Chapter 2
Service Advertisement and Discovery

Shanshan Yang and Mike Joy

Abstract Service Advertisement and Discovery is a fundamental process in service oriented computing, which also provides a precondition for other processes such as service selection and composition (these will be covered in detail in later chapters). This chapter provides an introductory overview of the concepts, standards and current developments related to Service Advertisement and Discovery, summarised from the perspectives of system architecture, data structures, system requirements and Web Services. The incorporation of agent-based technology into Service Advertisement and Discovery is covered, and the chapter concludes with a discussion of future research challenges in this area.

2.1 Introduction to Service Advertisement and Discovery

A service is "a software system designed to support interoperable machine-to-machine interaction over a network" [73]. The purpose of a service is to "provide some functionality on behalf of its owner—a person or organisation, such as a business or an individual" [73]. The service provider is the entity that provides a particular service, and the service requester (or consumer) is the entity that wishes to make use of a provider's service. The goal of finding an appropriate service (the process of performing discovery) requires the service requester and provider to "become known to each other", and it is necessary to ensure that service descriptions are published somewhere (in a registry) before that information is available to others. This task is performed by another entity—a service broker [49].

Shanshan Yang
Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
e-mail: Shanshan.Yang@dcs.warwick.ac.uk

Mike Joy
Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
e-mail: M.S.Joy@warwick.ac.uk

N. Griffiths, K.-M. Chao (eds.), *Agent-Based Service-Oriented Computing*,
Advanced Information and Knowledge Processing,
DOI 10.1007/978-1-84996-041-0_2,

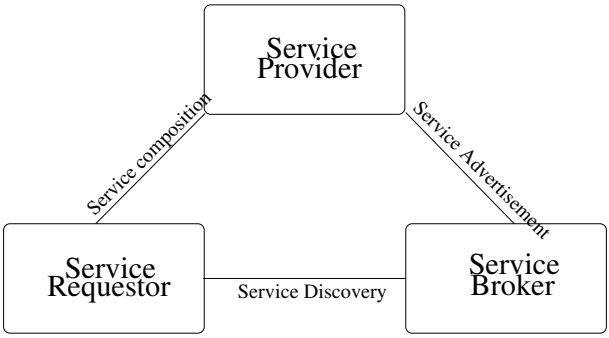

Fig. 2.1 Service oriented architecture

Most authors consider that a basic service oriented architecture consists of three different entities: services *providers* and *requesters* and a service *broker* (registry) [13,31,48,53,67], and the relationships between these entities are illustrated in Figure 2.1. Dustdar and Treiber [31] identify the role of the service provider as one of providing descriptions, and that of the broker as publishing them. The requester contacts a broker in order to locate a suitable service to fulfil a given task, and when an appropriate service has been identified, the broker will additionally provide information about how that service can be invoked. The broker uses a *service registry* (repository) to store the necessary information about services, allowing both user searches and the publication of service descriptions. Searching for and locating services, in order to identify matches between service requesters and providers, is regarded as a key issue, and service *brokers* (or *registries*) play a major role in this task. Thus the role of the service broker and its registry is central to the current model of service oriented computing [28].

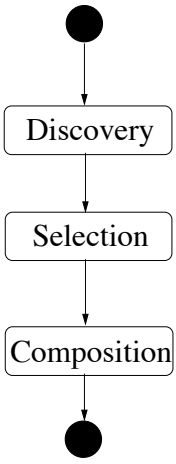

Fig. 2.2 Service execution workflow

Singh and Huhns' [75] summary (Figure 2.2) of the services execution workflow identifies the activity of service discovery as the first step, followed by the other processes including service selection and composition. Some or all of these steps can be performed offline or at runtime. Service discovery deals with finding services that meet a specified description, whereas selection deals with "choosing appropriate services from among those that are discovered for the given description" [80]. Service composition deals with combining small services into larger ones to meet a specified goal [8, 10]. As Wu [93] remarks, "As an essential SOA activity, [service discovery] paves the way for conducting further important SOA activities such as service sharing, reusing and composing in a dynamically changing environment." In this chapter, we consider the first phase—service advertisement and discovery—since in order to discover the services needed by the requester, it is necessary to specify and publish the services effectively first, which means that service advertisement provides a essential precondition for service discovery. We focus in particular on web services, that use web technologies to implement a service-oriented architecture.

There are no generally accepted formal definitions of either Service Advertisement or Discovery (or synonymous phrases), and different approaches to describing them have been employed. For example, Yu et al. [95] define *service publication* as "to make the service description available in the registry so that the service client can find it" and *service lookup* as to "query the registry for a certain type of service and then retrieve the service description." The service description is identified as containing both syntactic information (such as the data formats and protocols used by the services) and semantic information (relating to the domain in which the services are employed together with generic issues such as service functionality and quality of service). Vitvar [87] describes *discovery* as "tasks for identifying and locating services which can achieve a requester's goal", whereas Singh and Huhns [75] view discovery as "the act of locating a machine-processable description of a web service that may have been previously known and that meets certain functional criteria".

Advertising service information is normally considered at the same time as service discovery. Current research in service advertisement focuses on how web services are described, or specified, or published from a technical view, such as what standards people should adopt, or what architecture could be used effectively.

A number of researchers [12, 55, 58, 69] also suggest that agent technologies can be fitted into service oriented architectures, to improve the effectiveness of the service advertisement and discovery process. Agents can be members of multi-agent environments acting not only as brokers, but also as service providers and consumers. Details of agent based approaches will be covered after we have introduced the fundamental technologies and current developments of service advertisement and discovery.

2.2 Basic Technologies

It is commonly agreed that three basic standards are currently in use for web service advertisement and discovery [17, 20, 31, 37, 48], each with its own specific role.

- SOAP: Communication—how services can be used
- WSDL: Description—how services can be published
- UDDI: Discovery—how services can be discovered

Fundamental to the efficacy of these standards is the use of a common communications language [75], and XML is used by each. The communications protocol is defined by SOAP, and WSDL includes support for passing information about functions supported by services, including their names, parameters and result types. UDDI specifies the contents of the registry, enabling users to search for services and find sufficient information for their deployment—an essential prerequisite if web services are to be meaningful. These standards have been developed by organisations including the World Wide Web Consortium (W3C) [73], OASIS [61] and the Open Group [64] since 2000 with the latest versions published in 2007.

2.2.1 SOAP

In the context of web services, SOAP (Simple Object Access Protocol) is regarded as the standard message protocol for exchanging XML data over the Internet. SOAP is a stateless paradigm which enables complex interactions between services through request/response exchanges and other unidirectional messages. However, SOAP lacks support for the transmission of semantic data, such as routing and firewall traversal [25].

A SOAP message is essentially an XML element with two XML child elements, a head and a body. These contain descriptions of the message content and how to process it, encoding rules (for application-specific data types), and the representations of remote procedure calls and responses [86]. This information is then wrapped into an envelope, and is bound to a transport protocol for the purposes of the actual information exchange [78]. The following is an example of a SOAP message for invoking a web service for getting a stock price, which is cited from the W3C School website [88]:

```
<?xml version="1.0"?>
  <soap:Envelope
    xmlns:soap=
      "http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle=
      "http://www.w3.org/2001/12/soap-encoding">
       <soap:Body xmlns:m=
      "http://www.example.org/stock">
          <m:GetStockPrice>
            <m:StockName>IBM</m:StockName>
          </m:GetStockPrice>
       </soap:Body>
  </soap:Envelope>
```

2.2.2 WSDL

WSDL (Web Service Description Language) formally provides a model for describing interfaces for web services [75, 86, 89]. A WSDL description specifies the location of the service, the operations for invoking and consuming the web service, and supports binding for defining message formats and protocol details. The following is a typical structure of a WSDL document, which is cited from W3C School [88]:

```
<definitions>
  <types>definition of types</types>
  <message>definition of a message</message>
  <portType>
    <operation>definition of a operation</operation>
  </portType>
  <binding>definition of a binding</binding>
  <service>
    <port>definition of a port</port>
  </service>
 </definitions>
```

A typical WSDL document contains the following elements. The type element specifies the complex data types for a message, which describe the data being communicated between the web service and the requester. A set of messages and their directions (input or output) form the operations the service exposes. A set of operations then forms a port type, for each of which the concrete protocol and data format specifications are referred to as a binding. The association of a network address with a binding defines a port, and a collection of ports defines a service. In a single WSDL file multiple services can be described [92].

