An Extensible Discovery Service for Smart Things

Simon Mayer
Inst. for Pervasive Computing
ETH Zurich
Zurich, Switzerland
mayersi@inf.ethz.ch

Dominique Guinard
Inst. for Pervasive Computing
ETH Zurich
Zurich, Switzerland
dguinard@inf.ethz.ch

ABSTRACT

We present DiscoWoT, a semantic discovery service for Web-enabled smart things. The service is based on the application of multiple *Discovery Strategies* to a Web resource's representation, where arbitrary users can create and update strategies at runtime using DiscoWoT's RESTful interface. Its goal is to provide a future-proof mechanism for enabling both, human users and machines, to semantically discover functionality provided by Web-enabled devices. Ultimately, it aims to allow for the facilitated discovery, selection, and utilization of smart things. DiscoWoT incorporates a transparent mechanism for deferring resource discovery to external handlers and can thus interact with other services within discovery service federations. It may be accessed by arbitrary users for ad hoc discovery of functionality offered by Web resources or incorporated into infrastructures for Web-enabled smart things.

Keywords

Semantic Discovery, Smart Things, Web of Things

1. INTRODUCTION

In the Web of Things vision, smart things are interconnected using Web patterns (e.g., REST) and protocols (e.g., HTTP) [4]. When this vision will be implemented to its full extend, millions or even billions of smart things will be available and linked on the Web. As a consequence, it is expected that it will become increasingly difficult for computers as well as human users to find, select, and use smart things in a fast, reliable, and user-friendly way. The task of finding relevant smart things is significantly more complicated than searching for documents, not only because smart things should be identified according to dynamic, contextual information, but also due to the lack of a uniform way of describing the things, their properties, and the services they offer: A smart thing does not necessarily express its functionality such that it may be found by traditional search engines. Another issue when considering the discovery of smart things is that, for these, we require a mechanism that allows *machines* to discover them and understand their capabilities in order to enable automatic usage by software applications. Possibilities that would be enabled by machine-understandable semantic descriptions for smart things range from on-the-fly user interface creation based on the specific capabilities of the accessed device to enabling machines to discover required services themselves or support users in finding services within densely populated smart environments. The basis for such applications is a generic mechanism that allows smart devices to provide semantic descriptions of the services they offer.

The problem of describing resources on the Web is not inherent only to smart things, but rather is a general problem of providing service specifications that has been worked on mainly in the Semantic Web domain. To describe Web resources, multiple languages – such as the Resource Description Framework (RDF)[1] or Microformats[2] – have already been proposed or are currently in their development process, like Microdata. From the past, however, we learn that we most probably will not be able to create a uniform description scheme for heterogeneous devices and to position that system as the predominant semantic framework for smart things. As we consequently believe that there will be no single "best" way of describing a thing on the Web in a way suitable for both, humans and machines, we have developed an extensible discovery mechanism that incorporates multiple discovery strategies to semantically map Web resources and allows users to extend the pool of available strategies at runtime. In this paper, after a review of related work in the domain of semantic description schemes, we describe our extensible discovery system.

2. RELATED WORK

Semantic technologies that enable machines and human users to understand data are one promising solution for a discovery service in the Web of Things. They can facilitate or even automate the tasks of composing device functionality within the Internet of Things to yield new services [5]. A prominent approach to put forward common formats for the integration and combining of data and its relationship to real world objects is the Semantic Web [1], the most advanced concretization of which is based upon RDF. Critics, however, question its feasibility as a whole [7].

Approaches towards enriching HTML documents with semantic metadata most notably include Microformats that

