as recover from errors), and *self-protecting* (e. g. recognizing malicious services automatically and replace them in the results of a service discovery by other matching services). As a result, the discovery of services will be accomplished autonomously given only high-level objectives from administrators or users. The great complexity of service discovery will be transferred to the service infrastructure and human involvement will be reduced significantly, which will reduce TCO equally.

However, it is incontestable that achieving this high degree of automation and autonomy at once is virtually impossible. There exists a broad spectrum of autonomic maturity (see [7]) starting from the basic level, at which administrators or users perform all necessary tasks manually, right up to the autonomic level, at which service discovery will be automatically managed by established business rules and policies. But the higher the desired level of autonomic maturity, the more specific capabilities are required on the part of the service discovery mechanisms.

In this paper we screen a great number of currently existing service discovery approaches and provide a thorough insight into the current state of research towards the vision of an autonomic service discovery. We define prime criteria that are essential for an autonomic service discovery and the enabling of self-* properties (Section II). We survey existing

service discovery approaches (Section III) and compare as well as evaluate them on the basis of these criteria (Section IV). Finally, concluding remarks and an outlook on future research challenges for an autonomic service discovery are given (Section V).

II. Autonomic Service Discovery Criteria

A prerequisite for any service discovery approach enabling basic self-configuring and self-optimizing capabilities is the provisioning of expressive *service descriptions*, the enabling of high *matchmaking* levels, the provisioning of fundamental *service composition* functionality, as well as the involvement of actual *Quality of Service* (QoS) and *Cost of Service* (CoS) descriptions. For the enabling of basic self-healing capabilities, an approach is additionally expected to fulfill at least a high degree of *scalability*, *robustness*, and *up-to-dateness*. Finally, basic self-protecting capabilities in addition require at least policies or service level agreements contained in the QoS descriptions, which are appropriate to indicate malicious service behaviors, as well as a proper *service replacement* mechanism. Please note that some of these criteria are relevant for more than one self-* property. For instance, a proper service replacement mechanism is not only required for self-protection but also for self-healing. The reason is that self-* properties usually overlap and are closely linked to each other.

In summary, this yields the following eight prime criteria, which we will use for the comparison and evaluation of existing service discovery approaches in Section IV:

- *Service Description*: In order to automate a service discovery, the service description should define the functionality, scope, behavior and intention of a service in an unambiguous way. Already today, some description languages and technologies exist that annotate (web) services with ontologies, see e. g. OWL-S [8], WSMO [9], WSDL-S [10] and SAWSDL [11]. In general, the more distinct the contained information, the better is the precision of the retrieval.
- *Matchmaking/Reasoning*: This criterion is closely related to the expressiveness of the service description. The better the included service information are utilized to differentiate between offered services for combining them with a request, the better is the quality of the retrieval and the higher is the degree of automation.
- *Scalability*: The performance of a service discovery regarding message and space overhead considering increasing inquiries is an important factor for possible application scenarios. Apparently, it correlates with the architectural degree of decentralization (centralized, distributed, decentralized, cf. [12]) and to what extend the publication and request communication is distributed among multiple subsystems as well as managed by an active load-balancing mechanism.
- *Robustness*: A discovery mechanism should be capable to cope with faults and network changes without considerable performance or function losses. An example is recovery from errors or the structural re-stabilization of multiple sites, which relates the robustness closely to the degree of decentralization.
- *Service Composition*: Service discovery is expected to support service composition by retrieving the requested set of services. However, single services very seldom correspond to a request in a direct manner. Thus, a service discovery mechanism should also include service composition capabilities. This makes service composition a part of the service discovery and vice versa. Due to this iterative request splitting, the reutilization and recall of services can be optimized.
- *Quality of Service and Cost of Service*: In order to allow for a refined selection of services matched as equal, non-functional service criteria such as QoS and CoS should be included in the discovery mechanism. QoS represents a useful descriptive container to formulate security issues and other constraints for invocation. Ideally, multiple QoS and CoS combinations should be offered and automatically selected.
- *Up-to-dateness*: A further issue of service discovery is the up-to-dateness of utilized service information. The discovery process should consider the availability of a service respectively the probability that a service description may be outdated due to a change.
- *Service Replacement*: It would be desirable, if complementary services are offered respectively cached to allow for a replacement in case of a service failure.

In their entirety these criteria are vital for the provisioning of autonomic capabilities for service discovery. Of course, various (other) criteria for service discovery have been already proposed in diverse surveys (see e. g. [13], [14], [15], [16], [17]) by the web service community. However, these surveys evaluate publicly available service registries or categorize the latest service discovery approaches and therefore establish more general criteria, without regarding the special needs of autonomic service discovery. Similarly, the originally defined service description [18] and web service architecture [19] requirements also fail to incorporate autonomic requirements. In contrast, the AC community has identified the need for autonomic capabilities in SOAs already for a while (see e. g. [20]), but concentrates more on the realization of autonomic SOAs in their entirety (see e. g. [21]) along with the respective functional requirements [22], but did not explicitly focus on criteria for an autonomic service discovery as an integral part of autonomic SOAs, yet. Nevertheless, subsets of the criteria and requirements proposed by both communities have been reused in this paper. In addition to them, sophisticated architectural requirements and automation aspects have been considered (cf. [12]).