WSDL defines services as "collections of network endpoints or ports". The abstract definitions of messages and the endpoints/ports are then separated from their concrete implementation, such as protocols and data formats, allowing for reuse of those definitions [73].

2.2.3 UDDI

UDDI (Universal Description, Discovery and Integration) is a registry of web service descriptions, allowing users (such as businesses) to publish descriptions of themselves and their services (together with technical information about service interfaces), and clients (such as customers) to identify appropriate service descriptions and create bindings to them (using SOAP) [89, 96]. Wang [89] summarises a UDDI registry as being "similar to a CORBA trader and can be considered as a DNS service for business applications". It serves as a generic data model for providing detailed web service specifications including business entities, technical access information, natural language descriptions, keyword-based classification scheme and relevant technical specifications [25].

The initial idea of maintaining a central registry for publicly available web services by large vendors, such as IBM or Microsoft, has been abandoned because a

single repository can not meet all the needs for different specific SOA systems [96]. Version 3 of the UDDI specification is over 400 pages long and contains over 300 function calls. This complexity (for end users) has led to the closure of the public UDDI Business registry and has hindered its widespread adoption, and has led to speculation that future registries will be private [93]. As Chappell [21] remarks: "the public registry UDDI is too complex for end users since UDDI specification is more driven by its primary members than feedback from the real world end users". However, Baresi and Miraz [6] also suggest that the central registry will continue to be important since not all companies will have the facilities for servicing requests locally, and Wu [93] considers that "most private registries would focus on a specific, closed domain".

Both private and public registries follow the two principals of UDDI specifications relating to the composition, structure and operation of a registry—the information provided about each service (including its encoding) and an API specifying how to update the registry and how to make queries. The information encoded by UDDI is of three possible types—*white pages* (names, contact information), *yellow pages* (categories of information based on service types) and *green pages* (technical data) [25, 92].

A recent development is UDDIe, an extension to UDDIe which incorporates service leasing and replication. UDDIe includes the ability to search for services based on *blue pages* (user defined properties associated with a service). Support for service leases, by which a service is restricted to storage in the registry for a limited period of time, enhances the dynamic capabilities of the registry [74].

2.3 Web Service Registry Architectures

This section covers how a web service registry supports and implements service advertisement and discovery. Currently, a number of architectures for web service advertisement and discovery have been developed, influenced by the architectures of different service oriented systems, which can be viewed from both structural and functional perspectives. However, the technology is still emerging, and components still being developed include quality of service descriptions and interaction models.

The main structure difference between different architectures is about how the registries are distributed, and three types of architectures have been proposed, namely *centralised*, *decentralised* and *hybrid*.

2.3.1 Centralised Registries

In a centralised registry (such as UDDI), all web service registry entries are contained within a single "well known" central entity used by each web service provider, similar to a traditional client-server approach [22, 31, 39].

However, there are limitations on this type of architecture. First of all, a centralised registry is not scalable—it can only support small scale systems. Simple easy-to-use technologies such as UPnP, SLP and Jini [39] and the DS-1, Hawkeye and RGMA approaches for grid systems [47] have been reported as examples of small-scale centralised approaches which do not scale well. The second limitation is that it is unsuitable for dynamic environments, and Chamri-Doundane et al. [39] point out that frequent changes affect the system behaviour and its efficiency. The third is that a centralised registry does not handle fault-tolerance well, there being the possibility of a single point of failure [77].

Despite these limitations, several centralised approaches exist, and are effectively applied in situations where scalability, dynamism and fault tolerance are not paramount. Below, we introduce a selection of example centralised systems.

The ebXML (electronic business XML) standard defines a framework within which businesses can co-operate. It is similar to UDDI, but is broader in scope, being able to store arbitrary data and specifying interrelated components for business activities. Two interfaces are specified, LifeCycleManager (which handles the submission of new objects to the registry, the classification of existing objects and the removal of obsolete objects) and QueryManager (which handles the processing of client requests to locate web services using either SQL queries or filters) [33].

SLP (Service Location Protocol) is used by devices (such as printers) on a (local) network to announce services. The centralised service repository is known as a directory agent (DA), and service agents (SAs) and user agents (UAs) use the DA to register and locate services respectively [42, 70].

Sun Microsystems' Jini (now being developed by Apache as Apache River) is a networked technology which allows Java software to be accessed using a centralised service architecture. In addition to service information, lookup services store proxies which enable code to be executed either locally or remotely, thus supporting dynamic use of drivers at runtime [4].

The Salutation Consortium has created an open standard which is both a service discovery and a session management protocol. The architecture is principally targeted at device connectivity on local networks, and relies on devices communicating with a centralised repository (the Salutation Manager) in a fully distributed manner and using a message-passing paradigm. In a low-bandwidth wireless network without fixed IP addresses, the large volume of control traffic generated is problematic. The Consortium was disbanded in 2005 [71].

R-GMA (Relational Grid Monitoring Architecture) is based on a relational data model, and uses a relational database to implement the centralised GMA registry [47].

2.3.2 Decentralised Registries

As opposed to centralised systems, the localisation of services in a decentralised registry is completely distributed and diffused. This type of registry architecture

has been applied to different types of modern environments, including peer-to-peer networks, mobile-ad-hoc-networks and Grids [39].

2.3.2.1 Peer-to-Peer Networks

In a Peer-to-Peer (P2P) network, each node is (in some sense) equivalent to every other node. Applications rely on ad-hoc connections between nodes (peers) without a centralised server, and the advantage lies in scalability, robustness, and ease of deployment and maintenance. An example set of protocols that supports a P2P architecture is Sun Microsystem's JXTA, which includes features such as service advertisement and messaging in addition to basic peer management [55]. Each service provider has a local registry and performs the roles both of service provider and of registry, but only for the period of time that the provider is connected to the P2P network, thus limiting the lifespan of each registry entry and enabling a dynamic registry structure with resource localisation and sharing [31, 39, 66].

A number of P2P approaches to web services exist. Schmidt and Parashar's architecture [72] uses distributed hash tables and an indexing system based on the CHORD data lookup protocol. Web services are indexed using descriptive keywords, and a dimension reducing indexing scheme is used. Dustdar and Treiber's [32] VISR (View based Integration of Web Service Registries) is a peer to peer architecture which combines multiple web service registries with transient web service providers in a seamless integrated system. "Views" serve as an abstraction layer which uses web service profiles as a global data model, and are supported by a simple grammar (View Description Language). The web service profiles allow extra information to supplement the registry entries without affecting the original entries themselves [32].

2.3.2.2 Mobile Ad-Hoc Networks

In a Mobile Ad-Hoc Network (MANET), cooperating autonomous mobile devices acting as router nodes form a dynamic network infrastructure. Wireless technologies are usually employed, and the use of standard protocols and interfaces ensures that the devices communicate effectively so that advertising and discovery is possible [35, 54, 59].

