Grant Agreement N° 215483

Title:	*CD-JRA-2.3.3: Requirements for Service Infrastructures in Dynamic Environments and Evaluation of Existing Service Registries*
Authors:	*TUW, CNR, Tilburg*
Editor:	*Philipp Leitner (TUW)*
Reviewers:	*Jean-Louis Pazat (INRIA)*
	Angela Kounkou (CITY)
Identifier:	*Deliverable #* CD-JRA.2.3.3
Type:	*Deliverable*
Version:	*1*
Date:	*March 2, 2009*
Status:	*Final*
Class:	*Public*

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 215483 (S-Cube).

Grant Agreement N° 215483

Management Summary

Web service registries are tools for the implementation of loosely-coupled service-based systems. For instance, business processes query registries in order to find services which implement functionality that is needed in the process, and adaptable service compositions need to be aware of which alternatives are available for each service. Furthermore, there is a clear interrelation between end-to-end quality provisioning and monitoring, and service registries, since SLA monitoring and enforcement is based on the availability of a service repository providing an expressive set of metadata. Even more, with the advent of the Internet of Services, an Internet-scale Web service ecosystem with unique scale and heterogeneity characteristics, a number of new challenges for the next generation of Web service registries will arise. First of all, the sheer size of the ecosystem (in terms of number of clients, providers and services) will cause a need for new scalable service discovery mechanisms built on the notions of the Internet. This includes not only discovery of atomic services, but also of task flows (ad hoc service mashups). Additionally, the distributed and heterogeneous nature of the Internet of Services asks for new data dissemination methods between physically and logically disjoint registry entities, which work in spite of missing, untrusted, inconsistent and wrong data. Further challenging requirements are going to be put forward by mobile, human-provided and ad hoc services, which are common in the Internet of Services. These services are volatile in nature, and need to be actively tracked by the service registries. Finally, another class of challenges is introduced by the human factor in the Internet of Services -- since services are often consumed and provided by humans, new means of evaluating service performance based on user-perceived and fuzzy Quality of Experience metrics need to be devised. In this deliverable we describe these requirements for the next generation of service registries for large-scale service environments in detail, and explain why we consider existing registry approaches as not sufficient for these environments. The deliverable provides the baseline research topics to be covered by the ``registry segment'' of the work package WP-JRA-2.3 in S-Cube; research questions steering the second part of the work package focussing on adaptation are described within the deliverable CD-JRA-2.3.2.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 215483 (S-Cube).

Members of the S-Cube consortium:

University of Duisburg-Essen (Coordinator)	Germany
Tilburg University	Netherlands
City University London	U.K.
Consiglio Nazionale delle Ricerche	Italy
Center for Scientific and Technological Research	Italy
The French National Institute for Research in Computer Science and Control	France
Lero - The Irish Software Engineering Research Centre	Ireland
Politecnico di Milano	Italy
MTA SZTAKI – Computer and Automation Research Institute	Hungary
Vienna University of Technology	Austria
Université Claude Bernard Lyon	France
University of Crete	Greece
Universidad Politécnica de Madrid	Spain
University of Stuttgart	Germany
University of Hamburg	Germany
Vrije Universiteit Amsterdam	Netherlands

Published S-Cube documents

All public S-Cube deliverables are available from the S-Cube Web Portal at the following URL:

http://www.s-cube-network.eu/results/deliverables/

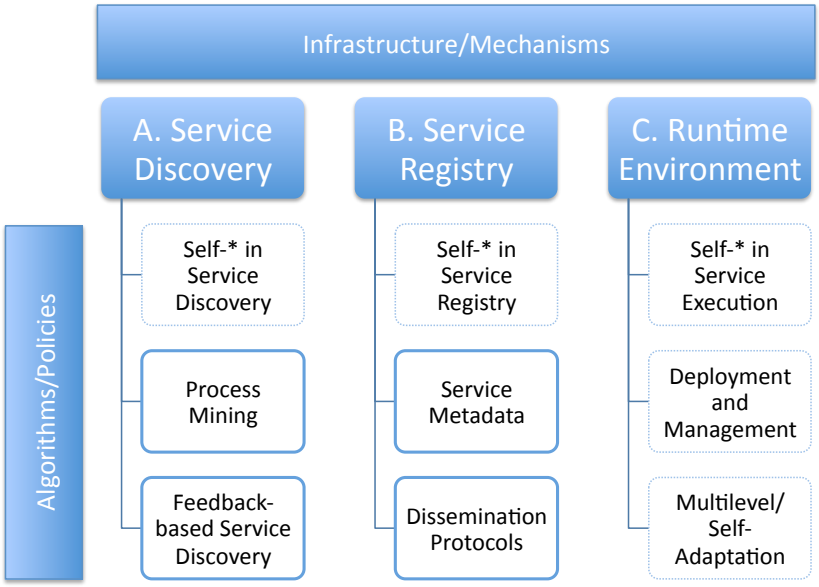

Figure 1.1: WP-JRA-2.3 Research Architecture

In Figure 1.1 we give an overview of the overall research architecture of WP-JRA-2.3: research on service infrastructures is structured in three threads, *Service Discovery*, *Service Registries* and *Service Execution*. To each of those categories, a number of concrete research items (research directions) are denoted. In this document we will describe the challenges of *process mining*, *feedback-based service discovery*, *service metadata*, and *dissemination protocols* in more detail. The other research challenges are described in the other WP-JRA-2.3 M12 deliverable, CD-JRA-2.3.2. Note that this document purposefully excludes some aspects of services infrastructure, since they are out of scope of the work package: service composition is handled by work package JRA-2.2, and has therefore been excluded here; similarly, we do not consider implementation processes for service engineering, since this is the focus of work package JRA-1.1. Security as a cross-cutting concern is not mentioned explicitly, but is inherently important for all research directions discussed below (e.g., security is an inherent concern of large-scale data dissemination).

- Service Discovery – service discovery is a fundamental element of service-oriented architectures. Other, and more complex, services heavily rely on it to enable the execution of service-based applications. Current discovery mechanisms are not well prepared to deal with the huge number of services which are envisioned for large-scale service ecosystems, such as the Internet of Service (IoS). In order to enable the IoS, novel discovery mechanisms must be able to deal with millions (or even billions) of services. Additionally, these discovery mechanisms need to consider new constraints, which are not prevalent today, such as Quality of Experience requirements and expectations (feedback) of users, geographical constraints, pricing and contractual issues, or invocability (not every service can be invoked and used by every client).

- Service Registry – service registries are tools for the implementation of loosely-coupled service-based systems. Current registries such as UDDI are not fault-tolerant. They are centrally managed entities, which do not scale well and do not offer support for rich service descriptions. With the advent of the Internet-scale service ecosystems a number of new challenges for the next generation of registries will arise. In such systems, fault tolerance and scalability of registries is of eminent importance. Autonomic registries need to be able to form loose federations, which are able to work 24/7, in spite of heavy load or failures. Additionally, a richer set of metadata (data describing services) is needed for services in such ecosystems, in order to capture novel aspects such as

self-adaptation, user feedback evaluation, or Internet-scale process discovery. Another research topic is the dissemination of metadata: the distributed and heterogeneous nature of these ecosystems asks for new dissemination methods between physically and logically disjoint registry entities, which work in spite of missing, untrusted, inconsistent and wrong metadata.

- Runtime Environment – the obvious need for automatic, autonomic approaches at run time have been addressed but either the scope is slightly different or the thoroughness, completeness of the solution does not fully meet the requirements. Current adaptation methods focus on components and put little emphasis on services. Opposed to the self-* idea (refer to the deliverable CD-JRA-2.3.2 for details), they are merely targeting one or few reflective properties, typically some form of fault tolerance. Fault tolerance itself is usually a less complete solution than self-healing. Most approaches are targeted at short term remedies, typically eliminating one problem. They are also acting locally (both spatial and temporal), not taking into consideration the overall state of the system or the chain of potential/possible causes and actions. There are various adaptation mechanisms for initial/bootstrapping configuration but there are few solutions that solve run-time dynamic adaptation by reconfiguration. As opposed to current approaches we envision an infrastructure that is able to adapt autonomously and dynamically to changing conditions. Such adaptation should be supported by past experience (learning), should be able to take into consideration a complex set of conditions and their correlations, act proactively to avoid problems before they can occur and have a long lasting, stabilizing effect. These research topics possess a common feature that is characteristics of the entire research thread: policies for adaptation, and a knowledge base for adaptation strategies are defined.

1.2 Outline

The rest of this document will be structured as follows: in Section 2 we sketch the model of a large-scale dynamic ecosystem of services (the Internet of Services) that we use as basis for the rest of this document, in Section 3 we detail this model by explaining a high-level case study of a GPS unit manufacturer in the Internet of Services, Section 4 contains the main contribution of this document, a detailed description of the main research issues that need to be dealt with within the S-Cube work, Section 5 summarizes the main reasons why current Web service registries such as UDDI or the ebXML registry cannot be used to tackle these issues, and finally Section 6 summarizes the document's most important points.

Chapter 2

A Dynamic Ecosystem of Services

In this deliverable we consider a dynamic ecosystem of services, which we see as the the upcoming unification of Web 2.0, Service-Oriented Architectures (SOA), Ubiquitous and Pervasive Computing, Semantic Web Services and Mobile Computing. Schroth has coined the term "Internet of Services" (IoS for short) to describe this next-generation services environment [5]. This vision is gradually becoming reality: broadband Internet connections are becoming ubiquitous (even "on the road", in the form of public Wireless LAN, EDGE or GPRS), more and more businesses, even small and medium-sized ones, are delivering business over the Internet; and previously rather separated research communities such as the services and mobile computing communities are beginning to explore possibilities of combinations to break the ground for novel applications, e.g., mobile service delivery. The current research and industry interest in Software as a Service [6] (SaaS) is just one example of a global Internet-based service economy slowly becoming reality.