[1] www.w3.org/RDF/
[2] www.microformats.org

WoT 2011, June 2011; San Francisco, CA, USA

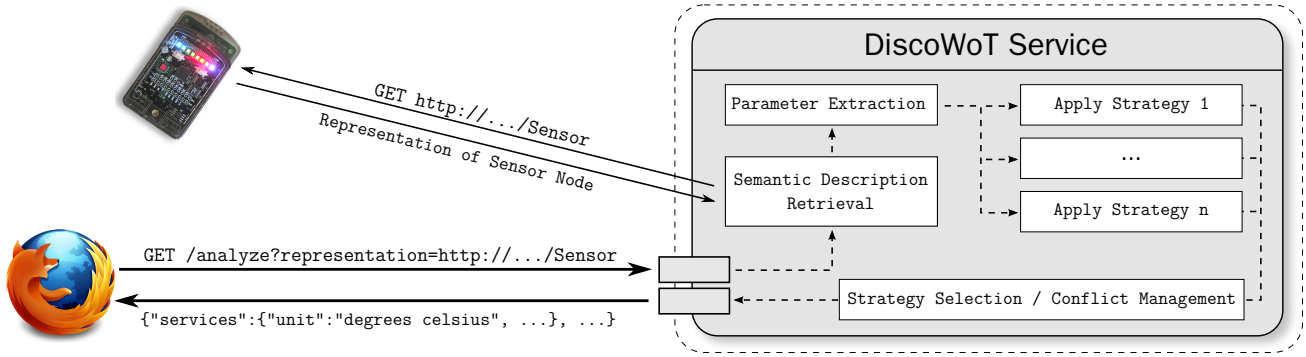

Figure 1: Client interaction with the DiscoWoT service. A client sends an HTTP GET request to trigger the semantic discovery of a resource. Multiple Discovery Strategies are applied to the representation retrieved from the resource's URL. The resulting resource description is returned to the client.

aim at re-using existing tags to attach semantics to data on the Web. The proliferation of Microformats is currently gaining momentum as several companies, including Google, Yahoo, and Microsoft have started to use Microformats in their products or expressed their intention to do so. Another related standard that targets specifically the modeling of sensors and sensor systems is SensorML[3]. In this work, we leverage this body of research while not enforcing one particular format over another but rather providing support for a broad spectrum of popular semantic description languages.

Middleware solutions that leverage the description languages for facilitating the management and interconnection of smart things have also been proposed and discussed. For instance in [5], where a central argument for a discovery tool based on semantic technologies is that these would not only facilitate the discovery but also the behavioral control of heterogeneous components. Similarly, in [3], the authors present a discovery, querying, and selection framework for WS-* and RESTful web services where resources expose their APIs using machine-readable formats. The framework supports either Microformat-based markup or WSDL documents and federates them in a meta-description format that is accessible to external clients through a (WS-*) resource discovery service. Furthermore, they propose a multi-layer querying mechanism that incorporates a query augmentation system and makes use of multiple query strategies. An extensive survey of sensor-actuator networks along with a resource repository implementation is presented in [9]. This repository makes Web-enabled things discoverable using a tag-based approach while publishing an OpenSearch[4] document to describe its resource retrieval capabilities. With DiscoWoT, we build upon these approaches but focus on providing a dynamic extension mechanism that allows clients to inject new resource discovery strategies. The success of community-driven platforms such as del.icio.us[5] and Wikipedia[6] alongside with recent studies [6] illustrate how crowdsourcing approaches (i.e., approaches that allow clients to contribute their needs and knowledge back to the service) can be leveraged to build globally successful knowledge bases.

[3] www.opengeospatial.org/standards/sensorml
[4] www.opensearch.org
[5] del.icio.us
[6] www.wikipedia.org

3. AN EXTENSIBLE DISCOVERY SERVICE FOR SMART THINGS

We propose DiscoWoT, a semantic discovery service for Web-enabled resources that relies on the application of *multiple* mapping schemes, which we call "Discovery Strategies", to semantically identify resources whose network addresses are known. The service may be used as an on-the-fly translation service for semantic information provided by resources a client wishes to interact with. To provide this functionality, any reference to or representation of a resource (e.g., its URL or corresponding JSON, HTML, or XML document) submitted by a client is analyzed using the registered strategies (cf. Figure 1). The extracted information is used to create an internal representation of the entity that contains structured data on the resource, its properties, and the functionality it provides. This information is transmitted to the client.