Tyan and Mahmoud [83] propose grouping mobile nodes into clusters, with one device in each cluster acting as a gateway for routing purposes, using a location-aware network layer routing protocol. The gateways also improve service discovery performance by acting as directories, and a context-aware agent-based service selection mechanism is included. This solution addresses issues of scalability and context-awareness since complex graph algorithms are no longer needed to maintain the clusters and support management of the network topology.

Carlos et al. [19] have developed a component-based service discovery framework, which can be used in both fixed and ad hoc networks, and supports adaptive

service discovery middleware. This approach enhances framework configurability and minimises resource usage.

Talwar et al. [82] have developed a novel resource and service discovery mechanism for MANETs using RIMAs (Routing Intelligent Mobile Agents), which collect and index information on service availabilities as well as network resource and routing data. Each RIMA is associated with a node in the network, and each mobile node is *close* to one or more RIMA nodes. Discovery agents are used by service requesters to identify resources by using the indices contained in the RIMA nodes. The mechanism has been tested by simulating MANETs with up to 800 nodes.

2.3.2.3 Grid Computing

Issarny et al. [50] characterise Grid computing as addressing "the creation of distributed communities that share resources such as storage space, sensors, software application and data, by means of a persistent, standards-based service infrastructure". Currently used principally by the scientific community as a high-performance computing infrastructure, a Grid can support more general large-scale applications requiring substantial data processing and computation. Grid computing often requires secure resource sharing amongst multiple institutions, and this model does not fit in well with the current Internet infrastructure [55].

Globus Toolkit (GT) is an open source set of libraries and programs that has been developed over the last few years by the Globus Alliance consortium to support the building of distributed system services and applications. It addresses the fundamental issues such as resource discovery, resource access, resource management, data movement and security [36]. The architecture contains three sets of components: a set of implementation services, a set of service containers, and a set of client libraries [40].

GISs (Grid Information Services) form a key component in many Grid architectures, and S-Club is a mechanism which supports efficient service discovery on a GIS mesh network. Using the existing CROWN (China Research and Development environment Over Wide-area Network) GIS network, S-Club forms an overlay in which services are clustered as "clubs", each club providing services of a given type. A given service may belong to multiple clubs, and a service requester will initially use the S-Club overlay to identify providers by searching appropriate clubs. The overlay is constructed dynamically, and a minimum-spanning tree topology is used in order to ensure that messages are transmitted efficiently. Experimental results show that the S-Club approach improves response times for searches as well as reducing traffic overhead [47].

Bell et al. [9] propose an extension of the Grid framework to include semantic services in a real-world commercial context—a "Business Grid". An upper service ontology is used to provide the semantic context, and web services taken from investment banks have been used to validate the approach.

Yu et al. [95] propose the Grid Market Directory (GMD), a registry which manages the provision of services efficiently using a pricing mechanism. It is designed

to be applied to market-oriented Grids to "support an infrastructure that enables the creation of a marketplace for meeting of providers and consumers". GMD contains two components: the portal manager and the query web service. The portal manager covers the tasks of "provider registration, service publication and management, and service browsing", and the query web service allows clients such as resource brokers to query the GMD and obtain resource information to identify those that satisfy the user's QoS requirements [95].

2.3.3 Hybrid Registries

In addition to pure centralised and decentralised architectures, some hybrid (federated) systems have been proposed, in which registry information is distributed amongst multiple entities in a peer-to-peer manner, but access to the registry information is through dedicated "super peer" nodes (peer registries). Such systems appear to users as centralised, since the use of peers is transparent and the user is unaware of the distributed implementation. This approach allows for registries to specialise in particular types of web service, although this benefit must be weighed against the increased communication overhead [31, 94].

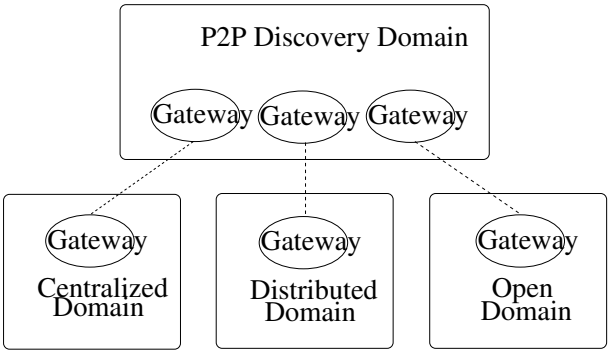

Fig. 2.3 Ghamri-Doudane's service discovery architecture

Ghamri-Doudane et al. [39] present a purely unstructured service discovery architecture containing components which include centralised, distributed and P2P discovery domains. The intention is to integrate all existing service discovery protocols but with a specific service gateway for each technology, as shown in Figure 2.3 [39].

Verma et al. [86] present a scalable, high performance environment for web service publication and discovery among multiple registries. Using an ontology-based approach, registries are organised into domains, so that web services can be classified using those domains. A semantic approach to the publication and discovery of web services is used, and it is claimed that this is appropriate for systems containing

large numbers of registries. METEOR-S is an architecture which supports this environment and an implementation has been tested [86].

Papazoglou and Heuvel [67] (Figure 2.4) introduce the concept of service-syndications, where related businesses form groups based on common interests, and each group has its own UDDI peer registry.

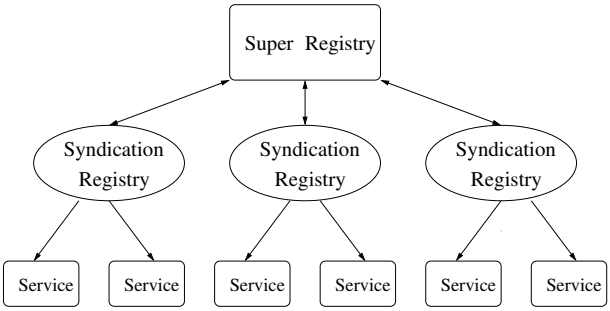

Fig. 2.4 Papazoglou and Heuvel's service discovery architecture

Caron's unstructured peer-to-peer network architecture extends traditional Network-Enabled Server (NES) by enabling tree-based service discovery which takes account of the underlying network topology. The benefits claimed for this approach include improved fault-tolerance and efficiency on wide-area networks [19].

2.4 Data Structures

Data on web services can include complex information, such as collaboration protocols and structured ontological information, in addition to more basic data such as the service name and information about the service provider.

Perhaps the most commonly used data model is UDDI, which is hierarchically structured and contains five data types defined using an XML schema [74]. The tModel data type is used to represent information about a given service, including a technical description of what the service does and how it does it, whilst the other data types contain information about the service providers, the range of services offered by each, and descriptions of the services on offer. Each entity in the model is allocated a unique identifier (UUID), and classified according to a published taxonomy [89]. In versions 1 and 2 of UDDI, the following classification schemes were used [95]:

- the North American Industry Classification System (NAICS),
- the Universal Standard Products and services Code System (UNSPSC), and
- the ISO Geographic taxonomy (ISO 3166).

The use of a taxonomy is seen as important in that it offers a structured framework which facilitates searching for services. The UDDI model is composed of four identity types, as the following diagram (Figure 2.5) illustrates [25]:

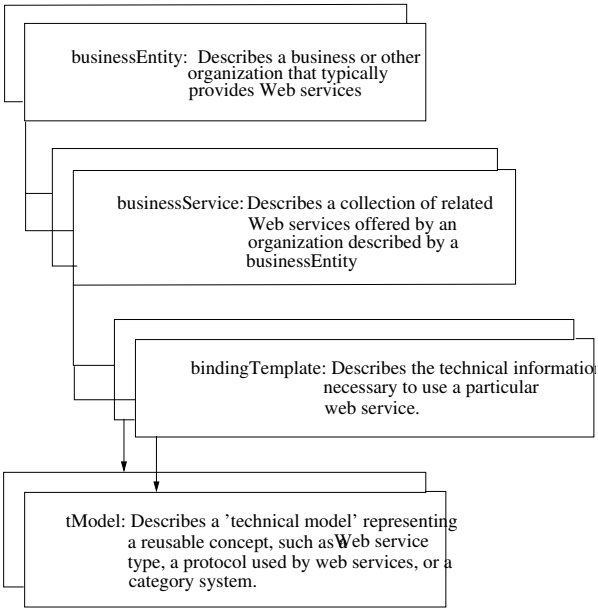

Fig. 2.5 UDDI data model

The ebXML data model is broader in scope than UDDI, and in addition to services it supports further data related to e-business. Data in an ebXML registry takes the form of metadata about objects in the registry (including, but not restricted to, web services) [33].