III. SURVEY

Today, a vast number of diverse service discovery approaches exists. In order to facilitate a reasonable overview and to indicate their potential application fields, we categorize the screened approaches along two dimensions (see Figure 1), their architectural degree of decentralization (i. e. centralized, distributed, or decentralized) and their matchmaking respectively reasoning level (i. e. syntactical, hybrid, or semantical). Please note that hybrid matchmaking is not based on syntactic information only, but either reveals hidden semantic information or combines syntactic with semantic matchmaking mechanisms. The UDDI registry [23] with its syntactical matchmaking serves as a basis for both dimensions. The quadrants spanned by both dimensions are ranked by numbers from 1 to 9. Starting with syntactical and centralized approaches similar to the mentioned UDDI approach in the bottom left quadrant, the ranking goes up to the top right quadrant, whereas it prefers the matchmaking and reasoning level over the degree of decentralization, which results in a meandering enumeration.

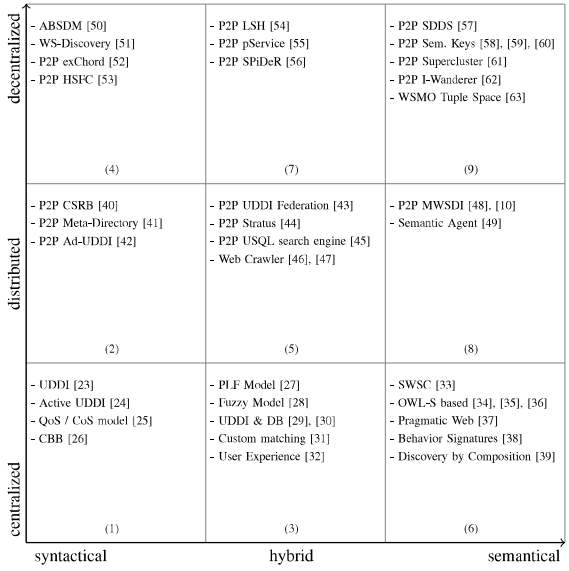

Figure 1. Categorization of service discovery approaches

The service discovery approaches described below are surveyed with focus on the described criteria in the order of centralized to decentralized registry architectures. Each architectural category is then differentiated regarding its matching level from syntactical to semantical involvement.

A. Centralized Registry Architectures

1) Syntactic Matching: The Active UDDI [24] is an information maintenance extension of the UDDI. An active service is set as a proxy between the original registry and their consumers which holds state information of all services. Thus, all retrieved service information are up-to-date. In addition, it offers a service replacement through a publish-subscribe process.

In [25] a QoS and CoS model is established while the search functionality is still based on an UDDI. Stored services provide a list of trade-offs between offered QoS and requested CoS. Service requester hold information about desired cost and benefit with upper and lower limitations. An integrative selection can be achieved by comparing the alternatives regarding cost and quality within the request boundaries.

Towards hybrid matchmaking, in [26] the service description respectively WSDL operations and service types are extended by constraints e. g. the attribute *price* of a type *book* is an integer with range from 0 to 1000. The UDDI is extended to a constraint-based brokering (CBB) whose matchmaking does not exceed or harm a provided constraint.

2) Hybrid Matching: In [27] a discovery mechanism is established that uses WSDL as input, but reveals hidden semantic concepts behind words for matching purpose by extracting words with their frequency such as name and textual description. These terms are in turn used as input of a Bayesian Network model, whose variables are adapted to correspond to the service description. The approach expresses that similar network variables refer to similar services. These latent variables are stored as vectors and are used to cluster the description space. A service request is alike preprocessed by assigning to a cluster. The vector distance to all other services of this cluster is measured and those with the smallest distance are filtered with QoS parameters to retrieve a ranked response. Some approaches (e. g. [28]) are extending the UDDI data-model with OWL-S, but use a combined syntactical and semantical matching approach. In [28] the so-called fuzzy matching decomposes a request to atomic processes, measures their overall similarity to stored services regarding its weighted description, functional and QoS similarity and compose suitable atomic processes to retrieve a service. After the search corpus is scaled down by a thesaurus keyword search, the functional similarity of each atomic process is measured according to their amount of common process attributes and, subtracted from this, their different attributes of the same ontology. The result candidate set is further processed with QoS and CoS constraints.

Established information retrieval mechanisms from other research fields have been transferred to the field of web service discovery to reveal hidden semantics. In [29] and [30] UDDI is extended by a database system. Each tag in WSDL is transformed into a path from the root tag and, according to the path, stored in a general pre-ordered tree structure [29]. A service is converted to a vector whose elements refer to itemIDs of the tree and are weighted regarding their occured frequency. The vector distance between a request

and stored descriptions is used for matching purposes. SQL queries with focus on the vector distance are used to search the database for a link to UDDI registry entries.

[31] also uses a vector space model with syntactical information retrieval concepts. The particular characteristic of this approach is the customizable hybrid matching. All matching techniques can be mixed and processed in parallel, cascaded to shrink the search corpus or changed according to customized desires.

User Experience retrieval [32] assumes that users with similar interests have similar significant action sequences. Data mining processes are used to discover similar patterns from monitored user actions that may refer to the similar services and give recommendations for an action that should be performed like a search request.

3) Semantic Matching: [38] combines OWL-S with a finite state automata to describe the service behavior in a semantic manner whereby it allows for a service composition. [34], [35] and [36] extend an UDDI registry matching process respectively its data model with OWL-S. In [35], annotated services are pre-computed during registration. Each concept node of an ontology tree has a list of services, including their matching degree (exact, plugin, subsume). An advertisement is added to each concept list that corresponds to one of its input or output concepts. To match a request, for each of its input and output concepts, the concept list is called. A registered service that is included in each of the referring concept lists is retrieved.