We have chosen to use a strategy-based mechanism as, in our view, any system that provides semantic discovery for smart devices has to be designed in an extensible fashion at multiple levels for achieving sustainable success: It is imperative that the Web integration and the evolvability of semantic recognition mechanisms for new smart things is not inhibited in any way. As the biggest weaknesses of any discovery service lie with its inability to achieve comprehensiveness and sustain it with respect to future description languages, our architecture is designed to allow for the injection of new discovery strategies during runtime of the system. It thus enables developers and users to create and submit new methods of semantically describing Web resources on demand: Whenever a new Web-enabled device is created or a service is launched that does not follow the specifications of a provisioned discovery strategy, a new strategy can be submitted to DiscoWoT to extend its scope to the new resource. This makes the new resource (and similarly described resources) discoverable and directly useable by all clients using the service.

The concrete outcome of the DiscoWoT project is a prototype Web service that clients may query to semantically identify smart things. The service has been created using the AutoWoT toolkit [8] and is thus based on RESTful principles. An entity (e.g., a manufacturer of Web-enabled devices) interested in rendering a device's semantic description accessible through DiscoWoT may submit a new strategy by

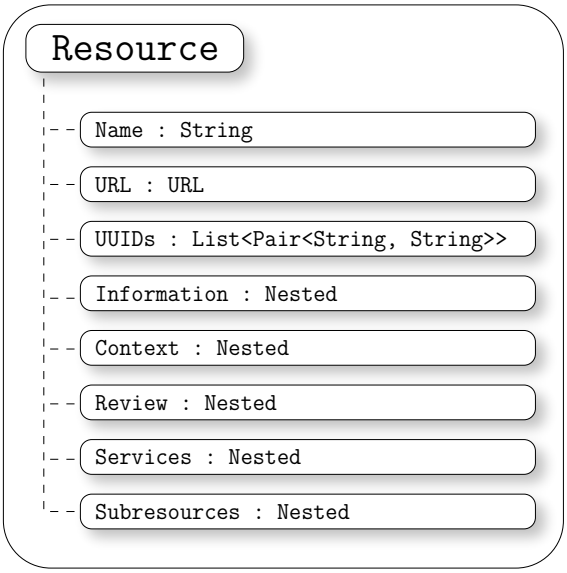

Figure 2: The structure of resource descriptions as held internally by the service.

issuing an HTTP `POST` request that contains a description of the strategy. DiscoWoT processes the request, uses the information therein to create a new strategy, and responds with the URL corresponding to the injected strategy. From this point onwards, DiscoWoT is capable of translating semantic descriptions of devices described in the same way to its internal resource description format. To semantically identify a smart thing of that very type, a client may now submit an HTTP `GET` request that includes a representation of the smart thing to be discovered (e.g., its URL) to the `/analyze` endpoint of the service. DiscoWoT then performs an analysis of the resource's representation and answers with a description of the Web resource as a JSON or XML document where the client may select the desired format by setting the HTTP *accept* header when contacting the service. The client may now use the information contained within DiscoWoT's response to interact with the discovered device.

3.1 Internal Representation of Web Resources

The way of representing a discovered resource internally before passing its description to a client is a central component of the DiscoWoT architecture that we have structured in accordance with a compound Microformat description for Web-enabled things proposed in [4] (cf. Figure 2). This format initially contains basic information on the resource (i.e., name, URL, UUIDs) as well as metadata relating to the resource's context (e.g., its geographical location, postal location, and hierarchical location[7]) and concrete product information relating to the resource (brand, category, photo, etc.). Additionally, the format includes information on reviews or ratings that a resource may have received and, finally, descriptions of the services that the resource offers and of attached sub-resources.

The internal representation format possibly represents a major bottleneck concerning the extensibility of the service: Even if DiscoWoT is able to incorporate every future way of describing resource properties, these have to be mapped to the internal meta-format of resource descriptions at some point. To not constrain the expressiveness of injected strategies, this format has to be as expressive as any future semantic description format. Since this property cannot be guaranteed using a static approach, we propose to not only allow the dynamic injection of discovery strategies but to furthermore make DiscoWoT's internal resource description dynamic.