The Web Service Discovery Architecture (WSDA) is a modular architecture which defines services, interfaces, operations and protocol bindings, based on industry standards. WSDA has the advantage of flexibility, since the modular components can be customised easily and adapted to support a range of behaviours [46].

The Web Services Inspection Language (WSIL) is a distributed approach to the provision of data for service discovery, in contrast to the centralised model adopted by UDDI. Each web service produces a WSIL XML file containing the necessary data (which is similar to the data stored within UDDI), and that file is made accessible by (for example) publishing it using simple naming conventions on an advertised web site [91].

2.5 System Requirements

In this section, registry architectures are viewed from the angle of their common functions and requirements from the users' perspective. Interoperability and matchmaking are the only two core requirements for service advertisement and discovery systems, and the others are optional. As Doulkerdis [29] remarks: "Existing service discovery mechanisms usually focus on exact or semantic matching of static attributes". Although each system is able to meet more that one requirement, no single system meets them all. In the remainder of the section, the requirements are defined, and systems that meet each requirement are discussed.

Interoperability means "the ability to exchange and use information between different heterogeneous web service registry environments" [31]. O'Brien et al. [62] mention that "increased interoperability is the most prominent benefit of SOA", and Yu et al. [95] reinforce this opinion, arguing that, "[i]nteroperability is the core functionality that web services endeavour to achieve".

Matchmaking is "a mechanism by which service requesters can find potential web services (providers) that have capabilities for meeting their specific requirements" [93], and is explicitly supported by models such as Garg's System Template approach [38] which seeks to match instances of related services into groups.

Scalability "defines how well a web service registry responds to increasing load" [31]. An example of an architecture for which scalability has been a major motivation is AtomServ [93], which uses standard web feed technologies (Atom and RSS) accessible through ubiquitous application interfaces such as browsers. The use of UPnP is another means of supporting scalability, such as has been used in CSSD [7].

Fault tolerance is "the ability of a web service registry to continue normal operation despite the presence of hardware or software faults" [31]. Service Address Routing (SAR), which supports a "location-independent" distribution of services across a network, is an example of a fault-tolerant mechanism which has been applied successfully both to tightly-coupled networks and to a loosely-coupled Grid [72].

Reliability means "the degree to which a web service registry is capable of maintaining the service at a given service quality" [31]. This has been addressed, for example, by an extension to Web Service Repository Builder (WSRB) architecture in which a Web service Relevancy ranking Function (WsRF) is used, to modify the service discovery process. WsRF uses QoS metrics, such as reputation and compliance, together with relevancy rankings based on clients' preferences, and the technique has been validated experimentally [1].

Security means "where necessary, communications are both encrypted and authenticated" [26]. As part of the Ninja project, the Secure Service Discovery Service (SSDS) supports a high level of security. SSDS provides clients with directory-style access to services, with encrypted communication facilitated by per-session keys. Individual components are allocated certificates which can be signed by clients and by service providers, and the model then allows a client to identify services they trust based on the levels of trust the client has in the signatories to the services'

certificates. Furthermore, SSDS supports signed messages (capabilities) which identify that a user has access to a set of services, thus restricting clients to those services which the system has identified as appropriate and allowed [24, 44, 76].

Context awareness is the ability "to seamlessly adapt behaviour according to the context within which the systems executes". This involves sensing the environment and adapting the behaviour of an application according to both the users' profiles and the available resources [50, 79].

CSSD is an example of a system which uses context (dynamically changing information about the services provided and the user, and the user's environment as provided by an external system) to inform the service discovery algorithm [7]. Another initiative has been the development of MobiShare—a cellular mobile resource architecture—to include Context-Aware Service Directories (CASDs) within the architecture's Cell Administration Servers (CASs) [30].

Mobility refers to the support offered by an architecture for mobile (wireless) devices. For example, the Siena architecture is implemented as an overlay on a GPRS mobile network, uses a distributed publish/subscribe paradigm, and supports a variety of Internet applications and services [18, 23, 29].

It is perhaps useful at this point to note that all systems surveyed here address the issues on interoperability and of matchmaking, most of those systems are also scalable, and roughly half consider fault-tolerance to be an important feature. The other issues are only addressed by few of the systems.

2.6 Advertisement and Discovery Services

A variety of common technologies are currently used by discovery services [85], although Hoffert et al. [45] note that "while discovery services are fairly mature and broadly applicable to today's systems much R&D remains to support emerging systems of ultra-large scale effectively, such as the Global Information Grid". This section discusses those technologies which can be considered mature.

The *Common Object Request Broker Architecture* (CORBA) is a technology which allows objects, possibly created using different languages and implemented on different platforms, to communicate across a network. The CORBA Naming Service is a database containing bindings of names and associated objects, which allows distributed objects to be located by name and accessed by clients—a "white pages" technology. The CORBA Trading Service, in contrast, allows objects to be located based on a requirements description rather than by name—a "yellow pages" technology [63].

The *Data Distribution Service* (DDS) for Real-Time Systems has recently been approved as an OMG standard. In contrast to the client/server approach, DDS adopts a data-centric publish/subscribe (DCPS) model, grouping data into "topics" (sets of related data-objects with a common data type), and allows the user to specify Quality of Service parameters [27].

The *Jini Lookup Service* (JLS) uses Java RMI (Remote Method Invocation) to allow Java clients to discover services (Java objects or proxies) by specifying an interface. This approach benefits from optimisation (such as bytecode and object caching) available through RMI, but Hoffert et al. [45] note that "it can also have undesirable side effects, such as increased latency and jitter when first transferring the object". Although Jini may superficially appear to be a Java version of CORBA, the differences in approach and implementation are substantial [4, 51].

Low-level protocols are used by networks in support of service discovery. For example, *Simple Service Discovery Protocol* (SSDP) is used by UPnP to allow services (such as external devices and resources) to be identified by clients which use those services. The Bluetooth Service Discovery Protocol (SDP) uses the Logical Link Control and Adaptation Protocol (L2CAP) layer to initialise connections for devices via the Logical Link Control and Adaptation Protocol (L2CAP) layer within the short-range wireless network used by Bluetooth. Service Location Protocol (SLP) is a packet-oriented protocol which allows devices to locate services across a LAN, without prior configuration, and is scalable to large networks. Three agents are employed—a user agent which seeks appropriate services, a service agent which provides information about available services, and an optional directory agent which enhances the performance of the service agents by providing a central repository which stores the locations of the services [45].

JXTA is a collection of open-source XML-based protocols which supports a peer-to-peer communication between networked devices and services via a network overlay. Low-bandwidth devices (edge peers), which may only be connected temporarily, are treated differently to super peers, which co-ordinate other peers and facilitate communication through firewalls and between subnets [52].

UDDI supports service discovery by registering service descriptions in the *UDDI Business Registry* (UBR), which users can query to find either a given provider or the category of service [84].