Semantic Web Service Clustering (SWSC) [33] is an UDDI extension that also uses OWL-S as semantic extension in combination with the WordNet ontology. Similar to the syntactic information retrieval approaches, purged terms are extracted to calculate the similarity between all registered services and assigned to similar clusters.

A semantic service discovery through service composition is presented in [39]. If a single service cannot fulfill the requested input and output descriptions, then services are searched that fulfill at least one original output. If a set of these discovered services can now fulfill the output, but not their original input, then the process is repeated by searching new services, that fulfill the new input as output, until the original request is satisfied through discovered service composition. Additionally, QoS parameter are influencing the choice between similar services.

The Pragmatic Web consists of tools, practices, and theories that describe why and how people use information [64]. A service is divided into their role (*who*), context parameters (*why*), the concept (input and output, *what* part) as similar semantic services are not implicitly similar to the user ideas [37]. Therefore, matching of the *why* part by a combination of natural processing techniques is necessary in addition to the ontology enriched *what* part by semantic similarity of input and output characteristics.

B. Distributed Registry Architectures

1) Syntactic Matching: In [40], a Peer-to-Peer (P2P) overlay network consists of multiple peer groups with one super peer as a local registry. Seemingly similar services are stored in a peer group. A service request of a peer group is first matched in the super peer registry. If the request cannot be satisfied, it is forwarded to a centralized broker, the Common Service Registry Broker, that mediates between different peer group registries.

[41] is also based on a P2P communication layer with meta-directory nodes that cluster the description corpus. Meta-directory peers store the location of UDDIs as keys and are searched before a certain UDDI registry is invoked. If the requested service is not found in a meta-directory, the request is broadcasted to the other meta-directories. Therefore, the load is shifted from the UDDI registries to the meta-directories and to the P2P network.

[42] stresses the realtime status of services information. A root registry serves as central entry point to a P2P network of specialized Active and Distributed UDDI (Ad-UDDI) registries. Service information are duplicated and deployed to Ad-UDDIs of the same domain concept. Web service state informations are cached through a monitoring process in each registry. A request is first looked up in the cache, then it is checked in the own registry or send to registries of the same domain if necessary. In addition, Ad-UDDIs allow for a service replacement.

2) Hybrid Matching: In [43] a federation of UDDI registries is presented that uses WSDL-S beside a syntactical search functionality to improve the matchmaking. Registries are clustered w. r. t. their domains in small federations. A request is either semantically matched against the local registry or send as a federated query to the directory that mediates between different tModels (technical data structure of an UDDI). Due to the different kinds of ontologies and registries, the tModel directory is used to translate a service request to the relevant registries and their concepts. After matching the translated request against stored services, the results are aggregated and the most suitable service is retrieved.

[44] is using a tuple model (meta-data, timestamps and relations) including syntactical or semantical information in the meta-data. Different P2P layers are combined to bridge the gap between different heterogeneous registries. Thereby, an operation layer offers an unified publication and inquiry API. A double-overlay P2P topology clusters the registries w. r. t. their domain and interconnects the clusters with a backbone network. On top of each registry a communication and inquiry engine is running, which hides the inherent complexity of different query languages used in web service registries. Due to the included timestamps, retrieved service information are up-to-date. Moreover, experimental results show good scalability and robustness.

In [45] a service search engine with a plugin system is presented. Document handler plugins support a variety of descriptions (QoS, syntax, semantic). The search engine can access several service registries, which are clustered w. r. t. an OWL-S ontology. Search queries are based on a SQL like language, USQL, and make use of different handlers in parallel to retrieve best performance. The collected results of the handler are consolidated to compute the response.

Web search crawler approaches are collecting service descriptions published either directly on the Internet or in different registries in a vast number of formats (e. g. WSDL or OWL-S). Collected services are indexed in a registry, e. g. in the Web Service Repository Builder [47], or assembled to general description container that provide pointers to more specialized information records in registries [46]. This makes them error-prone and decreases the inquire performance apart from good service publication scalability.

3) Semantic Matching: METEOR-S Web Service Discovery Infrastructure (MWSDI) [48] and [10] provide a discovery mechanism over federated registries. A P2P overlay network is used alike previous mentioned approaches to access UDDIs. However, MWSDI uses a light-weight semantic matching, which works mainly on the semantic tags in WSDL-S descriptions with the UDDI specific search functions. Mappings from service parameters to ontological concepts are included in the tModels.

A semantic agent [49] can enrich search keywords through an ontology database. First, the agent sends the keywords to an ontology database and retrieves an ontology which includes the keywords. The client can select one ontology. Then the database is called again and the terms are annotated with the chosen ontology and send to UDDIs to get information of all suited services. The agent connects to these services to extract the inputs, outputs, preconditions and effects (IOPE) of an OWL-S description. The retrieved information are matched against the user query and may result in an exact, plugin or subsume match.

C. Decentralized Registry Architectures

1) Syntactic Matching: In Agent Based Service Discovery Mechanism (ABSDM) [50], an agent is not only used to connect and extract service information on command of an user, but represents a service descriptions itself. Agents are stored independently on distributed servers in Backus Naur form. Joining agents are stored in an agent tree structure under the most similar agent node as a child. The search and matching process uses a distributed and parallel search request over the whole tree. Thereby, an agent checks its own library first and forwards the search request to its children. If no suitable agent and hence no suitable service was found, the request will be forwarded to its own father.