The main building blocks of the IoS are services. In general, we can assume that IoS services will exhibit some of the well-known characteristics of services in SOA, i.e., they are autonomous, reusable, composable and discoverable entities, which are loosely coupled to its clients and are defined using a machine-readable interface [7]. Additionally, as the name Internet of Services suggests, we assume that services are provided over the Internet, i.e., the service bus over which clients and providers communicate is the Internet (the *Internet Service Bus*). Furthermore, IoS services may have the following specific characteristics, which distinguish them from services in the current understanding of the SOA community:

- *Human-provided services* – the trend towards human-provided services is already visible today through research approaches such as HpS [8] or specifications such as WS-BPEL4People [9, 10]. The IoS is going to continue this trend by taking humans "into the loop": services, like today's web content, will be provided, consumed, rated, tagged and mashed up not only by machines (in a fully automated way), but also by humans (e.g., through their PDAs or mobile devices).

- *Ad hoc services* – unlike traditional enterprise services, IoS services are provided often in an "ad hoc" manner, that is, spontaneously and only for a brief period of time. These services are special in that they exist only very briefly, and for a very specific purpose and target audience. For example, think of a car driver who happens to be stuck at the beginning of a traffic jam, and spontaneously provides status updates of how the jam is progressing. This service has a very brief time to live, and a locally bounded target audience (other drivers stuck in the back of the jam, who cannot see what is going on). Additionally, IoS services are mashed up in an ad hoc manner, the resulting composition is often executed only once and discarded directly after use. The ad hoc nature of the IoS is a direct result of the inclusion of humans.

- *Dynamic services* – another direct result of human involvement is the dynamic and pervasive nature of the IoS. Service providers emerge and disappear quickly (e.g., as ad hoc services or mashups), and unlike the standard definition of a service in SOA, human-provided services cannot

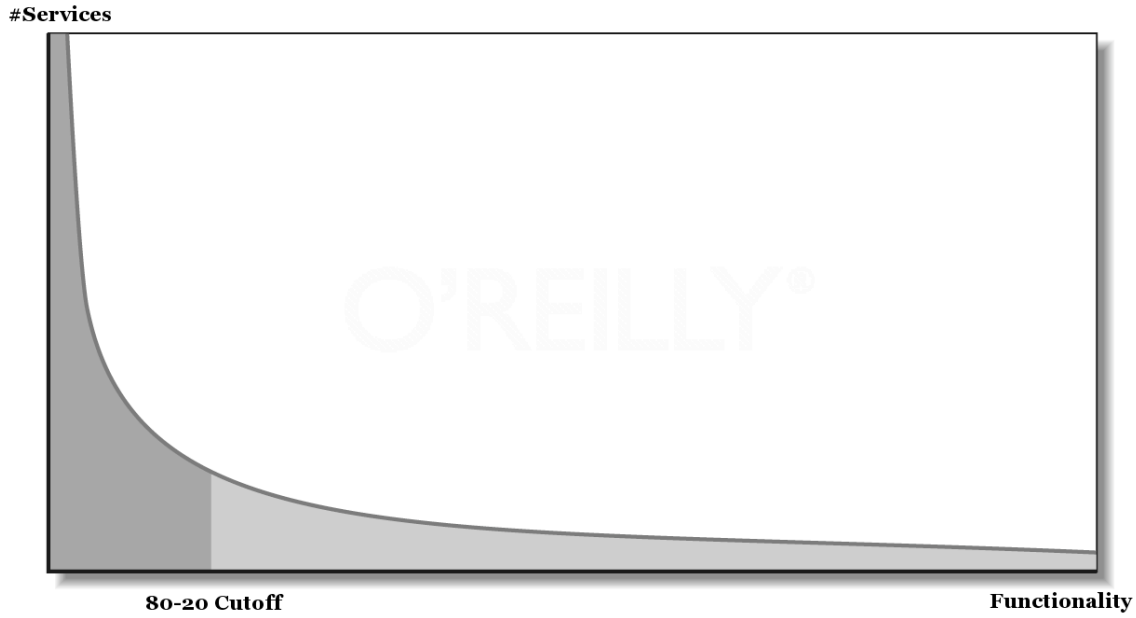

Figure 2.1: Power Law Distribution of Services in the IoS

be considered "always on". Another characteristic of dynamicity in the IoS are location-aware and mobile services. Such services are usually hosted on handheld device, and can therefore change their physical location easily and frequently.

- *Quality of Experience* – Quality of Service (QoS [11]) will be defined differently for human-provided and consumed services: while technical services are usually compared based on a limited number of technical parameters (response time, throughput, availability, ...), human services will be evaluated based on a broader spectrum of "fuzzier" metrics, such as the personal satisfaction of a human service consumer. We refer to this more abstract quality notion as *Quality of Experience* (QoE [12]). Note that, unlike QoS values, which are identical for every service client, QoE is dependent not only on the service provider, but also on characteristics, requirements and context of the service client. Estimating the QoE before using a service is therefore a very challenging problem.

- *Large-scale service ecosystem* – today's SOAs are centered around a limited number of often well-known services. The IoS, on the other hand, might consist of millions of services. Service discovery and rating will therefore be of much more significance than in today's SOAs, and challenging new research problems will come up. Service lookup through a single central registry entity will not be feasible in such a large-scale service environment.

An assumption that we will be able to inherit from traditional Web technologies is that the service ecosystem in the IoS will form a Small World [13, 14]: many services will be invoked just a few times whereas a very small subset of them will interact with a huge number of different services. In general, we can assume that the distribution of service in the IoS follows a power law distribution: a relatively small number of functionalities (features) will be provided by the majority of services in the IoS; the rest of the services will provide some rare special-purpose functionality (the "long tail" of the power law). Therefore the 80-20 rule can be expected to apply also to services on the IoS: a vast 80% of the services will provide only 20% of the functionality in the IoS. Figure 2.1 visualizes this dependency.

In current SOA research, services are often seen equivalent to SOAP and WSDL based Web services. However, in the IoS this is expected to change: in Web 2.0 both SOAP and WSDL are of rather low

importance; instead the Web 2.0 has seen the rise of RESTful Web services [15], that leverage existing standards and technologies such as HTTP, URI, POX (plain old XML) and JavaScript (JSON). We assume that the IoS will continue this trend towards a lightweight service model. However, this also demands for a paradigm shift in the services community – currently the common answer to most research problems is to add additional standards and layers of complexity (cp. the WS-* stack) and to formalize processes and interactions as far as possible. This kind of solution will not be in line with the human-centered nature of the IoS. Therefore, a lot of well-known solutions from the SOA community will need to be reconsidered in the changed environment of the IoS.

These properties of the upcoming Internet of Services are the basis of the research challenges that we present as one of the main contributions of this deliverable in Section 4.

Chapter 3

Illustrative Example

The Internet of Services as discussed in Section 2, is an Internet-scale dynamic ecosystem of public Web services, such as Pizza delivery services, hotel services, or taxi services. Services in the IoS are often mobile (in the sense that their physical location changes) and/or volatile, and provided services change frequently. We will now detail the relevance of the IoS to the S-Cube vision by presenting an exemplatory case study.

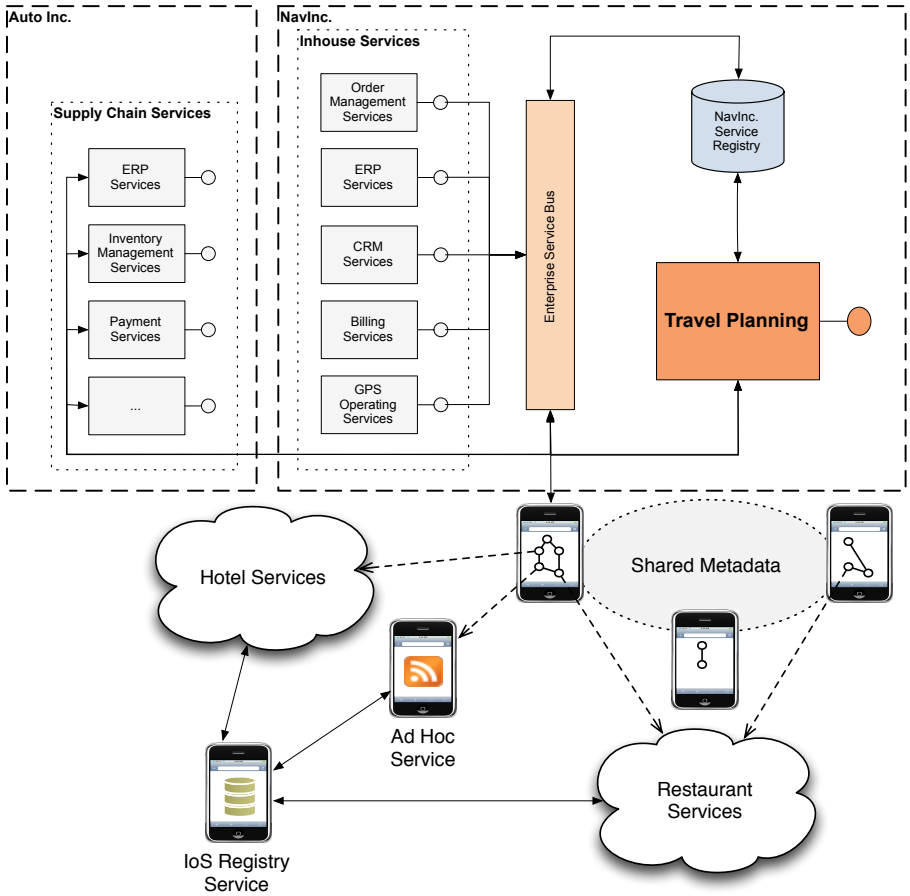

Figure 3.1: Illustrative Example in the Internet of Services

The case study considers a fictional producer of car GPS units, *NavInc.*, which is a supplier of *Auto Inc.*, the subject of the overall S-Cube case study. NavInc. produces high-end "intelligent" navigation units, which offer enough processing capabilities to execute non-trivial applications, and broad-band UMTS networking to connect to the Internet (and, consequently, to the Internet of Services).