Peer-to-peer (P2P) architectures—perhaps most commonly used for file-sharing and MP3 downloads rather than for more general resources—can also support distributed service provision. *Gnutella* (and its fork Gnutella2) is a P2P resource sharing network which—like products such as *Bittorrent*—is typically used to exchange files, and uses a network overlay scheme together with a number of optimisation techniques. These include QRP (Query Routing Protocol), which uses a hash table to prevent queries being forwarded to inappropriate network nodes, and DQ (Dynamic Querying) which caps the number of results returned by a search and so reduces network traffic [41]. Napster is an architecture which, unlike Gnutella, uses a centralised registry in addition to using network nodes as resource servers, so that the registry can direct traffic to an appropriate server [57].

2.7 Agents in Service Advertisement and Discovery

The technologies and approaches discussed in this chapter present service advertisement and discovery as typically decentralised and asynchronous activities. The software components which implement and support them have attributes—such as autonomy and adaptivity—which are characteristics of an agent-based approach, suggesting that the incorporation of agent technologies into service oriented architectures may improve the effectiveness of the process [15,65]. Singh [75] notes that:

> "Typical agent architectures have many of the same features as service oriented architectures. Agent architectures provide service directories, where agents advertise their distinct functionalities and where other agents search to locate the agents in order to request those functionalities."

Luck [55] also remarks:

> "It is natural to view large systems in terms of the services they offer, and consequently in terms of the entities or agents providing or consuming services. In this view agents act on behalf of service owners, managing across to services, and ensuring that contracts are fulfilled. They also act on behalf of service consumers, locating services, agreeing contracts, and receiving and presenting results."

2.7.1 Agents in Service Oriented Computing

In service oriented systems, an agent can assume a role such as that of service provider, consumer (user) or broker. The tasks a broker agent would be responsible for might include:

- identifying and locating appropriate service agents;
- implementing directory services;
- managing namespace services;
- storing, forwarding and delivering messages;
- managing communication between the other agents, databases and application programs.

Singh and Huhns [75] advocate a generic agent-based service-oriented system architecture containing agent types, as illustrated in Figure 2.6. Of these, directory and broker agents and resource agents perform the tasks of service advertisement and discovery. They claim that "[b]rokers simplify the configuration of multi-agent systems", and note that a broker's knowledge about other agents within a system allows it to identify and negotiate with potential agents which may be able to offer a desired service. Resource agents provide access to information based services, and user agents can behave as "an intermediary between users and information systems" [12, 58, 69, 75].

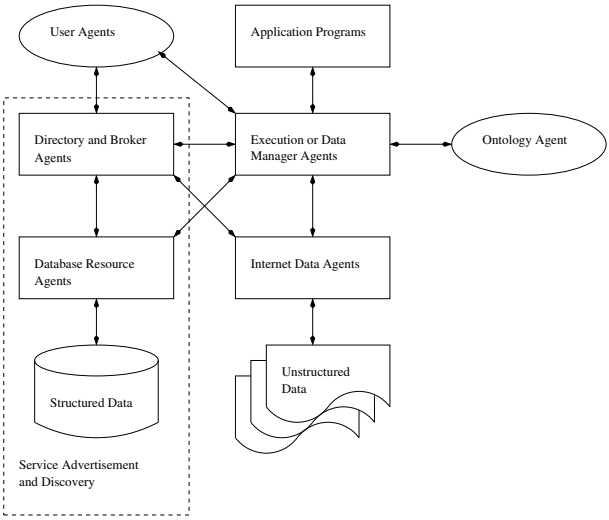

Fig. 2.6 Singh and Huhns' agent based service oriented architecture

2.7.2 Development of Agents in Service Advertisement and Discovery

The model offered by Singh and Huhns above is generic, and serves as a useful starting point for exploring other approaches.

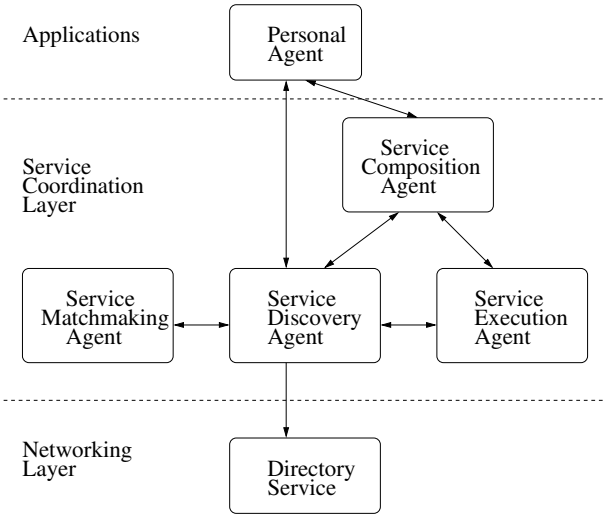

Fig. 2.7 CASCOM: Agent based service oriented architecture

The CASCOM project [14] focuses on semantic service coordination in intelligent agent-based peer-to-peer networks (IP2P). An abstract architecture has been developed (see Figure 2.7) which within the central Service Coordination Layer uses Service Discovery Agents (SDAs) and Service Matchmaking Agents (SMAs) to handle the semantic aspects of service discovery, together with Service Composition Agents (SCAs) and Service Execution Agents (SEAs) for coordination purposes. Personal Agents (PAs) handle user interaction, a Directory Services (DS) facility in the Networking Layer handles low-level service lookup, and two subsystems—Security and Privacy, and Context—provide the remaining functional support not handled by the other components. The CASCOM approach has been prototyped in the field of healthcare business, where its role based semantic service discovery approach provides a novel mechanism to support (for example) travellers requiring complex emergency medical and logistical support [14].

Ratsimor et al. [68] note that directory-based service discovery mechanisms do not work well in ad-hoc (especially mobile) environments. In response, Allia has been developed as a peer-to-peer caching based and policy driven agent service discovery framework, in which individual agents form *alliances*. An alliance (of a node) is a set of local agent nodes in the network whose service information is cached by that node. A member of an alliance is aware of the other agents in that alliance but is not aware of which alliances it is a member of. As the network changes in an ad-hoc manner, so do the alliances which have been set up, based on the local topology in the vicinity of a given node, and on the service advertisements that node has received. The dynamic nature of this approach is claimed to be effective in supporting agent-service discovery in dynamically changing ad-hoc environments. It has been implemented as an extension of the LEAP Agent Platform using Bluetooth as the network communications technology, and its performance has been evaluated in a GlomoSim simulator.

The proximity of agents in ad-hoc networks has also motivated work by Campo et al. [16], who proposed a Multi-Agent System for use in pervasive ad-hoc environments. Their system allows agents running on different devices to share services if those devices are close together on the network, and uses a Service Discovery Agent which supports the communication of service ability information between different agents in the network. Middleware supporting this architecture includes the Pervasive Discovery Protocol (PDP), which is fully distributed, supporting service discovery via both push and pull mechanisms. The associated Generic Service Description Language (GSDL) is an XML-based markup language tailored to hierarchical service descriptions in the context of pervasive environments [16].

The A4 (Agile Architecture and Autonomous Agents) management system for grid computing is a distributed software system which uses federating agents to provide services to a large-scale, dynamic, multi-agent system. A4 contains three models—an hierarchical model (a method for organising large numbers of agents), which is supported by a discovery model (for the locating of agent services) and a coordination model (for organising services to provide more complex services). The hierarchical model is illustrated in Figure 2.8. At the top of the hierarchy is a single broker agent, and each sub-level contains a single coordinator agent together

with individual agents and further sub-levels. The topology of the network is dynamic, and each agent can act as a router facilitating communication between agent requesters and service providers. Each agent's service information can be advertised either up or down the hierarchy, and service discovery is likewise facilitated by the topology of the hierarchy. As implemented, agents include the functionality of the PACE performance prediction toolset, allowing efficiency issues for such a system to be investigated [90].