[51] defines a service discovery based on multicast. A client searches for one or more target services by type or name by sending a probe message with at least one constraint to a multicast group. A target service that matches the probe respectively all contained constraints sends a response directly to the client. In order to retrieve network transport information of previous matched target services, a client sends a resolve message including web service addressing information. Again, if a target service matches the resolve message, it responds with a resolve match message. Due to this multicast design the retrieved service information are always up-to-date.

In [52] a P2P structure is extended to search with XPath, named eXCHORD. WSDL is converted into a node value tree. Nodes refer to WSDL tags respectively to their path from the root concept. Each node is hashed and distributed over the extended P2P network. The relationships and therefore the structure is preserved on the overlay structure as nodes refer to a path. Thus, the location of an information in the description is included. A query is also converted and matched against the node for range queries or matched against a single node with the key for an exact search.

WSDL documents span an abstract space of tags [53]. The complexity of this high dimensional search corpus is reduced to one dimension by a Hilbert Space Filling Curve (HSFC). The curve is arranged successively through the space and in doing so clusters are generated. A binary index is assigned to each cluster. Cluster can be further refined by the HSFC and may contain multiple nested clusters that have the same index prefix as their enveloping cluster. In course of the HSFC processing, a binary prefix tree of cluster is constructed. The binary tree index works as key for the P2P overlay network. Furthermore, the prefix of an index can be used for wildcard queries.

2) Hybrid Matching: The basic idea of [54] is to hash the input items with a specialized hash function. The locality sensitive hash function (LSH) maps similar service descriptions to the same buckets with a high probability. In the scope of web services each part of the service description such as an operation is hashed and stored as elements in an attribute vector. Each service description has a sourceID, that is published with the attribute vector to each node in the P2P network, whose key corresponds to a hashed element in the attribute vector. Similar services are hashed on a syntactical basis, but form an implicit search cluster due to the LSH and therefore reveal hidden semantics.

[55] combines a P2P overlay network for exact matches with a skip graph[1] for range queries. Thereby, path entries from a service description root element to a certain IOPE element are inserted to the skip graph at a position where the father element meets a syntactical prefix condition. A father element of the skip graph is distributed to all nodes that store a child entry to support range queries. Service

[1]A skip graph [65] is a distributed data structure based on double-linked lists, that provides the functionality of a balanced tree, whose elements are stored in separate nodes which may fail at any time.

description can be either WSDL or WSDL-S. If WSDL is used for description, then it gets annotated with OWL [66] symbols and translated to WSDL-S. In addition to a QoS support, this approach offers also actual state information and similar services in a replacement list to optimize the service choice.

[56] emphasizes the matching process. The architecture is likewise based on a P2P network with supernodes that are responsible for routing and querying and gathered when client peers are joining or leaving the network. The matching contains keywords, ontology based and behavior based search with BPEL [2] derived finite state automata. Apart from a composite matching possibility, a user-based QoS rating mechanism is presented.

3) Semantic Matching: In [57] the Semantically-enhanced Distributed Discovery System (SDDS) based on P2P is presented. Semantics are not specified by a certain ontology or language. Instead, an ontology manager is situated on each node of the P2P network and performs service discovery and deals with registration. Ontological subsets of possible input and output request concepts are distributed separately via the P2P hashing functionality. In addition, services with included operations are likewise stored in the network and are linked to a specific input or output subset concept. A requester defines possible input and output subsets of the request and continues by sending the retrieved input and output concepts separately via the Distributed Hash Table (DHT) lookup to certain nodes. Each called ontology manager searches for possible services that match the input respectively the output concept and returns the service with its operation. Resulting service sets that satisfy either the input or the output concept are intersected to achieve a matching result.

In [58] numerical keys refer to ontology concepts. Thereby a child concept has the same prefix as its father concept. Service advertisements and requests are vectors with encoded ontological concepts. A distributed tree structure is established by addressing concepts keys to a P2P node. Due to the prefix property the search can be performed on the tree. After the distributed tree returns a set of service advertisements, the matching engine that includes the knowledge base will complete the semantic matching.

In [59] services are likewise converted to vectors of numerical concept keys. In this P2P approach concept groups are created based on these keys. The distance between a concept group and a root concept group is calculated and used to separate the groups. A request is forwarded to a registry peer that converts the request analogue in a vector. The distance of the vector to the root concept is used to find suitable concept groups respectively services in this group that are matched regarding WSMO and QoS parameters.

Similar to this approach ontology concepts are also used as P2P keys in [60] and [67]. To search for a service, first an ontology has to be chosen and the node with the corresponding concept is retrieved. Then the sub-concepts and their input and output characteristics are defined. If the request is composed of several parts, then it is forwarded to the responsible nodes.

In [61] a double-overlay P2P network is presented to interconnect cluster managers on different layers. Supercluster manager are assigned to main goals of WSMO and may contain several clusters with corresponding sub-goals. A user request contains at least one WSMO goal. It is send to a cluster node on the lowest layer with the same goal. If the request has a broader goal or consists of further ones that are not in the scope of this cluster, then the request is send from the cluster head to the supercluster manager or even subsequent to multiple superclusters if necessary. Thus, obtained partial services are finally composed.

One of the most interesting approaches in the P2P environment is the I-Wanderer [62]. Query packets wander over nodes which are clustered based on the functional description based on OWL-S. They log visited concept clusters and their services. If two query packets meet coincidentally, they exchange their knowledge. If a suitable cluster is detected, then the query packet is proliferated and spread through the cluster. Services are evaluated through their QoS properties which include CoS besides timestamps that indicate their up-to-dateness. During the discovery process, other services can be found which are stored for future requests. Thus, this mechanisms enhance the efficiency of current and future search requests.