NavInc. wants to make use of the almost unlimited possibilities offered by the IoS, and launches a prototypical service to its users, which allows them to utilize their navigational units for ad hoc travel planning. For that, they can access both (selected) internal NavInc. services (e.g., payment can be handled over NavInc. for comfort and security reasons) and public services in the IoS. An application preinstalled on the unit acts as the gateway to the services world. This application grants access to the internal services, and uses a services-based crawling mechanism to search for external services. The application suggests typical ad hoc workflows for travel planning (i.e., get a hotel, get a taxi to the hotel, order something to eat, ...), which the user can use as well as adapt. Selection of concrete services to use for each activity in the workflow is handled by suggesting a defined number of most appropriate services to the user, based on user preferences, context and historical service data. Furthermore, after usage, the user can provide feedback in a structured (rate service from 1 to 5) or semistructured way (add tags to service). The travel service can use this data to evaluate the overall quality of services in the IoS, and enhance the experience of this and other users. This metadata is shared between different units in a decentralized P2P fashion.

As a second feature, the application allows users to actively contribute to the IoS by providing travel-related ad hoc services. For instance, an adventure tourist may visit a volcano in Hawaii during a big eruption, and spontaneously decide to provide live coverage of the event. The application allows her to provide this coverage as a Web service to other users (paid or unpaid), over a simple-to-setup RESTful interface. This is a typical example of a human-provided service. However, not only atomic services can be provided that way – users may also decide to provide value-added (composite) services in an ad hoc way. The application supports the provisioning of ad hoc workflows or explicitly composed services as composites trough a simple point-and-click composer. These composite services are adaptive, in the sense that the infrastructure monitors if the underlying requirements that led to the inclusion of a service are still valid (e.g., a high-quality composite service for complete trips may select only hotel services with a rating of > 4.5, and needs to reselect a hotel if the rating of the chose hotel drops below that border). Requirements are formulated in a simple rule language. Monitoring of compositions is a feature provided by NavIncs. travel planning infrastructure, and the user owning a composite is only alerted if the integrated adaptation mechanisms cannot handle the situation automatically (e.g., there is no valid service to choose any more).

Both atomic and composed ad hoc services can be registered in special IoS services, which we refer to as Registry Services. These registries are not provided by NavInc. – they are public directory services provided by external entities (e.g., Google, Yahoo, ...), comparable to the Open Directory project[1] in the current WWW. Note that registry services in the IoS are not Universal Business Registries. They are more similar to super peers in a P2P network – they have more knowledge about the environment (existing services, feedback from users, ...) than the average service client, however, no registry service is *the* central authority in the IoS.

We use this case study to as a background motivational example for the requirements and research challenges that we present in this deliverable. Therefore, Section 4 will present challenges which result from a scenario such as the one discussed here. The scenario is further extended in deliverable CD-JRA-2.3.2, which uses a similar case study in order to showcase the missing research challenges, which are not covered in this deliverable (cp. the research vision presented in Section 1).

[1] `http://www.dmoz.org/`

Chapter 4

Requirements for Next-Generation Service Infrastructures

The rise of the Internet of Services as the next-generation dynamic service environment will lead to a number of research challenges for the services community. Given the crosscutting character of the IoS, these challenges are of economical, social and technical nature. In this deliverable, we focus on the technical challenges that the IoS poses on the IT infrastructure, especially next-generation service registries. Even though the IoS is all about humans, it is the World Wide Web acting as a *service bus* that enables it. Humans use an IT infrastructure to publish, find, use, compose and rate services. Service providers and clients interact over the service bus (as opposed to interacting directly). However, note that given the global scope of the Internet of Services it would be naive to assume a single, homogeneous middleware. Instead, many different systems built on top of many different technological platforms have to work together to make the Internet of Services real, just like they do for the WWW. Referring back to the case study presented in Section 3 we can illustrate that point: NavInc. is one provider of middleware who supplies his customers with technical means to enter the Internet of Services, however, the NavInc. middleware needs to integrate with current or future systems of different vendors, some of which NavInc. might not even know about, in order to unleash the full power of the Internet of Services.

As discussed in Section 1, and shown in Figure 1.1, we discuss the requirements emerging from the research topics *process mining*, *feedback-based service discovery*, *service metadata*, and *dissemination protocols* in this document. To give an overview, we now summarize these research challenges in large-scale service ecosystems:

- *Process Mining*: today's enterprise SOAs rarely contain more than a few hundred services, usually developed in-house by well-known internal departments. With the advent of the IoS, and the opening of enterprise SOAs "to the cloud"[1] this number is going to increase by many orders of magnitude. Current service discovery mechanisms as presented within the S-Cube deliverable PO-JRA-2.3.1 are not well prepared to deal with such a multitude of services (also see Section 5 in this document). In order to enable the IoS, novel registries will have to be put in place which incorporate new discovery mechanisms which are able to deal with 10^6 to 10^9 services. Additionally, these discovery mechanisms need to consider new constraints which are not prevalent today, such as QoE requirements and expectations of users, geographical constraints, pricing and contractual issues, or invocability (not every service can be invoked and used by every client). Today, indices of service registries mainly rely on passive registration – it is the responsibility of service providers to register their service with every registry index. For the IoS, where service registries are diverse, this approach is infeasible. These registries will need to actively search for new services to add to their index. This is similar in scope to WWW search engines. Novel registry mining techniques need to be developed, which take the peculiarities of services in the IoS into account. Additionally, mashups

[1] `http://blogs.zdnet.com/service-oriented/?p=800`

(ad hoc compositions of services) will be a hot topic in the Internet of Services. Registries will need to support mashing up services by providing suggestions based on recorded previous compositions, geography, context, user preferences and collected usage data. Conceptually, such mashups are similar to Web service compositions (as discussed within S-Cube work package WP-JRA-2.2), however, mashups need to incorporate the ad hoc nature of the human-centric IoS: when executing a mashup, the involved humans can change their mind at any time, and may want to adapt their plans during execution. The infrastructure needs to be flexible enough to not only allow this spontaneous deviation, but even leverage on it by suggesting possible improvements during execution time. Additionally, registries will need to be prepared for re-discovery of services and possible execution paths based on changed user preferences or external events.

- *Feedback-based Service Discovery*: rating services based on Quality of Experience is much more difficult than measuring the QoS of traditional Web services. Usually, the service registry cannot handle this automatically, but needs to incorporate explicit and implicit user feedback into the rating function. Explicit feedback may be given by tags ("good", "bad", "did not work", ...), while implicit feedback may depend on monitoring the user's behavior – if a human consumed a certain service in a specific situation, and decides to select a different one when faced again with a similar decision, then the registry can take this metadata as a hint that the user was not satisfied with the experience delivered by the service. On the other hand, if a user keeps utilizing the same service over and over again the registry may conclude that this specific user is rather satisfied. Obviously, the main complexity associated with rating services in the Internet of Services is the inherent "fuzziness" of human-perceived quality. Many aspects and lessons learned of Web 2.0 will need to be incorporated in order to deal with this inaccuracy.

- *Service Metadata*: service metadata includes all data describing services, including functional contract descriptions, semantics, current position or status or user feedback on past executions. For the Internet of Services, a richer set of metadata than what is currently available (mostly functional interfaces through WSDL, and to a very limited extent service semantics through e.g., OWL-S) needs to be defined. The metadata model needs to be open and extensible, and provides the basis of the data which has to be spread using epidemic protocols (see below). Research in the *Service Discovery* thread of the work package relies on all necessary metadata to be available (i.e., for QoE-based service discovery enough user feedback and historical usage data needs to be available). Additionally, new service monitoring and tracking facilities need to be devised in order to deal with the inherently unreliable nature of services in the Internet of Services: human-provided services cannot be assumed "always on", mobile services may go offline in one location and turn up again somewhere different, and some services may fluctuate (go on and off) in a non-predictable way. Service registries will need to be aware of which services are currently available and which are not, and need to track the position of mobile services on their way. Services themselves will need to adapt to these adverse conditions in order to achieve an expected Quality of Service. Adaptation can be either human commenced or autonomously executed by the Self-* system. This monitoring and tracking needs to be loosely coupled, and scalable to Internet scale. Another important research question is the distributed monitoring of mobile services: responsibility for monitoring these services need to be handed over from registry to registry on the border of monitoring domains.

- *Dissemination Protocols*: the distributed nature of the IoS does not allow for a central UBR-like data dissemination model (i.e., there is no single authority of metadata of the whole network). Instead, data is scattered around all participants of the IoS (clients, services, registries). Therefore, mechanisms need to be put in place which can be used to collect, consolidate and disseminate metadata. Because of the unusually large scale of the system this data dissemination protocol needs to be very efficient, and cannot impose a large burden on the participants of the network (e.g., speaking in terms of the model in Section 3 it is not possible to just send metadata about a new

service to every client of the travel planning system). Additionally, since many competing parties participate in the IoS (e.g., many taxis provide a taxi service) the question of trust arises – if a consumer receives metadata that states that a given taxi service underperforms (e.g., does not take the fastest route, or charges a too high price) she needs to make sure that this feedback actually came from a client of that taxi service, and not from a malicious competitor.

In the following section we will explain these research challenges in detail.