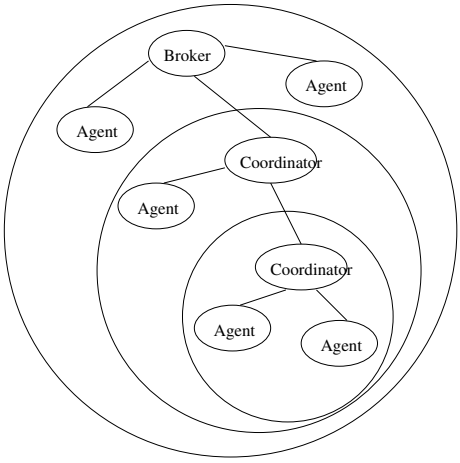

Fig. 2.8 A4: Agent based service oriented architecture

A similar architectural approach has been adopted by the Mobile Service Management Architecture based on Mobile Agent (MA-MSMA), which uses an hierarchical tree-like structure populated by identical mobile agents, each of which can be both a provider and a requester of Grid services. Agents forming internal nodes of the tree adopt the role of broker or lookup agent [43].

The Southampton agent Framework for Agent Research (SoFAR) project shows that the "agent concept can be closely aligned with a web service, in that an agent can be described as a web service and discovered using a standard mechanism UDDI". Using WSDL gives an agent the ability to describe and advertise their capabilities. The use of ontologies as a semantic enhancement to WSDL and UDDI enables services to be discovered and invoked by software through common terminology and shared meanings. Avila-Rosas [5] notes that "this is a vital property in an open system such as the Grid".

The Software Agent-Based Groupware using E-services (SAGE) project "incorporates the use of intelligent agents to integrate human users with web services". The approach taken by SAGE is to identify a (human) user's operational context, and for each agent in the system to learn the rule-based preferences for that user based on that contextual information. This allows for targeting of relevant web services to be identified by the system and presented to the user [12].

Matchmaking and brokering are multi-agent coordination mechanisms for web services. Sycara et al. [81] have used novel extensions to the Web Ontology

Language for Semantic Web Services (OWL-S) and to its process model to implement a broker which both provides discovery services and mediates between agents and web services. They suggest agents might subcontract, by finding and interacting with a provider who can solve a goal. The problems with this approach are similar to those associated with brokering, their current research concerns automatic multi-agent interaction and automatic Web service composition [81].

The Agent Approach for Service Discovery and Utilisation (AASDU) (Figure 2.9) is a flexible and scalable multi-agent system which allows dynamic insertion and deletion of services and lightweight autonomous agents. The approach is underpinned by web standards (including UDDI, SOAP, WSDL and XML), and a communication protocol is employed which does not depend on addresses of the agents sending and receiving messages. An extension to the Oak Ridge Mobile Agent Community (ORMAC) framework is used as the basis of the agent architecture [65].

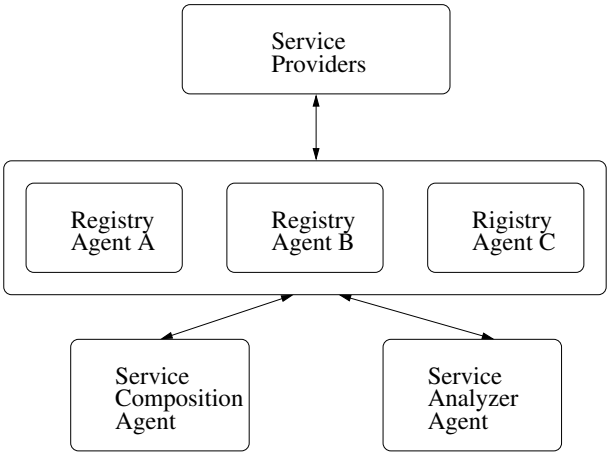

Fig. 2.9 Palathingal and Chandra's agent based service oriented architecture

2.8 Challenges in Service Advertisement and Discovery

Service advertisement and discovery is a focus of active research, and a number of popular areas of investigation and specific challenges have been identified, and can be categorised as for system requirements and for system modelling.

2.8.1 System Requirements

Scalability and adaptability [1, 15]: In particular, Wu [93] proposes that efficient mechanisms should be introduced to allow "system function gracefully at very high load of service discovery requests given reasonable resource consumptions".

Security: Different security requirements for different environments, such as in mobile computing, should be identified. Issues might include secure service registration, and deregistration; secure discovery and secure delivery, secure communication protocols and more appropriate trust models and communication paradigms [3, 24, 26, 44, 76].

Quality of service: Efficient protocols for the awareness of quality of service should be introduced, as "QoS information is particularly important for real time applications like streaming high quality video over wireless networks" [34].

Interoperability: Interoperability is also important since complex messages exchanged by web services are structurally and semantically heterogeneous [2, 56].

2.8.2 System Modelling

Theoretical foundations: Little work has been done on theoretical foundations. No generally accepted principle or procedure for system design and evaluation has been identified so far, although a number of definitions regarding the concepts and principles of service and service oriented architecture have been proposed by different scholars [49, 60].

System structures: A number of different styles of system structures have been summarised in Section 2, and it is challenge to design more and moreover to combine these styles of structures together to improve the overall effectiveness of the systems [32].

Agents based service oriented computing: We have introduced the idea of agent based service discovery in Section 7, and it is important to identify and adopt efficient approaches to merge agent technology into service oriented computing, and develop efficient algorithms for agents to search, match and compose services [5, 11, 76].

Technologies integration: It is also an opportunity and challenge to integrate SOA with other technologies, such as wireless communications and the Grid, in order to provide more powerful advertisement and discovery mechanisms [19, 54, 96].

2.9 Summary

The Service Broker (or Registry), which is one of the entities in current model of Service Oriented Computing, plays a key role in the process of service advertisement

and discovery. Three types of registry architectures have been introduced—*centralised*, *decentralised* and *hybrid*—together with major users' requirements for service advertisement and discovery systems. We have also described a number of *mature* discovery technologies in this area, and explored how agent based technology might improve the effectiveness of this process. This chapter provides a foundation for understanding the rest of the processes in Agent Based Service Oriented Computing.