Semantic shared tuple spaces [63] based on WSMO is another quite interesting approach. Web service descriptions including ontological concepts are exchanged through the space and matched locally. A response can be reassembled of multiple sub-responses. After a response is put to the space, the corresponding request can be removed from the space or left in order to receive also future responses. Additionally, a publish-subscribe mechanism can be established by a requester to get informed about service changes or new available services.

IV. Comparison

The aforementioned service discovery approaches meet the one or other criterion proposed in Section II with distinct gradations due to various mechanisms. Nevertheless, none of them meets all criteria. In order to compare these approaches against each other in a transparent way, we have developed an evaluation model that assigns a weighting factor to each criterion to indicate its relevance for the evaluation as well as rates the mentioned gradations to distinguish the different approaches. The result of the comparison is listed in Table I.

A. Criteria Weighting

A numerical weighting factor is assigned to each criterion (see brackets in each criterion box in Table I) in order

Table I
COMPARISON OF SERVICE DISCOVERY APPROACHES

Approach / Criteria (quantifier)	Description (4)	Reasoning (4)	Scalability (4)	Robustness (4)	Composition (3)	QoS (CoS) (3)	Up-to-dateness (2)	Replacement (1)	Quadrant	∑ Evaluation Points
P2P I-Wanderer [62]	+	+	+	+	0	+ (+)	+	–	9	45
P2P pService [55]	+	+	+	+	–	+ (–)	+	+	7	44
P2P SPiDeR [56]	+	+	+	+	0	+ (–)	–	–	7	41
P2P Sem. Keys [58], [59], [60]	+	+	+	+	–	+ (+)	–	–	9	38
P2P Supercluster [61]	+	+	+	+	+	– (–)	–	–	9	38
P2P Stratus [44]	+	+	+	+	–	– (–)	+	–	5	36
WSMO Tuple Space [63]	+	+	+	0	+	– (–)	0	0	9	34
P2P SDDS [57]	+	+	+	+	–	– (–)	–	–	9	32
P2P USQL search engine [45]	+	+	0	0	–	+ (–)	–	–	5	30
Discovery by Composition [39]	+	+	–	–	+	+ (+)	–	–	6	28
Fuzzy Model [28]	+	+	–	–	+	+ (+)	–	–	3	28
OWL-S based [34], [35], [36]	+	+	–	–	0	+ (–)	0	–	6	27
P2P MWSDI [48], [10]	+	+	0	0	–	– (–)	–	–	8	24
P2P UDDI Federation [43]	+	+	0	0	–	– (–)	–	–	5	24
Behavior Signatures [38]	+	+	–	–	+	– (–)	–	–	6	22
P2P LSH [54]	–	0	+	+	–	– (–)	–	–	7	20
Web Crawler [46], [47]	+	+	0	–	–	– (–)	–	–	5	20
WS-Discovery [51]	–	–	+	+	–	– (–)	+	–	4	20
SWSC [33]	+	+	–	–	0	– (–)	–	–	6	19
P2P Ad-UDDI [42]	–	–	+	0	–	– (–)	+	+	2	18
Pragmatic Web[37]	+	+	–	–	–	– (–)	–	–	6	16
P2P HSFC [53]	–	–	+	+	–	– (–)	–	–	4	16
P2P exChord [52]	–	–	+	+	–	– (–)	–	–	4	16
Semantic Agent [49]	0	+	–	–	–	– (–)	–	–	8	12
ABSDM [50]	–	–	+	0	–	– (–)	–	–	4	12
P2P Meta-Directory [41]	–	–	+	0	–	– (–)	–	–	2	12
PLF Model [27]	–	0	–	–	–	+ (–)	–	–	3	10
Custom matching [31]	0	0	–	–	–	– (–)	–	–	3	8
P2P CSRB [40]	–	–	0	0	–	– (–)	–	–	2	8
User Experience [32]	–	0	–	–	–	0 (–)	–	–	3	7
QoS / CoS model [25]	–	–	–	–	–	+ (+)	–	–	1	6
Active UDDI [24]	–	–	–	–	–	– (–)	+	+	1	6
UDDI & DB [29], [30]	–	0	–	–	–	– (–)	–	–	3	4
CBB [26]	–	0	–	–	–	– (–)	–	–	1	4
UDDI [23]	–	–	–	–	–	– (–)	–	–	1	0

to express the relevance of a specific criterion for autonomic service discovery. The most important criteria for service discovery automation are an unambiguous service description (using semantic technologies) and the associated reasoning. A stable discovery architecture that is highly scalable w. r. t. an increase of services and inquiries as well as robust w. r. t. to failures or crashes is equally important. Service composition and QoS including CoS are relevant to improve complex search queries and to further automate the service selection, but are less weighted. Outdated service information may give rise to problems and end in additional searches that hamper an autonomic discovery. Although being less important compared to service composition and QoS and hence less weighted, the criterion is more important than complemental service replacement to avoid anew and identical searches, which has the lowest weighting.

B. Criteria Rating

In order to make the different criterion gradations clear, each entry has a rating range from – to zero to + . Each rating in turn is paired with corresponding numerical values ranging from 0 to 2. A minus will be assigned, if an approach does not provide or support concepts or mechanisms that meet this criterion at all, which yields 0 evaluation points[2]. The other ratings are assigned according to the fact, to what extent a criterion is met. For example, a plus is assigned for the use of semantic reasoning, good scalability due to a decentralized architecture, low message and space overhead, etc. A zero is assigned e. g. for revealing hidden semantics, distributed respectively hierarchical architectures, simplified horizontal service composition, etc.