3.2 Community-driven Strategy Creation and Extension

DiscoWoT features a RESTful interface not only for allowing clients to resolve resource representations to semantic descriptions, but also for enabling them to *inject new ways of semantically resolving such representations* into the service. By permitting arbitrary users to create new discovery strategies and to revise existing ones, we hope to create the basis of a community effort that will help to arrive at a comprehensive smart things discovery system: When an individual or a company wants to use DiscoWoT for discovering a smart thing that is semantically described using any existing or future language, it should submit a corresponding discovery strategy to the service, thereby also rendering the thing discoverable for all other users. This approach will allow for DiscoWoT to remain up-to-date concerning novel and emerging description schemes and semantic markup and thus represent a future-proof mechanism for semantically identifying resources.

To better discuss about the strategy injection mechanism in the following, we introduce the notions of "Discovery Strategies", "Strategy Stubs", "Strategy Schemes", and "Strategy Mappings" in a more formal manner: We call the framework responsible for understanding a specific semantic description language or markup (e.g., Microformats) a *Strategy Stub*. Stubs specifically include the parsing of a resource's representation. The current version of DiscoWoT includes such stubs for resource descriptions based on JSON, RDFa, Microformats, and Microdata. Depending on whether a schema-driven or schema-less approach (cf. Section 3.2.1) is adopted for the given format, a strategy stub is either extended using strategy schemes with associated strategy mappings or by attaching strategy mappings directly to the stub. *Strategy Scheme*, here, denotes structural information associated with a Web resource's semantic description (e.g., JSON Schema[8] for JSON-based descriptions) while *Strategy Mapping* refers to the purely syntactical *mapping* of names in the resource's representation to identifiers internal to DiscoWoT. Finally, a *Discovery Strategy* is a composition made of a strategy stub with an optional strategy scheme and an attached strategy mapping.

We explicitly distinguish between the community-driven *integration* of strategy stubs and the *extension* of stubs to create new discovery strategies. In the following, we describe both mechanisms and DiscoWoT's `/strategies` endpoint, which is responsible for handling them. As an example for the strategy injection process, we use a simple JSON[9]-based representation of a Web-enabled temperature sensor throughout this section (cf. Listing 1).

[7]We model the location of a device also as a hierarchical logical place name, e.g., `/ethz/buildingCNB/floorH/room108`

[8]`json-schema.org/`

[9]`www.json.org`

```
{
  "name":"Temperature Sensor",
  "provides":{
    "result":"current temperature",
    "unit":"degrees celsius"
  },
  "rating":3.4
}
```

Listing 1: JSON-based representation of a Web-enabled temperature sensor.

3.2.1 Extension of existing Strategies

The extension of an already existing strategy (e.g., integrating new JSON-based resource descriptors) may happen automatically by having a client submit a corresponding mapping that is integrated as an instance of an already existing strategy stub (e.g., a JSON-based strategy). To allow this, we have chosen to evaluate two approaches, *schema-driven* and *schema-less* strategy injection.

Schema-less Strategy Injection. To provide an easy and straightforward way of extending strategy stubs, we have created an injection mechanism that does not require clients to submit *explicit* information about the structure of resource representations. Rather, this information is directly inferred from the identifiers of a submitted mapping. For instance, in order to make the "Temperature Sensor" resource discoverable by DiscoWoT, it is sufficient to submit a `POST` request that specifies the mapping of the representation's identifiers to DiscoWoT's internal resource structure (as JSON array):

```
[
  {"name":"Name"},
  {"provides.result":"Services.Output"},
  {"provides.unit":"Services.Unit"},
  {"rating":"Review.Rating"}
]
```

DiscoWoT uses this information to create a new internal mapping associated with the JSON strategy stub. From then on, corresponding resource representations are mapped to a new instance of `Resource`, where the "Name", "Review", and "Services" members are created and initialized with the parsed information (e.g., "Temperature Sensor", "degrees celsius", etc.).

Schema-driven Strategy Injection. The schema-less approach has advantages with respect to the ease-of-use of the associated REST interface. However, as possibly valuable information about data types (e.g., the rating's *number* type) is lost, this method may not be suitable for all kinds of resource representations. We have therefore also implemented a mechanism for schema-driven strategy injection.