4.1 Requirements for Process Mining

Giving that the Internet of Service is becoming reality, its users will be increasingly interested in — and consequently looking for — specific *Web-mediated processes* (or *tasks*), each of which could cover a large spectrum of traditional activities. A typical example of those activities is the use case mentioned and explained in Section 3. Here, the workflow for travel planning (i.e., get a hotel, get a taxi to the hotel, order something to eat, etc.) clearly represents a Web-mediated process composed of different services available through the Internet.

This vision is strongly supported also by two expert persons, namely Peter Norvig from *Google* and Prabhakar Raghavan from *Yahoo!*.

During the DEMOFall08 Conference, there was a discussion panel on "Where the Web is Going"[2], in which Norvig and Raghavan basically agree that, rather than supporting only one search at a time, search engines will soon focus on helping people get a bigger task done.

A significative quote extracted from Raghavan's speech is the following: "*People intrinsically don't want to search. People don't come to work every day saying "I need to search"...* ***They want to run their lives!***".

So, there is an interesting shift here from a model where each search is independent to one where a search engine may expect searchers to do multiple searches when trying to accomplish their tasks.

Hence, a service on the Web represents the main building block and the atomic unit of a certain Web-mediated task. A Web-mediated task can be accomplished by one *atomic service* (i.e., the Web page for booking a hotel online) or it can be obtained through a specific *composition* of *orchestrated* and *mashed-up* atomic services, referred to as a *taskflow* (i.e., the complete travel plan). This term is used to describe a particular composition of atomic services in which, differently from classical *workflow*, services can be both software- and human-provided.

A software-provided service is a software component which performs some particular task according to the classical SOA view, and which is traditionally executed in a *software service container* by a *software service engine*. On the other hand, a human-provided service is completely performed by humans, that is its running phase is fully accomplished by humans.

As an example of a human-provided service, let's consider an Internet-based taxi service which accepts client reservations via Web as input, and provides taxicabs according to some policies. Such policies could take into account some logistic constraints like the area in which the taxi was requested (i.e., depending on the area the service could prefer to deviate a near free taxicab as well as send a new one directly from the headquarters). Moreover, the number of passengers who requested for a single taxicab should affect the choice made by the service, taking into account also the overall taxicabs availability. This clearly represents an example of a human-provided service because it is *reachable* and *callable* through the Internet, but it is also *enacted*, *executed* and *completed* by human efforts which, in this particular case, involve all the steps needed for deploying taxicabs (i.e., communicating via radio with taxicab drivers, driving the taxicabs, etc.).

Following this new approach, the Web can also be viewed as a potential *collection* of services and composition of those instead of a collection of "simple" documents, leading it towards the new IoS.

[2] http://blogoscoped.com/archive/2008-11-06-n63.html

Hence, the increasing number of services and composition of those needs scalable *discovery mechanisms*. An efficient service discovery infrastructure requires a new and enriched way for describing services: WSDL will not be enough because it describes only functional aspects of a service without taking into account other non-functional features. Those shortcomings could be overcome by refining the service descriptions making them "well-indexable" and adding information about QoE requirements, geographical constraints, price and contractual issues.

It is arguable that both atomic services and taskflows will be described in some *machine-readable* ways, following both *WS-** standards and the *RESTful* architectural style of Web services, or any other future and suitable specification.

In the World Wide Web which is constituted by billions of Web pages, the problem of contents discovery has been approached by applying classical *Information Retrieval* (IR) techniques to that domain (Web Information Retrieval or WIR)[16]. Typically, users try to gain their *information* (or *user*) *needs*, namely their intents, through Web search engines, which have become a sort of universally accepted *interface* to the whole information contained on the WWW.

On the other hand, the Internet of Services is constituted by billions of atomic or composed services growing rapidly, sharing the same "infrastructure" of the WWW, namely the Internet. So, there should be a new distributed and decentralized service discovery infrastructure suitable for managing the high number of indexed services, based on very powerful crawling and indexing techniques inherited by traditional WIR theory, which supports the retrieval of relevant services and taskflows in response to queries submitted by users that are trying to perform specific activities.

Of course, such goal requires solutions to manage research challenges. For example, novel mechanisms for extracting information about possible interactions between human and services on the Web must be developed. Those mechanisms could be based on *Web Usage Mining* techniques which analyze *query logs* and user *click-through data*[17, 18].

As an initial step, let us consider a recent work by Richardson [19] in which he analyzes interactions of users with a Web search engine in the long-term.

To date, most work on query logs analysis considers only short-term (within-session) query information. In contrast, this work shows that query effects are long-lasting and contain useful and valuable information concerning users behavior.

The results are encouraging, even if they are really preliminary, and show that interactions of users looking for services and tasks to be accomplished are already there on the current Web.

Moreover, new query and web resource[3] classification mechanisms using *traditional* or *fuzzy Machine Learning* approaches should be investigated [20].

In our vision, service discovery mechanisms in the Internet of Services will address three different aspects on three different levels of granularity:

- atomic services;
- taskflows;
- inferred taskflows.

This section will describe the requirements needed for discovering atomic services, while the next section will cover the requirements needed for discovering mashed-up services, that is taskflows.

Discovery of atomic services:

Atomic services are the building components of the Internet of Services. Searching for a specific service among a huge number of services should be the most important capability of the discovery mechanism, as searching for suitable Web pages is the main activity for a Web search engine. Using ad-hoc service descriptions, the service discovery infrastructure should identify the most suitable service for the users according to their needs. Services should be located by a Unique Service Identifier (USI) like an URI, a "geotagged" link, etc.

[3] this term is a *placeholder* which represents different kind of Web entities.

The service discovery engine will be able to build a query expressing user needs. Processing that query using state-of-the-art WIR techniques, the discovery infrastructure will retrieve the most appropriate atomic service. Following programming languages theory and layer-structuring systems design principles, atomic services will be primitive constructs (*policies*), which can be composed and organized in taskflows.

Discovery of taskflows: the discovery mechanism could be improved by enabling taskflows searching capabilities. A taskflow can be abstract or concrete: an abstract taskflow concerns the definition of an abstract generic activity composed by abstract atomic services, while concrete taskflows represent a concrete generic activity instance like a composition of concrete atomic services instances.

Based on her needs and experiences, a user will exploit the discovery infrastructure to search for a:

- concrete taskflow without being aware of the single "steps" that this taskflow is composed of;
- concrete taskflow being aware of the single "steps" that this taskflow is composed of.

In the first case, starting from a concept a user is looking for a concrete instance of a generic activity which is a composition of tasks a priori unknown by her. The discovery infrastructure will extract the corresponding abstract taskflow and it will retrieve a list of matching concrete taskflows according to specific matching criteria. Those instances represent historical concrete taskflows performed by other users. They also should be rated and ranked taking into account both the contextual information of the user who made the search and the feedback coming from the others, exploiting collective intelligence and social/collaborative aspects.

On the other hand, in the second case a user is searching for a concrete instance of a generic activity having in mind every single task by which it is composed, that is the corresponding abstract taskflow. Based on that abstract taskflow the discovery infrastructure will directly retrieve a list of candidate concrete taskflows following the same criteria described above. Furthermore, it should be remembered that taskflows entail the nature of time. Taking into account the different configurations of the taskflow graph at different timesteps might give some insights on service reputation and relevance.

Discovery of inferred taskflows: searching for a concrete taskflow will depend on the corresponding abstract taskflow, both if it has to be extracted by the discovery infrastructure or if it is directly provided by the user. If the wished *abstract taskflow* TF_i is more general than another abstract taskflow TF_j , that is TF_i is a subgraph of TF_j or even more formally TF_i subsumes TF_j , the discovery mechanism could return to the user also the instances of TF_j as the search result. Moreover, as a suggestion to the user the discovery infrastructure should also provide all the instances of concrete taskflows performed in relation to TF_i, taking into account both *contextual information* and *collective intelligence*.

That novel and smart service discovery mechanism should require a new kind of Web search engine, which we refer to as the *IoS Search Engine Framework*.

4.2 Requirements for Feedback-based Service Discovery

Both in traditional service-based systems and the Internet of Services, services can be ranked and compared using *Quality of Experience* (QoE) metadata. This is specifically true for services with large human involvement, i.e., services which are provided or consumed by humans. QoE is a measure of the overall level of customer satisfaction with a provider. QoE is related to but differs from *Quality of Service* (QoS), which embodies the notion that *hardware* and *software* characteristics can be measured, improved and perhaps guaranteed. In contrast, QoE expresses user satisfaction both objectively and subjectively. The QoE paradigm can be applied to any consumer-related business or service, hence it should be significantly suitable for the IoS.

To some extent, QoE is user-dependent because some customers are easier to please than others. The best QoE evaluations are obtained by polling or sampling a large number of users. Major factors that affect QoE include *cost*, *reliability*, *efficiency*, *privacy*, *security*, *interface user-friendliness* and *user confidence*.

QoE, while not always numerically quantifiable, is the most significant single factor in a real-world evaluation of the "user experience", giving meaningful feedback to the whole IoS users community.

Strongly related to QoE is the concept of "Service Level Agreements" (SLAs) [21]. A Service Level Agreement is a contract between a client and a provider of a given service and sets all the terms (mainly QoS attributes) that have to be respected during the service usage. A first draft of QoE knowledge in enterprise SOAs can be created and managed collecting positively expired SLAs. In business SOAs, where the negotiation context is quite fixed, users can collect terminated SLAs representing successful transactions sending them to a third party [22]. This trusted third party should manage those SLAs extracting negotiation data and using them in a ranking process aiming at discovering and ranking Service Providers. Hence, this approach gives the possibility to exploit historical data from successfully terminated transactions from users to evaluate service providers in advance.