C. Evaluation

The evaluated approaches are ordered based on their overall evaluation points (rightmost column of Table I). The evaluation points for an approach are calculated by summing up all numerical values assigned to its ratings, each multiplied with its corresponding criterion weighting. The second last column shows the corresponding quadrant of Figure 1 the approach has been categorized in.

Apparently, approaches with a higher quadrant ranking are likely to have higher evaluation points, even though there are some exceptions: For instance, the *Semantic Agent* [49] approach reveals that including only reasoning capabilities is not enough for an autonomic service discovery, but that the consideration of other criteria is important as well. In turn, the *Stratus* [44], *P2P USQL search engine* [45], *Discovery by Composition* [39], and *Fuzzy Model* [28] approaches come out on top of the approaches based on centralized respectively distributed architectures, as they additionally provide semantic service descriptions and reasoning capabilities. It is also noticeable that the *WS-Discovery* [51] approach is ranked by far the best syntactical and non-P2P-solution, as it does not need to overcome typical shortcomings that arise from a registry architecture. Thus, the autonomic capabilities of a service discovery approach cannot only be derived from the two dimensions spanning Figure 1 (represented by the criteria in the four leftmost columns of Table I), but a more fine-grained evaluation of all eight criteria is necessary. Nonetheless, Figure 1 represents a convenient way to categorize the screened service discovery approaches.

In this regard, the *I-Wanderer* [62] and the *pService* [55] approach, which only differ by one point in our evaluation, have been the two approaches providing the highest autonomic capabilities for service discovery. Especially the nature-inspired search mechanism of the *I-Wanderer* is noticeable as it also takes future requests into account.

V. CONCLUSIONS

In this paper we presented a survey and comparison of service discovery approaches in order to determine the

[2]We assigned a minus with 0 evaluation points in order to avoid negative evaluation points. Even though an approach may not meet any of the established criteria, the approach at least does not act contrary to it. So the minimal sum of evaluation points is 0, as it is the case for UDDI.

state of research towards an autonomic service discovery. We screened 42 approaches and categorized them into nine fields. Additionally, we evaluated these approaches based on eight prime criteria for autonomic service discovery, which we have defined before.

As one can see, research in recent years has focused especially on semantic service description and reasoning techniques as well as on scalability and robustness. There are already some approaches that support or fulfill most of the proposed criteria, which principally enables self-* properties. However, the fulfillment of even most of the criteria does not automatically entail an autonomic service discovery. Quite the opposite, they are a prerequisite for autonomic service discovery, but what is missing is an holistic approach focusing explicitly on providing autonomic self-* properties. Most of the approaches concentrate only on one or two autonomic capabilities, if any, but not on autonomy as a whole.

With respect to our evaluation model, the *I-Wanderer* and the *pService* approach had returned the best result. Please note that the evaluation model we used in this paper and the resulting evaluation points served only for the ranking between the surveyed approaches. If we had used different weighting factors or a more fine-granular evaluation model, the result would have looked somewhat different, of course. But from our point of view the chosen weighting factors represent the relevance of each criterion correctly. However, the evaluation points do not provide an indication on the current level of autonomic maturity.

A qualitative evaluation of the autonomic maturity would require a different and more complex evaluation model, which goes beyond the scope of this survey and is hence left as future research. Additionally, research in autonomic service management, business-driven automated composition, dynamic connectivity capabilities, etc. is required. But as we have seen, 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 and to reduce TCO.

REFERENCES

[1] M. P. Papazoglou, "The challenges of service evolution," in *CAiSE 2008*, ser. LNCS, Z. Bellahsene and M. Leonard, Eds., vol. 5074. Springer-Verlag Berlin Heidelberg, 2008, pp. 1–15.

[2] OASIS, *Web Services Business Process Execution Language*, 2007. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[3] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, and B. J. Kraemer, "Service-oriented computing research roadmap," Dagstuhl Seminar Proceedings 05462, April 2006, online available at http://drops.dagstuhl.de/opus/volltexte/2006/524 as of 2008-12-30.

[4] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, January 2003.

[5] M. Parashar and S. Hariri, Eds., *Autonomic Computing: Concepts, Infrastructure, and Applications*. CRC Press, 2006.

[6] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—degrees, models, and applications," *ACM Computing Surveys*, vol. 40, no. 3, pp. 1–28, 2008.

[7] IBM, "Autonomic computing whitepaper: An architectural blueprint for autonomic computing," June 2006.

[8] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic markup for web services," W3C, Tech. Rep., 2004.

[9] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, H. Lausen, E. Oren, A. Polleres, D. Roman, J. Scicluna, and M. Stollberg, "Web service modeling ontology (WSMO)," W3C, Tech. Rep., 2005.

[10] R. Studer, S. Grimm, and A. Abdecker, *Semantic Web Services; Concepts, Technologies and Applications*. Springer-Verlag Berlin Heidelberg, 2007.

[11] S. A. for WSDL Working Group, "Semantic annotations for WSDL and XML schema — usage guide," W3C, Tech. Rep., 2007.

[12] NEXOF-RA, "Requirements report," NESSI Deliverable D10.1, October 2008, online available: www.nexof-ra.eu as of 2008-12-30.