In the schema-driven mechanism, a client who would like the "Temperature Sensor" resource to be discoverable by the service needs to proceed in a two-step-process: First, a new strategy scheme is created by sending a `POST` request containing the corresponding schema (cf. Listing 2) to the `/strategies/JSON` endpoint. As for the schema-less approach, the client needs to submit the mapping corresponding to the resource representation. However, since this syntax should be associated to the created scheme, the `POST` is now sent to the `/strategies/JSON/{idScheme}` endpoint.

```
{
  "description":"Sensor JSON Schema",
  "type":"object",
  "properties":{
    "name":{
      "type":"string"
    },
    "service":{
      "type":"object",
      "properties": { ... }
      }
    },
    "rating":{
      "type":"number",
      "pattern":"[1-4]{1}[\.]?[0-9]*
    }
  }
}
```

Listing 2: JSON Schema document submitted to DiscoWoT (the service properties have been omitted for conciseness).

The integration of a schema-driven mapping requires more interaction with the service – two `POST` requests instead of a single one – and thus leads to a more complex interface. In the current version of the service, schema-based mappings are applied where appropriate (e.g., for JSON-based representations), while the schema-less approach is used with formats such as Microdata (cf. 3.3.2) that don't contain explicitly typed elements. The main drawback of this approach is that the service no longer provides a uniform interface for all types of strategies. It may therefore be more beneficial to use a best-effort mechanism for resolving typed values.

3.2.2 Creation of new Strategy Stubs

Apart from extending existing strategy stubs, we aim at enabling clients to also create completely new types of strategies that correspond to maybe not yet existing semantic description methods, i.e., implement strategy stubs themselves. However, the creation of novel strategies is more involved: To achieve that goal without restricting the extensibility of the service, we have to allow clients to inject code into the service at runtime, which inherently creates major security risks. To mitigate these, we have implemented a semi-automatic injection mechanism where a privileged user needs to monitor submitted strategies and block malicious code from being incorporated into the service. A more scalable approach, though, would be the use of a sandbox to prevent injected code from affecting the rest of the system.

3.2.3 Strategy Conflict Handling

As clients may submit any kind of strategy to the service, DiscoWoT is prone to conflicts when analyzing Web resources. The current implementation applies all registered strategies to a submitted device representation and assigns a confidence score to every analysis that depends on the ratio of correctly matched fields of the resource representation. The service then attempts to merge the information retrieved using the different strategies into a single instance of the internal resource format. If this fails, i.e., if two or more strategies produce different data for the same field, the scores are used to break ties. As an alternative for clients, DiscoWoT allows for direct strategy access, meaning that a client, rather than sending a resource's representation to the `/strategies` endpoint, may submit the request directly to

a specific, registered, strategy. In this case, only this very strategy is used to analyze the resource representation.

3.3 Discovery Strategies

In this section, we give an overview of semantic annotation formats that are incorporated as strategy stubs in the current version of DiscoWoT.

3.3.1 Microformats

Designed for both, human users and machines, Microformats provide a simple way to add semantics to Web resources by embedding information directly within (X)HTML tags. Several Microformats have been proposed in the literature, including *hcard* (or "HTML vCard") to encode contact information, *geo* and *adr* for specifying geographical locations, or *hProduct* for describing product details. The proposed *hRESTS* property is interesting in the domain of RESTful web services and devices as it allows to annotate a Web resource's markup with information on the specific services it offers, including a description of the input and output data as well as its format[10]. For us, Microformats are especially interesting because they provide a way of adding semantic, machine-understandable information to a resource's representation without affecting in which way the resource is perceived by humans.

In order to identify resources using Microformat-based tags, our service parses the (X)HTML representation found at the device URL and analyzes the embedded semantic information. The current version of DiscoWoT by default includes a single Microformats-based discovery strategy which uses a compound Microformat (composed of *hProduct*, *hReview*, *hListing*, *geo*, and *hRESTS*). Originally designed to support the optimized integration of resources into current search engines [4], this compound format also offers advantages with respect to the semantic identification of arbitrary, Microformat-described resources: Should a resource incorporate only *geo* markup, it can still be mapped using the provided discovery strategy. The Microformat-based strategy stub is implemented adopting a schema-less approach.