References

1. AI-Masri, E., Mahnoud, Q.H.: Discovery the Best Web Service. In: WWW Poster Paper, pp. 1257–1258. ACM Press, Canada (2001)
2. Anjum, F.: Chanllenges on Providing Services in a Ubiquitous, Mobile Environment. In: the 3rd International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 1–3. IEEE Press, California (2006)
3. Antonopoulos, N., Shafarenko, A.: An Active Organisation System for Customized, Secure Agent Discovery. The Journal of Supercomputing. 20, 5–35 (2001)
4. Arnold, K., Osullivan, B., Scheifler, R.W., Waldo, J., Wollrath, A., O'Osullivan, B.: The Jini Specification. Addison Wesley, Reading (1999)
5. Avila-Rosas, A., Moreau, L., Dianlani, V., Miles, S., Liu, X.: Agents for the Grid: A Comparison with Web Services. In: Workshop on Challenges in Open Agent Systems, PP. 238–244. Bologna (2002)
6. Baresi, L., Miraz, M.: A Distributed Approach for the Federation of Heterogeneous Registries. In: 4th International conference on Service Oriented Computing, pp. 240–251. Chicago (2006)
7. Balken, R., Haukrogh, J., Jensen, J.L., Jensen, M.N., Roost, L.J., Toft, P.N., Olsen, R.L., Schwefel, H.P.: Context Sensitive Service Discovery Experiment Prototype and Evaluation. Wireless Personal Communications. 40, 417–431 (2007)
8. Baresi, L., Nitto, E., Ghezzi, C., Guinea, S.: A Framework for the Deployment of Adaptable Web Service Compositions. SOCA. 1, 75–91 (2007)
9. Bell, D., Ludwig, S.A., Lycett, M.: Enterprise application reuse: Semantic Discovery of Business Grid Services. Information Technology Management. 8, 223–239 (2007)
10. Benbernou, S., Hacid, M., Liris,: Resolution and Constraint Propagation for Semantic Web Services Discovery. Distributed and Parallel Databases. 18, 65–81 (2005)
11. Blake, M., Cheung, W., Jaeger, M.C., Wombacher, A.: WSC-06: the Web Service Challenge. In: the IEEE international Conference on E-Commerce Technology, pp. 62. IEEE Press, New York (2006)
12. Blake, M.B., Kahan, D. R., Nowlan, M. F.: Context-aware Agents for Use r-oriented Web Services Discovery and Execution. Distributed and Parallel Databases. 21, 39–58 (2007)
13. Bucur, D., Bardram, J.E.: Resource Discovery in Activity-Based Sensor Networks. Mobile Networks and Applications. 12, 129–142 (2007)
14. Caceres, C., Fernandez, A., Ossowski, S., Vasirani, M.: Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach. In: IEEE Intelligent Systems, pp.11–20. IEEE Press, New York (2006)
15. Cao, J., Kerbyson, D.J., Nudd, G.R.: High Performance Service Discovery in Large-Scale Multi-Agent and Mobile-Agnet Systems. International Journal of Software Engineering and Knowledge Engineering. 11, 621–641 (2001)
16. Campo, C.: Service Discovery in Pervasive Multi-agent Systems. In: Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices, pp. 133–146. Bologna (2002)
17. Campo, C., Munoz, M., Perea, J.C., Mann, A., Garcia-Rubio, C.: PDP and GSDL: A New Service Discovery Middleware to Support Spontaneous Interactions in Pervasive Systems. In: 3rd IEEE International Conference on Pervasive Computing and Communications, pp. 178–182. IEEE Press, New York (2005)

18. Caporuscio, M., Carzangiga, A., Wolf, A.L.: Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications. IEEE Transactions on Software Engineering. 29, 1059–1071 (2003)
19. Caron, E., Desprez, F., Tedeschi, C.: Enhancing Computational Grids with Peer-to Peer Technology for Large Scale Service Discovery. Journal of Grid Computing. 5, 337–360 (2007)
20. Chakraborty, D., Joshi, A., Yesha, Y., Finin, T.: Toward Distributed Service Discovery in Pervasive Computing Environments. IEEE Transactions on Mobile Computing. 5, 97–112 (2006)
21. Chappell, D.: Who Cares about UDDI. Addison Wesley, New York (2002)
22. Charlet, D., Issarny, V., Chibout, R.: Service Discovery in Multi-radio Networks: An assessment of Existing Protocols. In: MSWiM'06, pp. 229–238. ACM Press, New York (2006)
23. Chen, H., Joshi, A., Finin, T.: Dynamic Service Discovery for Mobile Computing: Intelligent Agents Meet Jini in the Aether. Cluster Computing. 4, 343–354 (2001)
24. Cotroneo, D., graziano, A., Russo, S.: Security Requirements in Service Oriented Architectures for Ubiquitous Computing. Middleware for Pervasive and Ad-Hoc Computing. In: 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing, pp.172–177. ACM Press, Canada (2004)
25. Curbera, F., Duftler, M., Khalaf, D., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web Services Web, An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing. 6, 86–93 (2002)
26. Czerwinski, S., Zhao, B., Hodes, T. D., Joseph, vA.D., Katz, R.H.: An Architecture for A Secure Service Discovery Service. In: International Conference on Mobile Computing and Networking, pp. 24–35. Washington (1999)
27. Data Distribution Service, `http://www.omg.org`
28. Degwekar, S., Lam, H., Su, S.Y.W.: Constraint-Based Brokering(CBB) for Publishing and Discovery of Web Services. Electronic Commerce Research. 7, 45–67 (2007)
29. Doulkeridis, C., Vazirgiannis, M.: Querying and Updating a Context-aware Service Directory in Mobile Environments. In: IEEE/WIC/ACM Int. Conference on Web Intelligence (WI'04), pp.562–565, IEEE Press, New York (2004)
30. Doulkeridis, C., Zafeiris, V. N?rv?g, K., Vazirgiannis, M., Giakoumakis, E.A.: Context-Based Caching and Routing for P2P Web Service Discovery. Distrib Parallel Databases. 21, 59–84 (2007)
31. Dustdar, S., Treiber, M.: A View Based Analysis on Web Service Registries. Distributed and Parallel Databases. 18, 147–171 (2005)
32. Dustdar, S., Treiber, M.: View Based Integration of Heterogeneous Web Service Registries—the Case of VISR. World Wide Web. 9, 457–483 (2006)
33. ebXML Project, `http://www.ebxml.org`
34. Fan, Z., Ho, E.G.: Service Discovery in Ad Hoc Networks: Performance Evaluation and QoS Enhancement. Wireless Personal Communications. 40, pp. 215–231 (2007)
35. Flores-Cortés, C.A., Blair, G.S., Grace, P.: A Multi-Protocol Framework for As-hoc Service Discovery. In: MPAC'06, pp.10. ACM Press, New York (2006)
36. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: the Proceddding of the IFIP International Conference on Network and Parallel Computing, pp. 2–13, Springer-Verlag, New York (2006)
37. Friday, A., Davies, N., Wallbank, N., Catterall, E., Pink, S.: Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments. Wireless Networks. 10, 631–641 (2004)
38. Garg, P., Griss, M., Machiraju, V.: Auto-Discovery Configurations for Service Management. Journal of Network and Systems Management. 11, 217–239 (2003)
39. Ghamri-Doudane, S., Agoulmine, N.: Enhanced DHT-Based P2P Architecture for Effective Resource Discovery and Management. Journal of Network and Systems Management. 15, 335–354 (2007)
40. Globus Project, `http://www.globus.org/`
41. Gnutella Project, `http://www.gnutella.com/`
42. Guttman, E.: Service Location Protocol: Automatic Discovery of IP Network Service. IEEE Internet Computing. 3, 71–80 (1999)