Enabling such mechanisms in enterprise SOAs needs:

- to have a well defined set of data (not only QoS ones) representing a sort of QoE that can be used as basis for evaluations;
- to give the possibility to service clients to store and share expired SLAs with a third party that is responsible for ranking Service Providers;
- to define a third party able to use such QoE metadata in an efficient way;
- to define new metrics for ranking service providers;
- and to modify the general SOA negotiation process including the service providers evaluation step.

Moreover it is obvious that exploiting QoE metadata in the business SOA context is hard. A more interesting use of QoE could be done in the IoS context in which the negotiation phase is less narrow and "fixed" than in traditional service-based systems. Three main systems should affect and exploit the QoE metadata associated to a specific IoS service, that is:

- *Tagging systems:* a tag is a non-hierarchical keyword or term assigned to a piece of information (such as an Internet bookmark, digital image, or computer file). This kind of metadata helps describes an item and allows it to be found again by browsing or searching. Tags are chosen informally and personally by the item's creator or by its viewer, depending on the system. Coming back to the use case of Section 3, users could benefit and make their travel plan choices exploiting tags provided by other users. In that way, users are encouraged to do tagging because the collective intelligence will get improved. Tagging systems were popularized by websites associated with Web 2.0 and are an important feature of many Web 2.0 services, and we expect tagging to be gaining popularity in service-based systems, too. Moreover, tagging systems in the IoS allow to express new types of information, such as contextual and geographical information.
- *Recommender systems:* a recommender system forms a specific type of *Information Filtering* (IF) technique that attempts to present information items (movies, music, books, news, images, web pages) that are likely of interest to the user. Typically, a recommender system compares the user's profile to some reference characteristics. These characteristics may be from the information item (the content-based approach) or the user's social environment (the collaborative filtering approach). Recommendation will be very useful when humans will be involved in the execution of the activity. Still referring to the use case, it would be arguable for a user requesting some restaurants that the system takes into account her food preferences, basing on some kind of context knowledge. A recommendation will enable humans to exploit past knowledge on expired activities to optimize the current one.
- *Reputation systems:* a reputation system is a type of collaborative filtering system which attempts to determine ratings for a collection of entities, given a collection of opinions that those entities

hold about each other. This is similar to a recommendation system, but with the purpose of entities recommending each other, rather than some external set of entities (such as books, movies, or music). Reputation systems are often useful in large online communities in which users may frequently have the opportunity to interact with users with whom they have no prior experience or in communities where user generated content is posted like *YouTube* or *Flickr*. In such a situation, it is often helpful to base the decision whether or not to interact with that user on the prior experiences of other users (users feedback). Reputation systems may also be coupled with an incentive system to reward good behavior and punish bad behavior. For instance, users with high reputation may be granted special privileges, whereas users with low or unestablished reputation may have limited privileges. A simple reputation system, employed by *eBay*, is to record a rating (either positive, negative, or neutral) after each pair of users conducts a transaction. A user's reputation comprises the count of positive and negative transactions in that user's history. A similar approach could be applied also between any pairs of service consumer and service provider (i.e., the rating of a hotel, the quality of a restaurant, etc.).

4.3 Requirements for Service Metadata

For the Internet of Services a much richer metadata set is necessary than for traditional service-based systems. We summarize the categories of metadata important for services below:

- *Interface Information*: naturally, the functional service interface needs to be published using a machine-readable service description language, e.g., WSDL or WADL[23]. Our model is not based on the usage of a single description language, but basically any language understood by the requestors can be used. Note that it is not important that the service registry is able to interpret the interface description language used.
- *Intra-Service Protocol Information*: if a service is stateful, a machine-readable description of the legal sequences of invocations needs to be given. You can think of this protocol description as the "manual" that a requestor needs in order to accomplish certain tasks (e.g., in order to do "A" invoke operations "b" and "c"). However, note that we consider stateless services to be the dominating architectural style in the Internet of Services.
- *Service Semantics*: for every service, we need to know what it actually does. We may use Semantic Web Services (SWS) [24] technology, such as OWL-S[25], for this task.
- *Positional Data:* for mobile services, positional data is very important. Note that positional data is more than just the current GPS location – this data may also include information about where the provider was in the past, and where he is likely to go in the future. The challenge here is to figure out a good trade-off between a useful and still realistic model, which is still an open research topic.
- *Current Status*: for volatile services, registries also need to keep track of their current online status. As with positional data, this may be more than just a flag indicating whether the service is currently available, but contain data that allows for predictions about the status of a service at a point in time in the future.
- *Prerequisites:* some services may formulate restrictions on who or when a service can be used, e.g., "only customers of store A".
- *QoE*: this category of metadata includes feedback from service requestors, both explicit and implicit. See Section 4.2 for more details on QoE metadata.
- *Price*: the price of a service. This is a topic were lessons learned from work package JRA-1.3 on pricing and service level agreements should be taken into account.

- *User Tags*: basically, this category is a catch-all set of metadata for practically any type of requestor-provided metadata. Service clients may use tags to indicate the purpose of a service (in that sense user tags add information to the semantic data), the quality of a service ("super service", "didn't work", ...) or anything else a user may want to express. It is obvious that user tags are therefore rather orthogonal to the rest of the metadata set. Future research needs to be conducted in order to formulate algorithms and methods to mine useful information from this unstructured or semi-structured data.

Not all kinds of metadata are applicable for all types of services: protocol information, for example, is only important for stateful services. Some services and clients may exchange additional metadata, which does not fit in any of the above categories. For such cases the registry can accept various "extensibility metadata". The registry only stores and forwards such extensibility metadata, it does not interpret it in any way.

Some of the metadata above is mandatory for any service, and registries should reject to register services where such mandatory information is not provided and cannot be extracted reliably. Mandatory metadata includes interface and pricing information. Other metadata can be deemed mandatory based on the requirements of the registry. Some existing technologies may be used to specify metadata of more than one of the above categories at the same time: for instance, OWL-S may be used to specify service semantics, service interface, prerequisites and protocol information at the same time.

Traditional Web service registries such as UDDI [4] or ebXML [2] are built on a push notion: these registries are passive entities in the environment, which list only services that have been registered with them. The same notion is also followed by younger research approaches such as VRESCo[26, 27]. Active Web service registries have been proposed recently [28], however, these approaches are only "active" in the sense that they push new or updated query results actively to the user. Actively discovering new services, or monitoring the currently existing ones, is currently out of scope. We consider that a major problem for registries in the Internet of Services: given the mobile, transient and ad hoc nature of services in the Internet of Services we reckon that the next generation of service registries needs to do more.

For the NavInc. case study presented in Section 3 service metadata is needed for NavIncs. users to (1) find services based on expectations and requirements, (2) invoke previously unknown, dynamically discovered services, and (3) exchange information about service qualities, such as price, reliability or the subjective satisfaction with other users.

4.3.1 Active Service Tracking

An active registry for the Internet of Services needs to track services that are currently registered. This tracking includes all metadata as defined above. Tracking services can be done in three different ways: (1) actively tracking a service, e.g., by periodically polling the service provider about its current status and position, (2) passively tracking a service, e.g., by waiting for the service provider to update its metadata whenever he sees fit, or (3) on demand, e.g., by polling the service metadata prior to consumption or during query. The characteristics of these three types of service tracking are summarized in Table 1.

Style	Responsible	Timing of Metadata Update
Active	Registry	Periodically, according to a metadata update strategy
Passive	Provider	When metadata is changed
On Demand	Registry	When metadata is used

Table 4.1: Styles of Service Tracking

For active tracking, an update strategy needs to be defined. This strategy can be as simple as a polling frequency, however, more complex strategies are also possible (for instance, a service may change its

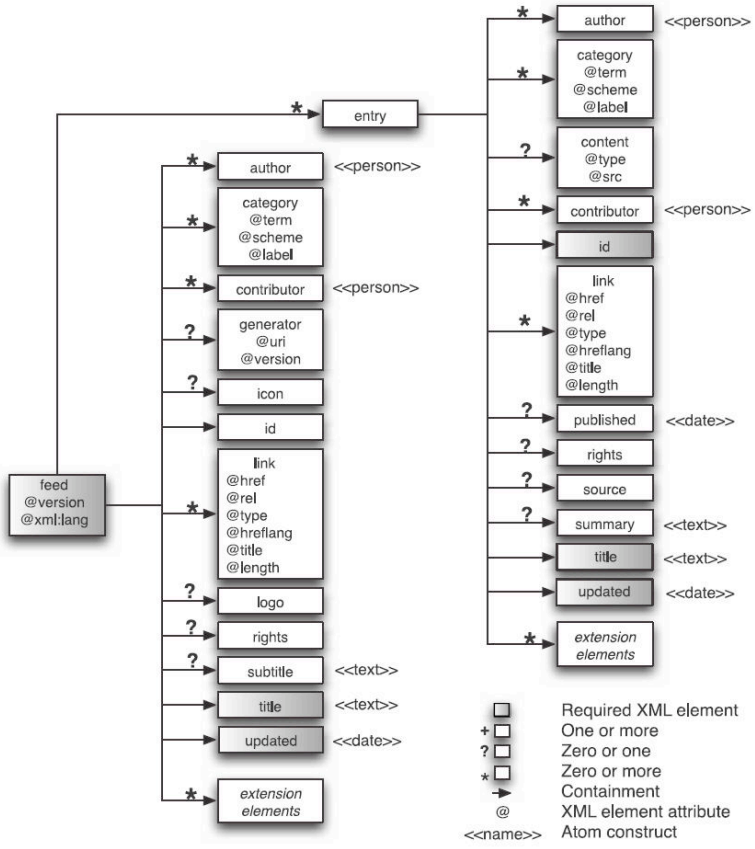

Figure 4.1: ATOM Publishing Format

location more often during day time than at night, therefore, metadata should be polled more frequently at day time). Service providers may suggest update strategies, however, registries may overrule this suggestion with a different update strategy', for instance, if the suggested strategy would overload the registry.