3.3.2 Microdata

Microdata, which is part of the HTML 5 drafts, definitely is a candidate for a semantic description with the potential to close the gap between simple but constrained Microformats and more abstract concepts based on RDF. It enriches the (X)HTML syntax with additional attributes, for instance *itemscope* or *itemprop*, which may be used to describe arbitrary concepts. This new format in principle allows for general-purpose semantic markup while remaining easy to use. However, the standard defines no uniform vocabulary for describing entities which gives rise to incompatibility issues. As for the Microformats, this stub features schema-less strategy integration.

3.3.3 JavaScript Object Notation (JSON)

Apart from being human-readable and easily understandable, the JSON format offers advantages with respect to its small footprint and facilitated parsing when compared to other languages. Probably due to these properties and to the conciseness of the format, we are currently witnessing a strong increase in JSON-based Web APIs. While we do not expect that Web resources will be describing their functionality using JSON on a Web-wide level, we envision such representations to act as interchange formats between single services or even confederations of Web resources. The corresponding stub can be used with and without attaching a JSON Schema to mappings.

3.3.4 RDFa

RDFa is a specification for the embedding of structured data – as attributes – within Web documents' markup. It is different from Microformats in that it does not specify vocabulary terms for the information to be included. Rather, RDFa relies on the external specification of ontologies like, for instance, Dublin Core[11]. RDFa-based stubs are extended using schema-driven strategy injection.

3.3.5 Deferred Discovery

Deferred discovery strategies have been incorporated into DiscoWoT in order to extend its functional range towards utilizing external services in conjunction with locally implemented discovery strategies. This is achieved by providing a simple interface for the registration of Web services that also provide service discovery mechanisms. Whenever DiscoWoT fails to semantically map a queried resource representation using local strategies, it falls back to the deferred discovery and forwards the resource's representation to the registered remote services. The remote service's answer is then again analyzed and the result is relayed to the client.

To register a new deferred strategy, DiscoWoT requires the definition of the remote service's URL and the name of the parameter to be used within requests. These two main pieces of information about a remote search engine may be submitted to DiscoWoT using an OpenSearch[12] document. Using this mechanism, it is for instance possible to add the Google search engine as a fallback mechanism by submitting a description containing its OpenSearch description.

3.3.6 HTTP-supported Discovery / Crawling

Since we consider Web of Things devices, we can assume that they will serve their functionality through a RESTful interface. As a consequence, for smart things not providing any explicit semantic description method, useful meta information can still be extracted simply by crawling their HTML representation as demonstrated in [2].

Thus, we provide a crawling discovery strategy that uses the properties of RESTful interfaces and of HTTP. From the root HTML page, the crawler typically is able to find a brief textual description of the device as well as the links to subresources which it can then follow for further extraction. It also uses an HTTP method called `OPTIONS` for each resource. This returns all HTTP methods supported for a particular resource, e.g., `PUT`, `POST`, `GET`, etc.

3.4 Service Architecture & Applications

DiscoWoT has been implemented as a stand-alone web service based on the REST principles. The service itself offers semantic information about its services using Microformats markup and offers an OpenSearch description. `HTTP POST` and `PUT` requests shall include data in JSON-format

[10]Note that hRESTS is sometimes criticized because some of the descriptions it offers are already standardized as part of HTTP. However, we believe that it proposes some additional interesting markup as well.

[11]`dublincore.org`

[12]`www.opensearch.org`

according to the description accessible via the respective URL. The main HTTP verbs map nicely onto the service's functions and provide an intuitive and user-friendly interface: `GET` is used to retrieve resource descriptions from the `/analyze` endpoint and to acquire information on registered strategy stubs, schemes, and mappings (`/strategies` endpoint). `POST`, `PUT`, and `DELETE` are used to create, update, or delete strategy stubs, schemes, and mappings, respectively.

We suggest three ways of how DiscoWoT may be utilized within smart things environments:

3.4.1 Smart Things Infrastructures

One primary task of any infrastructure for smart things is to provide a system for the network- and application-level discovery of physically connected items. DiscoWoT could be used by such infrastructures to help achieve the latter, as it provides a semantic discovery mechanism that enables the integration of further device discovery capabilities during operation of the system. The main benefits associated with the integration of an extensible device discovery mechanism are major simplifications of machine-to-machine communication scenarios and support for developments such as *Mashup Editors* that would allow end-users to configure and control ubiquitous systems.