43. He, Y., Wen, W., Jin, H., Liu, H.: Agent based Mobile Service Discovery in Grid Computing. In: Proceedings of the Fifth International Conference on Computer and Information Technology, pp. 78–101. IEEE Press, New York (2005)
44. Hodes, T.D., Czerwinski, S.E, Zhao, B.Y., Joseph, A.D., Katz, R.H.: An Architecture for Secure Wide-Area Service Discovery. Wireless Networks. 3, 213–230 (2002)
45. Hoffert, J., Jang, S., Schmidt, D.C.: A Taxonomy of Discovery Services and Gap Analysis for Ultra-Large Scale Systems. In: ACMSE 2007, pp. 355–361. ACM Press, New York (2007)
46. Hoschek, W.: The Web Service Discovery Architecture. In: ACM/IEEE SC Conference (SC'02), pp.38. IEEE Press, New York (2002)
47. Hu, C., Zhu, Y., Huai, H., Liu, Y., Ni, L.M.: S-Club: An Overlay-Based Efficient Service Discovery Mechanism in CROWN Grid. Knowledge and Information Systems. 12, 55–75 (2007)
48. Huang, A. C., Steenkiste, P.: Network-Sensitive Service Discovery. Journal of Grid Computing. 1, 309–326 (2003)
49. Huhns, M., Singh, M.: Service Oriented Computing: Key Concepts and Principles. IEEE Internet Computing. 9, 75–81 (2005)
50. Issarny, V., Caporuscio, M., Georgantas, N: A Perspective on the Future of Middleware-Based Software Engineering. In: Future of Software Engineering, pp. 244–258. IEEE Press, New York (2007)
51. Jini Lookup Service, `http://www.jini.org/`
52. JXTA Project, `https://jxta.dev.java.net/`
53. Kontogiannis, K., Smith, G.A., Litoiu, M., Müller, H., Schuster, S., Stroulia, E.: The Landscape of Service Oriented Systems: A Research Perspective. In: the International Workshop on Systems Development in SOA Environments, pp. 1. IEEE Press, New York (2007)
54. Li, J., Mohapatra, P.: PANDA: A Novel Mechanism for Flooding Based Route Discovery in Ad-hoc Networks. Wireless Netw. 12, 771–787 (2006)
55. Luck, M., McBurney, P., Shehory, O., Willlmott, S.: Agent Technology: Computing as Interaction. University of Southampton, Southamptom (2005)
56. Nagarajan, M., Verma, K., Sheth, A.P., Miller, J., Lathem, J.: Semantic Interoperability of Web Services—Challenges and Experiences. In: Proceeding of the IEEE International Conference on Web Services, pp.373–382. IEEE Press, New York (2006)
57. Napster Project, `http://www.napster.co.uk/`
58. Naumenko, A., Nikitin, S., Terziyan, V.: Service Matching in Agent Systems. Applied Intelligence. 25, 223–237 (2006)
59. Nedos, A., Singh, K., Clarke, S: Mobile Ad Hoc Services: Semantic Service Discovery in Mobile Ad Hoc Networks. Springer, Berlin (2006)
60. Newcomer, E., Lomow, G.: Understanding SOA with Web Services. Addison Wesley, London (2005)
61. OASIS Homepage, `http://www.oasis-open.org/home/index.php`
62. O'Brien, L., Merson, P., Bass, L.: Quality Attributes for Service Oriented Architectures. In: Internal Workshop on Systems Development in SOA Environments, pp. 216–122. IEEE Press, New York (2007)
63. Object Management Group, `http://www.omg.org/gettingstarted/`
64. The Open Group Homepage, `http://www.opengroup.org/`
65. Palathingal, P., Chandra, S.: Agent Approach for Service Discovery and Utilization. In: Proceedings of the 37th Hawaii International Conference on System Sciences, pp. 1–9. IEEE Press, New York (2004)
66. Papazoglou, M.P., Krimer, B.J., Yang, J.: Leveraging web services and Peer to Peer Networks. Springer, Berlin (2003)
67. Papazoglou, M., Heuvel, W.: Service Oriented Architectures: Approaches, Technologies and Research Issues. The VLDB Journal. 16, 389–415 (2007)
68. Ratsimor, D. Chakraborty, D., Joshi, A., Finin, T.: Allia: Alliance-Based Service Discovery for Ad-Hoc Environments. In: International Workshop on Mobile Commerce, pp. 1–9. ACM Press, New York (2002)

69. Ratsimor, O. Chakraborty, D. Joshi, A., Finin, T., Yesha, Y.: Service Discovery in Agent-Based Pervasive Computing Environments. Mobile Networks and Applications. 9, 679–692 (2004)
70. Richard III, G.G.: Service Advertisement and Discovery: Enabling Universal Device Cooperation. IEEE Internet Computing. 5, 18–26 (2000)
71. Salutation Architecture Specification, http://www.salutation.org/specordr.htm
72. Scherson, I.D. and Cauich, E., Valencia, D.S.: Service Discovery for GRID Computing Using LCAN-mapped Hierarchical Directories. Journal of Supercomputing. 42, 19–32 (2007)
73. Service Oriented Architecture, http://www.w3.org/TR/ws-arch
74. ShaikhAli, A., Rana, O.F., AI-Ali, R., Walker, D.W. UDDIe: an tetended registry for web services. In: the Proceedings of Application and the Internet Workshops, pp.85–89, IEEE Press, New York (2003)
75. Singh, M.P., Huhns, M.N.: Service Oriented Computing, Semantics, Processes, Agents. John Wiley & Sons, Chichester (2005)
76. Singha, A.: Web Services Security: Chanllenges and Techniques. In: 8th IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 282. IEEE Press, New York (2007)
77. Sivavakeesar, S., Gonzalez, O.F., Pavlou, G.: Service Discovery Strategies in Ubiquitous Communication Environments. IEEE Communications Magazine, 12, 106-113 (2006)
78. SOAP Specification, http://www.w3.org/TR/soap/
79. Soldatos, J., Dimarkis, N., Stamatis. K., Polymenakos, L.: A Breadboard Architecture for Pervasive Context-Aware Services in Smart Spaces: Middleware Components and Prototype Applications. Personal and Ubiquitous Computing. 11, 193–212 (2007)
80. Sreenath, R., Singh, M.: Agent based service selection. Web Semantics: Science, Services and Agents on the World Wide Web. 1, 261–279 (2004)
81. Sycara, K., Paolucci, M., Soudry, J., Srinivasan, N.: Dynamic Discovery and Coordination of Agent Based Semantic Web Services. IEEE Internet Computing, 66–73 (2004)
82. Talwar, B., Venkataram, P., Patnaik, L.M.: A Method for Resource and Service Discovery in MANETs. Wireless Personal Communications. 41: 301–323 (2007)
83. Tyan, J., Mahmoud, Q.H.: A Comprehensive Service Discovery solution for Mobile Ad-Hoc Networks. Mobile Networks and Applications. 10, 423–434 (2005)
84. UDDI Project Version 3.0.2, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm#_Ref8884251
85. Vanthournout, K., Deconinck, G., Belmans, R.: A Taxonomy for Resource Discovery. Personal and Ubiquitous Computing. 9, 81–19 (2005)
86. Verma, K., Sivashanmugam, K., Sheth, A. Patil, A., Oundhakar, S., Miller, J.: METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Information Technology and Management. 6, 17–39 (2005)
87. Vitvar, T., Mocan, A., Kerrigan, M., Zaremba, M., Zeremba, M., Moran, M., Cimpian, E., Haselwanter, T., Fensel, D.: Semantically-Enable Service Oriented Architecture: Concepts, Technology and Application. In: Service Oriented Computing and Applications. 1, 129–154 (2007)
88. W3C School, http://www.w3schools.com/
89. Wang, H., Huang, J. Z., Qu, Y., Xie, J.:Web Semantics: Science, Services and Agents. World Wide Web. 1, 309–320 (2004)
90. Warwick University Computer Science Department High Performance Systems Research Group, http://www.dcs.warwick.ac.uk/research/hpsg/A4/A4.html
91. Web Services Inspection Language, http://www.ibm.com/developerworks/library/ws/wsilover/
92. WSDL Specification, http://www.w3.org/TR/wsdl
93. Wu, C., Chang, E.: Aligning with the Web: an Atom-based Architecture for Web Service Discovery. SOCA. 1, 97–116 (2007)
94. Yang, Y., Dunlap, R., Rexroad, M, Cooper, B.: Performance of full text search in structured and unstructured peer to peer systems. In: Proceedings of the 5th IPTPS, pp. 27–28. Santa Barbara, USA (2006)

95. Yu, J., Venugopal, S., Buyya, R.: A Market-Oriented Grid Directory Service for Publication and Discovery of Grid Service Providers and their Services. Journal of Supercomputing. 36, 17–31 (2006)
96. Yu, Q., Liu, X., Bouguettaya, A., Medjahed, B.: Deploying and managing Web Services: Issues, Solutions and Directions. The VLDB Journal The International Journal on Very Large Data Bases. 17, 537–572 (2006)