Technically, active and on demand tracking are implemented using service callbacks. That means that every service that wants to use these styles needs to implement the respective service functionality (*service feature* according to the notion introduced in [29]) to retrieve metadata. This is conceptionally similar to the ideas of WS-MetadataExchange[30].

We summarize active, passive and on demand tracking as reactive metadata tracking, because all these approaches involve updating the registry data after a change has happened. These approaches are valid for services with slow to medium metadata change rate. If metadata changes very frequently, e.g., the location of fast-moving mobile services, these reactive approaches are not sufficient. In such cases registries need to fall back to a more heuristic, proactive solution, which we refer to as predictive service tracking: instead of relying on exact metadata received from the service provider the registry tries to estimate values based on the last secured metadata and a model of how this concrete item of metadata for this service changes over time. Such a model can either be constructed from past information (e.g., using methods of statistics and data mining), or specified by the service provider (using again the same mechanisms as described above). The later approach is especially useful when a service provider already knows "in advance" how some of his metadata (most importantly, location and status) are going to change in the future.

The NavInc. case study exemplifies the need for service tracking: the NavInc. middleware allows users to publish "ad hoc" services, e.g., live coverage of a volcano eruption. These services have a very brief time to live, and need to be removed from the registry data when their provisioning stops.

4.3.2 Push-Style Client Notifications

After discovering changes the registry needs to actively inform the service clients about updates. This goes into the direction of the active Web service registry proposed in [28]. However, unlike the registry proposed there, we foresee a registry that pushes not only new service contracts towards the clients. In our vision all kinds of metadata changes can be published towards interested clients in order to enable "intelligent" service clients to autonomously decide on appropriate reactions. Conceptually, this is similar to the publish / subscribe communication paradigm, which is only sparsely used in Web service registries today [31]. As a possible communication protocol for this kind of active client notifications a number of alternatives are feasible: (1) from the services community the WS-Eventing [32] and WS-Notification [33] standards are available to implement active notifications using the "Solicit Response" message exchange pattern, while (2) the Atom protocol [34] was proposed as an XML publishing protocol in the Web 2.0 world. [28] used the Atom model to implement an active registry, because of the lower overhead, the alignment with REST architectural principles and the strong support in the Web 2.0 community. One additional advantage of ATOM is the improved scalability of the newsfeed model in comparison to the point-to-point notification model of WS-Notification.

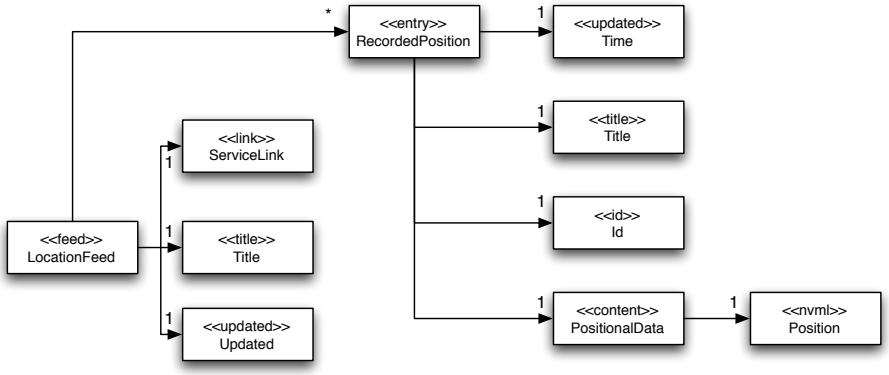

Figure 4.2: Positional Service Metadata in ATOM

In Figure 4.1 we sketch the general ATOM publishing model as defined in [34]. Registries using the ATOM publishing model for active client notification will need to map metadata as defined above into this general publishing framework, by defining (1) what an ATOM feed element represents (e.g., all services, all variants of a functionally similar service, the history of status changes of a service, ...), (2) what a an ATOM feed entry represents (e.g., a service, a datum such as the current position of a mobile service, ...) and (3) how to use the ATOM extension elements. One example of such a mapping is sketched in Figure 4.2. The figure shows the representation of the movements of a mobile service in the ATOM format. The actual position is encoded as NaVigation Markup Language (NVML) element [35]. An example instance of this feed is given in Listing 4.1.