[13] J. Garofalakis, Y. Panagis, and E. Sakkopoulos, "Web service discovery mechanisms: looking for a needle in a haystack," in *Proceedings of International Workshop on Web Engineering*, 2004, pp. 9–13.

[14] D. Bachlechner, K. Siorpaes, D. Fensel, and I. Toma, "Web service discovery - a reality check," DERI, Tech. Rep., 2006.

[15] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Contemporary web service discovery mechanisms," *Journal of Web Engineering*, vol. 5, pp. 265–290, 2006.

[16] S. Hagemann, C. Letz, and G. Vossen, "Web service discovery - reality check 2.0," in *Proceedings of NWESP 2007*. IEEE Computer Society, 2007, pp. 113–118.

[17] I. Toma, K. Iqbal, D. Roman, T. Strang, D. Fensel, B. Sapkota, M. Moran, and J. M. Gomez, "Discovery in grid and web services environments: A survey and evaluation," *Multiagent Grid Syst.*, vol. 3, no. 3, pp. 341–352, 2007.

[18] Web Services Description Working Group, "Web services description requirements," W3C, Tech. Rep., 2002. [Online]. Available: http://www.w3.org/TR/ws-desc-reqs/

[19] Web Services Architecture Working Group, "Web services architecture requirements," W3C, Tech. Rep., February 2004. [Online]. Available: http://www.w3.org/TR/wsa-reqs/

[20] J. Almeida, V. Almeida, D. Ardagna, and M. Trubian, "Resource management in the autonomic service-oriented architecture," in *Proceedings of ICAC 2006*. IEEE Computer Society, 2006, pp. 84–92.

[21] D. Tosi, G. Denaro, and M. Pezze, "Towards autonomic service-oriented applications," *International Journal of Autonomic Computing*, vol. 1, no. 1, pp. 58–80, 2009.

[22] L. Liu and H. Schmeck, "A roadmap towards autonomic service-oriented architectures," *International Transactions on Systems Science and Applications*, vol. 2, no. 3, pp. 245–254, 2006.

[23] OASIS, "UDDI version 3.0.2 - UDDI spec technical committee draft," Organization for the Advancement of Structured Information Standards (OASIS), Tech. Rep., 2004.

[24] M. Jeckle and B. Zengler, "Active UDDI - an extension to UDDI for dynamic and fault-tolerant service invocation," in *Proceedings of WS-RSD 2002*, 2002.

[25] P. C. K. Hung and H. Li, "Web services discovery based on the trade-off between quality and cost of service: A token-based approach," *SIGecom Exch.*, vol. 4, no. 2, pp. 21–31, August 2003.

[26] S. Degwekar, H. Lam, and S. Y. W. Su, "Constraint-based brokering (CBB) for publishing and discovery of web services," in *LLC*, 2007.

[27] Y. Zhang and J. Ma, "Discovering web services based on probalistic latent factor model," in *APWeb/WAIM, LNCS 4505*, 2007, pp. 18–29.

[28] Q. Qiu, Q. Xiong, Y. Yang, and F. Luo, "Study on ontology-based web service discovery," in *IEEE Internationl Conference on Computer Supported Cooperative Work in Design*, vol. 11, 2007, pp. 641–645.

[29] K.-H. Lee and K.-C. Lee, "To maximize web service retrieval," in *Proceedings of ICCIT 2007*, 2007, pp. 2318 – 2325.

[30] A. Brogi and S. Corfini, "SAM: A semantic web service discovery system," in *KES/WIRN, LNAI 4694*, vol. 3, 2007, pp. 703–710.

[31] N. Kokash, W.-J. van den Heuvel, and V. D'Andrea, "Leveraging web service discovery with customizable hybrid matching," in *Proceedings of ICSOC 2006*, ser. LNCS 4294, 2006, pp. 522–528.

[32] N. Kokash, A. Birukou, and V. D'Andrea, "Web service discovery based on past user experience," in *BIS 2007, LNCS 4439*, 2007, pp. 95–107.

[33] R. Nayak and B. Lee, "Web service discovery with additional semantics and clustering," in *Proceedings of WI 2007*, 2007, pp. 555–558.

[34] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated discovery, interaction and composition of semantic web services," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, pp. 27–46, July 2003.

[35] N. Srinivasan, M. Paolucci, and K. Sycara, "An efficient algorithm for OWL-S based semantic search in UDDI," in *SWSWPC 2004, LNCS 3387*, 2005, pp. 96–110.

[36] J. Yu and G. Zhou, "Web service discovery and dynamic invocation based on UDDI/OWL-S," in *Proceedings of BPM 2005 Workshops*, ser. LNCS 3812, 2006, pp. 47–55.

[37] E. Tamani and P. Evripidou, "Combining pragmatics and intelligence in semantic web service discovery," in *Proceedings of OTM 2007*, ser. LNCS 4806, vol. 2, 2007, pp. 824–833.

[38] Z. Shen and J. Su, "Web service discovery based on behavior signatures," in *Proceedings of SCC 2005*, 2005.

[39] L. Aversano, G. Canfora, and A. Ciampi, "An algorithm for web service discovery through their composition," in *ICWS 2004*, 2004.

[40] R. Li, Z. Zhang, W. Song, F. Ke, and Z. Lu, "Service publishing and discovering model in a web services oriented peer-to-peer system," in *Proceedings of ICWE 2005*, ser. LNCS 3579, 2005, pp. 597–599.

[41] A. Kassim, B. Esfandiari, S. Majumdar, and L. Serghi, "A flexible hybrid architecture for management of distributed web service registries," in *Communication Networks and Services Research (CNSR)*, vol. 5, 2007.