3.4.2 Discovery Service Federations

Using deferred discovery strategies (cf. 3.3.5), it is possible to construct federations of discovery services where one of them – upon failure to successfully map a device internally – would contact other services to retrieve semantic information. This mechanism could also be used as a load-balancing mechanism within a federation of DiscoWoT entities where the resource discovery would be "outsourced" if local overload occurs. To register a remote instance of the service in this way, a new deferred strategy may be created by sending an OpenSearch document that includes the URL `http://remoteServer/analyze?representation={searchTerms}` to the `/strategies/deferred` endpoint.

3.4.3 Ad hoc Identification of Smart Things

Any entity interested in semantically mapping a Web-enabled device may use the service for maximum flexibility. This way of utilization may in particular prove useful for ad hoc interface generation or customization for Web resources.

4. CONCLUSIONS

DiscoWoT is a prototype implementation of a future-proof semantic discovery service for smart things. The service's aim is to facilitate device integration and interaction within the Web of Things on a semantic level. Its main advantage lies with the adoption of a strategy-based discovery mechanism that allows the injection and extension of discovery strategies at runtime. The service's decoupled architecture together with the concept of *deferred strategies* permits its use within federations of discovery services and thus further increases its flexibility and, ultimately, the utility it provides for clients. Interaction with the service is straightforward using its RESTful, browseable interface: All injected strategy schemes and mappings may be inspected and extended. For strategy stubs, we currently use a super user approach to avoid the insertion of malicious code. To further support clients when using DiscoWoT, its interface is published in the form of an attached OpenSearch document.

The service is continuously being extended, mainly with respect to improving the strategy injection mechanisms and the handling of the dynamic internal resource representation. As a next step, we will evaluate the service within simulated and testbed environments. After arriving at a stable version to be used by the public, we will use the system within the context of a distributed infrastructure for the Web of Things, where our ultimate goal is to create a reliable search mechanism for smart things. Regarding this project, we plan to interface DiscoWoT with other projects within the Web of Things domain, for instance with [2], where the authors present a platform that enables the social sharing of smart things based on existing social networks. Another interesting work that we imagine could be integrated well with DiscoWoT is the framework presented in [3]. Here, we are especially curious about how the proposed query augmentation approach is going to interoperate with our system, not only concerning user queries but also with respect to the extension of discovery strategies (e.g., concerning synonyms).

5. REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[2] D. Guinard, M. Fischer, and V. Trifa. Sharing Using Social Networks in a Composable Web of Things. In *Proceedings of the 1st IEEE International Workshop on the Web of Things (WoT 2010) at IEEE PerCom 2010*, Mannheim, Germany, Mar. 2010.

[3] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio. Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. *IEEE Transactions on Services Computing*, 3(3):223–235, July 2010.

[4] D. Guinard, V. Trifa, F. Mattern, and E. Wilde. From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices. In D. Uckelmann, M. Harrison, and F. Michahelles, editors, *Architecting the Internet of Things*, chapter 5. Springer, 2011.

[5] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan. Smart Semantic Middleware for the Internet of Things. In *ICINCO-ICSO*, pages 169–178, 2008.

[6] A. Kittur, E. H. Chi, B. A. Pendleton, B. Suh, and T. Mytkowicz. Power of the few vs. wisdom of the crowd: Wikipedia and the rise of the bourgeoisie. 2007.

[7] C. C. Marshall and F. M. Shipman. Which semantic web? In *HYPERTEXT '03: Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, pages 57–66, New York, NY, USA, 2003. ACM.

[8] S. Mayer, D. Guinard, and V. Trifa. Facilitating the integration and interaction of real-world services for the web of things. In *Urban Internet of Things - Towards Programmable Real-time Cities (UrbanIOT 2010); Workshop at the Internet of Things 2010 Conference (IoT 2010)*, Tokyo, Japan, Nov. 2010.

[9] V. Stirbu. Towards a RESTful Plug and Play Experience in the Web of Things. In *IEEE International Conference on Semantic Computing*, pages 512–517, Los Alamitos, CA, USA, Aug. 2008.