4.3.3 Active Service Monitoring

When adding monitoring to an infrastructure usually a choice between intrusive and non-intrusive monitoring needs to be made. Both have their advantages and disadvantages. Service attached monitoring can provide the "freshest" data of its sensors as well as react the quickest to ordered adjustments. The problem of intrusive monitoring is that it can be performance blocking and hinders the heterogeneity and development of new services by forcing a predefined interface. Non-intrusive methods allow the services to be developed and evolve independently and monitoring can adapt to the services' changes. The previously mentioned VRESCo already supports non-intrusive QoS monitoring which can be used for sensing services and be adapted for future needs.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
 <title>Location Feed for Service X</title>
 <link href="http://vitalab.tuwien.ac.at/service/X"/>
 <updated>2008-12-01T18:30:02Z</updated>
 <entry>
   <title>Position 2008-12-01T18:30:02Z</title>
   <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
   <updated>2008-12-01T18:30:02Z</updated>
   <content type="http://www.w3.org/TR/NVML">
      <nvml>
        <head>
          <title>NVML Position</title>
        </head>
        <body>
          <navi>
            <point>
              <name>      Tokyo Station      </name>
              <category>  Station            </category>
              <latitude>  N35.40.39.0        </latitude>
              <longitude> E139.46.18.1       </longitude>
              <address>   Chiyoda-ku, Tokyo  </address>
              <zip-code>  123-4567           </zip-code>
            </point>
          </navi>
        </body>
      </nvml>
   </content>
 </entry>
 <entry>
   <title>Position 2008-12-01T18:15:04Z</title>
   <id>urn:uuid:1235c695-cfb8-4aaa-cccc-80da34666666</id>
   <updated>2008-12-01T18:15:04Z</updated>
 <content type="http://www.w3.org/TR/NVML">
      <nvml>
            ......
      </nvml>
   </content>
 </entry>
            ......
</feed>
```

Listing 4.1: Positional Metadata Example

Resource limited components and ones with enough resources coexist in the Internet of Services. This fact must be considered and respected when designing the monitoring facilities. The solution will be a basic light-weight but extensible and pluggable monitoring service. In this ever changing system, fluctuations and transient errors of any kind could cause not only unnecessary oscillation in registry entries but also between several configurations if constantly reported by the monitors to the evaluation/adaptation mechanism. Instead monitoring must contribute to the system's balance by reporting only relevant data.

Finally, the wast number of participating services suggests that only parts of the system can be controlled and adapted successfully. The same as with registries, no overall and central collection of monitored data fits this environment. Instead, sentinel services must report to monitoring proxies that keep a status overview of a minor section of the whole system.

4.4 Requirements for Dissemination Protocols

Another major issue with current registries is the strong centralization. This is due to the traditional idea of the Universal Business Registry, which has generally been depreciated by now. As described in Section 3, there will be no single registry, "oracle" or infallible guide because the IoS is a much more dynamic environment than the Web or today's service-oriented environments. The services hosted by providers

and their consumers will be joining and leaving the IoS non-deterministically and without announcement as software, server and network components fail or as mobile consumers go out of range of a network, resulting in the partitioning of providers and their services from their consumers. Thus, in this dynamic and loosely-coupled environment, maintaining and using registries becomes problematic for the following reasons:

- Given the absence of a single registry in the IoS, when a provider or new service joins this environment or the properties of a participant change, such as when a provider updates their system-wide policies, the availability of an individual service is modified or the consumer changes location, who should this "announcement" be made to? The service must be announced and registered somewhere, otherwise a consumer will never find the service (except possibly through a network broadcast, but these are often disabled by network administrators, and not feasible in a large-scale ecosystem).
- With no central registry of providers and services, how does a consumer discover the services with the properties that meet their requirements? Querying one registry is not enough as no single registry can know the entire set of providers and services. On the other hand it is also unfeasible and inefficient to expect the consumer to query every single registry.
- Because of the potential of an unannounced service or network failure, how do we discover if an entity is no longer part of the network? One method could be through polling services, but individually polling each service is inefficient, uneconomical and does not scale. As there is no central registry, how will the participant that discovers this information inform the other participants?

Fundamentally, in the illustrative example described in Section 3, the challenge of using decentralised registries for discovery is in solving the two processes of *announcement* and *search*. Both of these require *metadata dissemination* amongst registries, service providers and consumers. We first describe how both these of these processes rely on metadata dissemination then describe the high-level requirements for the dissemination mechanism.

- The announcement process is carried out as a provider, service or consumer joins the IoS or as the properties of a provider, service or consumer change. Because there is no central registry, there is no "one" registry to contact about these additions or changes and a registry already known by the provider, service or consumer may not be available. Thus, the information should be sent to one registry and disseminated to others so the provider, service or consumer can be discovered later through a search on any registry.
- The search for a required service is performed by a potential consumer. Again, because of the decentralised nature of the IoS, there is no central registry to "ask". Therefore the search, itself a piece of metadata, must be spread (disseminated) through the registries until a match is found and returned to the entity performing the search.

Thus, both announcement and search using decentralised registries containing provider, service and consumer information can be solved using a metadata dissemination mechanism. The high-level requirements for the metadata dissemination mechanism are:

- The metadata dissemination mechanism must be decentralised, i.e., not dependent on one entity to collect the announcements or answer searches. Thus, there must be no single points of failure in the mechanism so information on the Internet of Services is always available. Meeting this requirement will ensure adaptation of a business process through the discovery of other services is always possible.

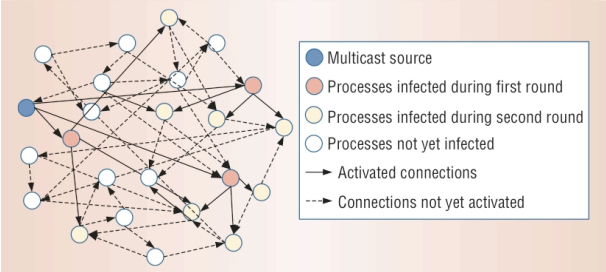

Figure 4.3: Gossip-Style Data Dissemination, from [37]

- Closely related to the decentralised aspect, the metadata dissemination must be *robust* so when a failure occurs (such as a registry going off-line due to a network outage or a service crash) there is no impact on the announcement and search functions to, again, ensure that adaptation can always be made.
- As described in Section 2, the number of participants in this ecosystem will consist of millions of services with many more consumers. Therefore the metadata dissemination process must be scalable to a "web-scale" of services.
- The mechanism must not be a burden to any one registry. Thus, any registry participating in the decentralised search must have a bounded load so that even if it is the only entity participating in the announcement and search, it will not be overwhelmed by new information and search requests.

By meeting these requirements and providing a robust, decentralised and scalable mechanism for metadata dissemination we can provide a novel method for providing decentralised registries in the potentially vast IoS. As was mentioned above, the principles underlying gossip-style protocols (also known as *epidemic* protocols) may meet the criteria we have outlined. For example, as described in [36], this class of protocols have been shown to scale and perform well whilst being reliable when components fail. Protocols based on these techniques operate by exchanging a limited amount of state with other randomly-selected peers (actors in the IoS in that case). Any "new" metadata received when gossiping is updated and the process repeated continuously.

Eugster et al. describe the general principal of gossip-style communication as follows: "A process that wishes to disseminate a new piece of information to the system does not send it to a server, or a cluster of servers, in charge of forwarding it, but rather to a set of other peer processes, chosen at random" [37]. This variant of random walking is visualized in Figure 4.3. This strategy is proactive, in that it is able to circumvent temporary or permanent link outages or process failures automatically, without the need for explicit reconfiguration. Another advantage is that the load that is imposed on every process in the network is constant.

These protocols are founded on mathematical principles they "lend themselves to theoretical analysis, making it possible to predict their behaviour with high confidence" [36], an advantage when wishing to prove to a new provider or consumer that there will be "no surprises" when participating in the IoS. Therefore, we conclude that gossip-style data dissemination may be a good starting point to construct a dynamic and distributed service registry for the IoS.

Note that this distributed registry is significantly different than existing research work carried on P2P-based registries, such as [38, 39, 40, 41]. Those approaches are physically distributed, however, they still represent "one logical entity" (the federated registry). In the IoS registry data is scattered over many physically and logically separated entities, with different concerns, goals and heterogeneous hard- and software. The data dissemination framework for such an environment therefore needs to deal with untrusted, incomplete and probably maliciously wrong information on a large scale, which has to the best of our knowledge not been considered so far.

Chapter 5

Evaluation of Current Registries

The main task of Web service registries is to act as repository for metadata concerning Web services. During the discovery process, Web service registries play a crucial role since a registry is the central entity for the discovery of Web services. In the last years, industry driven approaches like UDDI [1] and ebXML [2] strived to support the discovery process of Web services with the introduction of public, centralized and later federated registries. The goal of these approaches was to create the foundation of Web service marketplaces where end users are able to search for arbitrary services that fulfilled their requirements. On the other hand, Web service providers were given the opportunity to publish their services at a well known location which potentially acts as global advertising platform. However, these approaches were not well adopted by the service community at all. This resulted in the abandoning of public UDDI registries by IBM and Microsoft [42] which basically used as public testbed and contained a large number of non-working service descriptions. In academia, there was a trend towards semantic enriched Web service registries that partially build upon UDDI concepts and tried to integrate semantic information [43]. In the following section we will briefly discuss current Web service registries, with special emphasis on the requirements we have discussed earlier. Note that a full survey of existing work in Web service registries is out of scope here (in that respect, a more detailed analysis has been provided in PO-JRA-2.3.1, and also in [3]).

In the following subsections, we analyze the current state of the art of Web service registries with regard to requirements of the future Internet of Services. The list below provides examples of registries belonging to different categories. We consider (1) Web portal based approaches which support users in the discovery of Web services, (2) Semantic approaches that provide semantic metadata to support automated service discovery, (3) Industry Standard approaches driven by industry, and (4) Academia approaches of Web service registries. Notice that semantic approaches are usually prototypes in academia. However, we also also discuss ”non-semantic” approaches from academia and thus differentiate between these two approaches.

1. Seekda: Is a public Web portal that provides Web service related information and supports keyword based service discovery [44].
2. Strikeiron: Is a public Web portal that hosts Web service related information [45].
3. Meteor-S: Is an academic prototype that provides semantic Web service descriptions in a peer to peer registry [46].
4. VRESCo: Is an academic prototype Web service registry framework that supports complex event processing [26, 27, 31].
5. AWSR: Is a lightweight Atom Feed [34] based Web service registry implementation [28].
6. UDDI: Is the industry driven approach to store Web service information in centralized repositories [1].

7. ebXML: The OASIS registry working group defined a registry standard that is able to store any kind of metadata concerning Web services [2].

5.1 Metadata Support

Registries need to provide metadata of the registered services to support the discovery process.The hierarchical UDDI data model divides information into technical aspects and human readable descriptions. While this approach seems to be consequent in separating human readable service descriptions from technical descriptions, it turned out to be poorly supported by the implementation of the data model, the API respectively. The ebXML approach is broader in scope and provided a meta model to store any service relevant data. However, the data model is very generic and there are different ways to store the same information. This makes a general retrieval policy infeasible, because the information may be represented by different data structures. To overcome this problem, best practices were proposed by the OASIS committee as guidelines. Semantic approaches like Meteor-S integrate knowledge in form of ontologies and attach this kind of information to Web service descriptions to support a (semi-)automated discovery process. The VRESCo approach uses a "light-weight" semantic model [29], which is used to map concrete Web services to company-internal domain models, e.g., to define what functionality a given internal or external service provides in the context of a given business. Portal based approaches like Strikeiron and Seekda offer a set of human readable service information with the focus on technical issues. They provide general descriptions of the service, describe how to use a service and example input/output data for developers to integrate and inform about service pricing. Strikeiron furthermore offers sample code in different languages (e.g., PHP, Ruby, etc.) to illustrate service invocations. Furthermore, user tags can be used to attach service related information to registered services. AWSR integrates arbitrary service related information in a Atom based data model. Similar to ebXML, AWSR is able to represent all types of information and attach metadata to it. Unlike ebXML, AWSR structures metadata in separate feeds that represent different types of metadata.