[42] Z. Du, J. Huai, and Y. Liu, "Ad-UDDI: An active and distributed service registry," in *Proceedings of TES 2005*, ser. LNCS 3811, 2006, pp. 58–71.

[43] K. Sivashanmugam, K. Verma, and A. Sheth, "Discovery of web services in a federated registry environment," in *Proceedings of ICWS 2004*, 2004, pp. 270–278.

[44] J. Hu, C. Guo, Y. Jia, and P. Zou, "Stratus: A distributed web service discovery infrastructure based on double-overlay network," in *APWeb 2005, LNCS 3399*, 2005, pp. 1027–1032.

[45] M. Pantazoglou, A. Tsalgatidou, and G. Athanaspoulos, "Discovering web services and JXTA peer-to-peer services in a unified manner," in *Proceedings of ICSOC 2006*, ser. LNCS 4294, 2006, pp. 104–115.

[46] S. Willmott, H. Ronsdorf, and K. H. Krempels, "Publish and search versus registries for semantic web service discovery," in *Proceedings of WI 2005*, 2005.

[47] E. Al-Masri and Q. H. Mahmoud, "WSCE: A crawler engine for large-scale discovery of web services," in *Proceedings of ICWS 2007*, 2007.

[48] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Millder, "METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of web services," *Information Technology and Management*, vol. 6, pp. 17–39, 2005.

[49] D. Celeik and A. Elci, "Discovery and scoring of semantic web services based on client requirement(s) through a semantic search agent," in *Proceedings of COMPSAC 2006*, vol. 30, 2006.

[50] S. Li, C. Xu, Z. Wu, Y. Pan, and X. Li, "ABSDM: Agent based service discovery mechanism in internet," in *Proceedings of ICCS 2004*, ser. LNCS 3036, 2004, pp. 441–444.

[51] J. Beatty, G. Kakivaya, D. Kemp, T. Kuehnel, B. Lovering, B. Roe, C. St.John, J. Schlimmer, G. Simonnet, D. Walter, J. Weast, Y. Yarmosh, and P. Yendluri, "Web services dynamic discovery (WS-Discovery)," Microsoft Corporation Inc., Tech. Rep., April 2005.

[52] Y. Li, F. Zou, F. Ma, and M. Li, "Build a distributed repository for web service discovery based on peer-to-peer network," in *Proceedings of NPC2 004*, ser. LNCS 3222, 2004, pp. 175–182.

[53] C. Schmidt and M. Parashar, "A Peer-to-Peer approach to web service discovery," *World Wide Web: Internet and Web Information Systems*, vol. 7, pp. 211–229, 2004.

[54] F. Yan and S. Zhan, "A Peer-to-Peer approach with semantic locality to service," in *Proceedings of GCC 2004*, ser. LNCS 3251, 2004, pp. 831–834.

[55] W. Lv and J. Yu, "pService: Peer-to-Peer based web services discovery and matching," in *Proceedings of ICSNC 2007*, vol. 2, 2007.

[56] O. D. Sahin, C. E. Gerede, D. Agrawal, A. E. Abbadi, O. Ibarra, and J. Su, "Spider: P2P-based web service discovery," in *Proceedings of ICSOC 2005*, ser. LNCS 3826, 2005, pp. 157–169.

[57] R. Romeikat and B. Bauer, "Towards semantically-enchanced distributed service discovery," in *Proceedings of ICIW 2007*. Washington, DC, USA: IEEE Computer Society, 2007.

[58] Y. Li, S. Su, and F. Yang, "A Peer-to-Peer approach to semantic web service discovery," in *Proceedings of ICCS 2006*, ser. LNCS 3994, vol. 4, 2006, pp. 73–80.

[59] L.-H. Vu, M. Hauswirth, and K. Aberer, "Towards P2P-based semantic web service discovery with QoS support," in *Proceedings of BPM 2005 Workshops*, ser. LNCS 3812, 2006, pp. 18–31.

[60] S. Yu, J. Liu, and J. Le, "Intelligent web service discovery on large distributed system," in *Proceedings of IDEAL 2004*, ser. LNCS 3177, 2004, pp. 166–172.

[61] B. Sapkota, L. Vasiliu, I. Toma, D. Roman, and C. Bussler, "Peer-to-Peer technology usage in web service discovery and matchmaking," in *Proceedings of WISE 2005*, ser. LNCS 3806, 2005, pp. 418–425.

[62] Z. Changyou, Z. Dongfeng, Z. Yu, and Y. Minghua, "A web service discovery machanism based on immune communication," in *Proceedings of ICCIT 2007*, 2007, pp. 456–461.

[63] B. Sapkota, D. Roman, and D. Fensel, "Distributed web service discovery architecture," in *Proceedings of ICIW 2006*, 2006.

[64] E. Tamani and P. Evripidou, "A pragmatic methology to web service discovery," in *Proceedings of ICWS 207*, 2007.

[65] J. Aspnes and G. Shah, "Skip graphs," *ACM Transactions on Algorithms*, vol. 3, no. 4, 2007.

[66] W3C, "OWL web ontology language," W3C, Tech. Rep., February 2004. [Online]. Available: http://www.w3.org/TR/owl-features/

[67] S. Yu, J. Liu, and J. Le, "DHT facilitated web service discovery incorporating semantic annotation," in *Proceedings of DS 2004*, ser. LNAI 3245, 2004, pp. 363–370.