5.2 Query Support and Query expressiveness

In order to find relevant service information, registries provide different means to discover Web services using metadata. UDDI provides a keyword search that allows users to define keywords and to query the registry for services which descriptions contain these keywords. The ebXML registry is more powerful in terms of expressiveness. It provides an extensible filtering mechanism, which allows for efficient retrieval of services. However, no explicit query language is defined, instead inline SQL can be used. Strikeiron and Seekda provide a keyword based approach for the discovery of services and support tag based discovery of Web services. AWSR offers a XQuery interface to search for Web service related information and supports tag based search for registered services.

5.3 Scalability

Centralized registry architectures have a natural single point of failure. To overcome limitations of centralized registries, federation approaches were introduced to centralized models, such as UDDI and ebXML. UDDI's federation approach allowed for the replication of registry content on distributed registries that follows pre defined replication protocols. Meteor distributes registry content on a peer to peer network. The distributed content is managed by a gateway peer that is required upon service registration. Discovery services work without the gateway peer and distribute queries among all registry peers. VRESCo currently does not consider registry federation, and therefore also provides a single point of failure. AWSR uses a Atom feed based approach to integrate distributed AWSR registries into a common registry.

5.4 Accuracy of Registry Content

As noted before, public Web service registries were often used as a testbed and thus contained a large number of non working services. Therefore, the content was often inaccurate and made it difficult to discover production services. Furthermore, since the registries were designed to be passive, they did not support extensive notification mechanisms that are required to notify clients of service related changes. UDDI and ebXML based registry approaches did not support notification mechanisms in their first implementations and added rudimentary notification support in later versions. VRESCo and AWSR explicitly support the notification of changes of registry content. Vresco supports complex eventing and allows to register for complex events. AWSR offers a lightweight approach and provides the means to register different changing aspects of Web services such as interface changes, QoS changes, and so on. Seekda offers statistical information about the availability of services and thus provides information about the current state of the service.

5.5 Support for Service Tracking

Closely related to accuracy is the support for dynamic services, i.e., services that are of volatile nature in terms of their availability. Therefore, notification mechanisms are required to inform interested parties about service changes. However, in the current state, registries do not offer mechanisms that support service tracking services directly and rely on their build in notification mechanisms to provide rudimentary service tracking support. VRESCo and AWSR provide the means for notification when services change their availability. AWSR provides a basic publish/subscribe approach based on Atom feeds. VRESCo provides extended support for the handling of complex events concerning registered services. UDDI and ebXML provide a notification mechanism to register to arbitrary registry events, such as registration, un-registration of services.

5.6 Support for Versioning

As Web services change during their life cycle, registries are required to support different versions of Web services, because of compatibility issues. From all these approaches only VRESCo offers explicit support for versioning of services.

5.7 Support for Mashups

Current registry approaches do not provide explicit support for the discovery of mashups or, therefore, for fragments of business processes.

5.8 Support for Service Ranking

The registry approaches that are discussed above, do not offer service ranking. They rely on the expressiveness of the supported query languages that can be used to generate rankings within the search result. However, this approach is very difficult from the a end user's perspective, because this requires explicit knowledge of the data model.

5.9 Discussion

The slow adoption of public Web service registries leads to a situation where only a small number of working Web services were published in public available registries. These public repositories contained

several hundreds of references to Web services which where either test services or which were discontinued. Another major reason for the failure of public Web service registries was their inability to provide meaningful metadata in a format that can easily be accessed by humans as well as machines [47] during the discovery process. Keyword based approaches provide limited precision if applied to all registry content. We summarize features of the aforementioned registries in Table 5.1.

Requirement	UDDI	ebXML	Meteor	Vresco	AWSR	Strikeiron	Seekda
Metadata Support	+	+	+	+	+	+	+
Query Support	-	-	+	+	+	-	-
Scalability	+	-	+	+	+	-	-
Accuracy	-	-	-	+	+	-	-
Service Tracking	-	-	-	-	+	-	-
Versioning	-	-	-	+	-	-	-
Mashups	-	-	-	-	-	-	-
Ranking	-	-	-	-	-	-	-

Table 5.1: Summary of registry features

Chapter 6

Conclusions

In this document we have sketched the S-Cube vision of the Internet of Services, an Internet-scale dynamic ecosystem of services. The ecosystem is characterized as the unification of Web 2.0, SOA, Semantic Web Services and Mobile Computing, and has characteristics of each of these areas. The Internet of Services is characterized by ad hoc service provisioning and consumption, human provided services, service mobility and volatility, the relevance of subjective quality metrics (Quality of Experience), and, most importantly, by a tremendous number of service instances. We assume that service distribution in the IoS will follow a power law distribution, just like web pages in the traditional web. Because of the ad hoc nature of the Internet of Services we assume that services in this ecosystem will generally follow the REST architectural style. We presented a case study from the car manufacturing domain, which illustrated a typical business scenario in the Internet of Services. In this case study, a GPS unit manufacturer provides a service for travel planning, which makes use of the capabilities of the Internet of Services. Additionally, the company provides means for users to actively contribute by providing ad hoc Web services through their GPS units.

In this document we summarized the research challenges presented in five research directions: (1) requirements for implicit business process and service discovery refer to the obvious problem of scaling current service discovery mechanisms up to Internet scale and the challenge of discovering usage patterns and ad hoc service compositions in a mobile and dynamic service environment, (2) requirements for service ranking consider the evaluation of a subjective quality measure for services, the Quality of Experience, (3) requirements for service metadata describe the challenges which revolve around the necessity of a powerful and extensible metadata model for the Internet of Services, (4) requirements for large-scale data dissemination consider the decentralized nature of the Internet of Services, and the lack of a central authority therein, and discuss the need to disseminate data in a decentralized way, and (5) requirements for monitoring and tracking refer to the challenges associated with active Web service registries, which need to actively track metadata about services in a dynamic environment, and actively push this information towards clients (instead of passively responding to client requests) .

Finally, we briefly evaluated current Web service registries with regard to the requirements presented in this document, and have shown how they are not able to satisfyingly deal with the challenges presented in this document. Therefore, research will be conducted within the remainder of the S-Cube project in all of these areas to find solutions to the challenges presented here, and to develop an integrated and active registry solution for large-scale dynamic service environments, which can form the basis of research within the other connected work packages of the S-Cube project.

Bibliography

[1] UDDI.org, "UDDI Technical White Paper," http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, 2000, visited: 2007-07-31. [Online]. Available: \url{http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf}

[2] O. for the Advancement of Structured Information Standards (OASIS), "OASIS/ebXML Registry Services Specification v2.0," http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf, 2002. [Online]. Available: http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf

[3] S. Dustdar and M. Treiber, "A view based analysis on web service registries," *Distributed Parallel Databases*, vol. 18, no. 2, pp. 147–171, 2005.

[4] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, no. 2, 2002.

[5] C. Schroth and T. Janner, "Web 2.0 and soa: Converging concepts enabling the internet of services," *IT Professional*, vol. 9, no. 3, pp. 36–41, 2007.

[6] A. Dubey and D. Wagel, "Delivering software as a service," *The McKinsey Quarterly*, 2007.

[7] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[8] D. Schall, H.-L. Truong, and S. Dustdar, "Unifying human and software services in web-scale collaborations," *IEEE Internet Computing*, vol. 12, no. 3, pp. 62–68, 2008.

[9] *WS-BPEL4People*, IBM, 2007. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/

[10] T. Holmes, M. Vasko, and S. Dustdar, "Viebop: Extending bpel engines with bpel4people," in *PDP '08: Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 547–555.

[11] D. A. Menascé, "Qos issues in web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 225–236, 2002.

[12] A. van Moorsel, "Metrics for the Internet Age: Quality of Experience and Quality of Business," HP Labs, Tech. Rep., 2001.

[13] L. Amaral, A. Scala, M. Barthelemy, and H. Stanley, "Classes of small-world networks," in *Proceedings of the National Academy of Sciences*, 2000.

[14] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000, pp. 163–170.

[15] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: making the right architectural decision," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, NY, USA: ACM, 2008, pp. 805–814.

[16] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley, May 1999. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/020139829X

[17] S. Chakrabarti, *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002. [Online]. Available: http://www.cse.iitb.ac.in/~soumen/mining-the-web/

[18] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer, January 2007. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/3540378812

[19] M. Richardson, "Learning about the world through long-term query logs," *ACM Trans. Web*, vol. 2, no. 4, pp. 1–27, 2008.

[20] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, March 1997. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0070428077

[21] J. Skene, D. D. Lamanna, and W. Emmerich, "Precise service level agreements," in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 179–188.

[22] D. Laforenza, F. M. Nardini, and F. Silvestri, "Collaborative ranking of grid-enabled workflow service providers," in *HPDC*, M. Parashar, K. Schwan, J. B. Weissman, and D. Laforenza, Eds. ACM, 2008, pp. 227–228.

[23] M. J. Hadley, "Web Application Description Language (WADL)," https://wadl.dev.java.net/wadl20061109.pdf, 2006, visited: 2008-09-03. [Online]. Available: https://wadl.dev.java.net/wadl20061109.pdf

[24] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001.

[25] 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," World Wide Web Consortium, 2004.

[26] A. Michlmayr, F. Rosenberg, C. Platzer, and S. Dustar, "Towards Recovering the Broken SOA Triangle – A Software Engineering Perspective," in *Proceedings of the International Workshop on Service Oriented Software Engineering (IW-SOSE'07)*, 2007.

[27] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "End-to-End Versioning Support for Web Services," in *Proceedings of the International Conference on Services Computing (SCC 2008)*. IEEE Computer Society, July 2008.

[28] M. Treiber and S. Dustdar, "Active web service registries," *IEEE Internet Computing*, vol. 11, no. 5, pp. 66–71, 2007.

[29] F. Rosenberg, P. Leitner, A. Michlmayr, and S. Dustdar, "Integrated Metadata Support for Web Service Runtimes," in *Proceedings of the Middleware for Web Services Workshop (MWS'08), co-located with the 12th IEEE International EDOC Conference*, September 2008.

[30] *Web Services Metadata Exchange (WS-MetadataExchange)*, 2004. [Online]. Available: {http://xml.coverpages.org/WS-MetadataExchange.pdf}

[31] Anton Michlmayr and Florian Rosenberg and Philipp Leitner and Schahram Dustdar, "Advanced Event Processing and Notifications in Service Runtime Environments," in *Proceedings of the 2nd International Conference on Distributed Event-Based Systems (DEBS'08)*, 2008.

[32] *Web Services Eventing (WS-Eventing)*, W3C, 2006, http://www.w3.org/Submission/WS-Eventing/.

[33] *Web Services Notification (WS-Notification)*, OASIS International Standards Consortium, 2006, http://oasis-open.org/committees/wsn/.

[34] *The Atom Syndication Format*, IETF, 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4287.txt

[35] *NaVigation Markup Language (NVML)*, World Wide Web Consortium (W3C), 1999. [Online]. Available: http://www.w3.org/TR/NVML

[36] W. Vogels, "Web Services Are Not Distributed Objects," *IEEE Internet Computing*, vol. 7, no. 6, 2003.

[37] P. Eugste, R. Guerraoui, A. M. Kermarrec, and L. Massoulie, "From Epidemics to Distributed Computing," *IEEE Computer*, vol. 37, no. 5, pp. 60–67, May 2004.

[38] K. Sivashanmugam, K. Verma, and A. Sheth, "Discovery of web services in a federated registry environment," *Web Services, 2004. Proceedings. IEEE International Conference on*, pp. 270–278, July 2004.

[39] S. Dustdar and M. Treiber, "Integration of transient web services into a virtual peer to peer web service registry," *Distributed Parallel Databases*, vol. 20, no. 2, pp. 91–115, 2006.

[40] C. Schmidt and M. Parashar, "A peer-to-peer approach to web service discovery," *World Wide Web*, vol. 7, no. 2, pp. 211–229, 2004.

[41] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services," *Inf. Technol. and Management*, vol. 6, no. 1, pp. 17–39, 2005.

[42] Microsoft, 2005. [Online]. Available: http://uddi.microsoft.com/about/FAQshutdown.htm

[43] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara, "Importing the semantic web in uddi," in *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*.

[44] "Seekda," 2008. [Online]. Available: http://seekda.com/

[45] "Strikeiron," 2008. [Online]. Available: http://strikeiron.com/

[46] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services," *Journal of Information Technology and Management*, vol. 6, p. 2005, 2005.

[47] S. Dustdar and S. Haslinger, "Testing of service oriented architecture - a practical approach," Tech. Rep., 2004.