Towards Leveraging Semantic Web Service Technology for Personalized, Adaptive Automatic Ubiquitous Sensors Discovery in Context of the Internet of Things

Kobkaew Opasjumruskit
*Advisor:*Birgitta Koenig-Ries

Institute for Computer Science, Friedrich-Schiller-University Jena
{Kobkaew.Opasjumruskit,Birgitta.Koenig-Ries}@uni-jena.de

Abstract. Semantic web service technology equipped with user-context and environment awareness can enhance the automatic discovery of services. This thesis focuses on the service discovery aspect of MERCURY, a platform for straightforward, user-centric integration and management of heterogeneous devices and services via a web-based interface. The service discovery is used to find appropriate sensors, services, or actuators to perform certain functionalities required within each user-defined scenario. In contrast to existing works, the proposed service discovery approach is geared towards non-IT-savvy end users and supports various service-description formalism. Moreover, the matchmaking algorithm should be user-aware and environmentally adaptive rather than depends entirely on text-based searches. Hence, the goal of this thesis is to develop a service discovery module on top of existing techniques, which shall serve user requests according to their personal interests, expertise and circumstances.

Keywords: Service discovery, Adaptive computing, Semantic Service Description, User centric device

1 Motivation & Problem Statement

The idea of the Internet of Things[1]—where physical things are becoming smart devices—has been around for a decade. The technology allows us to combine sensors, actuators and services together seamlessly. Especially, in the era of pervasive mobile devices like these days, we are equipped with heterogeneous sensors and Internet access ubiquitously. To control and monitor devices remotely, a prominent approach is to expose them as web services and allow users to create new applications from services through development tools. Consequently, the non-IT-savvy users are withhold from utilizing this technology.

Though various works devote to the Internet of Things by enabling users to control smart devices through web-based applications, none of them can disentangle non-technical users from the integration of sensors, e.g., [2] requires

programming skills in order to formulate the required tasks. Even though [3] solves the aforementioned problem, it provides a limited set of web services, which of most are social media applications. While [4] offers a broader range of sensors and services, but it requires programming and hardware skills. The key aspect of this thesis is to find the best-fitting sensor among numerous candidates by leveraging semantic web services technology. This distinguishes MERCURY from the other competitors with a simple integration of sensors.

The remainder of this paper is organized as follows: beginning with the brief introduction to MERCURY by describing how it assists users with IoT, then explaining the situations in which the service discovery is used, and discussing the existing service discovery techniques. Next, the details of the service discovery module are elaborated. Lastly, the status of this work and future plan are concluded.

2 Brief Introduction to MERCURY

MERCURY is a platform for straightforward, user-centric integration, and management of heterogeneous devices and services via a web-based interface. It offers any user to connect sensors, actuators and services together from any sites via any accessible devices. First, these devices and services must be registered to the system. The sensor/service discovery module aids the registration process by utilizing a service description repository. Afterwards, the user can search devices and services, and links them together using the scenario modelling module. When this user saves a scenario, the executable script will be created from the scenario model and sent to the execution engine module. During the runtime, execution results can be monitored via the runtime UI module.

In [5], I demonstrated how to aid a user in accomplishing the desired task by utilizing environment-adaptive capabilities. Consider the following example: Imagine a user, Ann, decides to be woken up at 6 A.M. and go jogging if it does not rain in the morning. Otherwise, she prefers to receive an alarm message at 7 A.M., and postpone her jogging schedule to the evening. To accomplish this scenario, a GPS locator, rain sensors close to Ann's current locations and her calendar application need to be registered. Afterwards, Ann can wire sensors or services to meet the desired functionality.

3 The Thesis

3.1 The Topic

From the previous scenario, the user may experience some difficulties in sensor discovery. During the registration process, if there are sensors with the same name appear in the same network at the same time, this user might be unable to specify the preferred sensor. Although this problem can be resolved by using technical information to identify the preferred one, this approach is obviously inconvenient for non-technical users.

When the user starts to add a rain sensor into a defined scenario, the relevant devices or services should be suggested to aid modelling the scenario. Afterwards, the scenario can be saved for further execution. This user can use the dynamic service discovery function to enhance the execution process. For example, when the rain sensor in Ann's backyard is disconnected, the service discovery module should automatically search for a nearby sensor and replace it in this defined scenario.

The usage of the automatic sensor and service discovery can be deployed in three use cases.

- Service Registration: In order to register the right service, the user must search for the service through the automatic service discovery, which uses the semantic web service technology .
- Scenario Modelling: During a scenario creating process, a service request is created implicitly, and the matched services are put on the recommendation list. The recommendation is generated based on collaborative filtering, such as user rating, or based on descriptions of services. Then, the user can save and define a description for each scenario in order to discover them with the same technique later.
- Scenario Execution: If this user needs to change devices or services in a scenario according to his or her current context during the runtime, it is possible to define a generic scenario that is automatically updated with a proper device or service.

Furthermore, the service discovery can be enhanced by utilizing the personalized information, such as preferences or locations.

3.2 State of the Art

The basic idea behind the automatic service discovery is to annotate services with machine interpretable descriptions, which is explained in [6]. When a user or an application looks for a service, a service request with the capabilities of the service in a predefined format must be specified. The matchmaking component then compares available service descriptions with the request, and returns the service(s) that best match the user request.

This work assumes that the descriptions of all web services are in WSDL (Web Services Description Language). Because a WSDL description alone is insufficient for an automatic service discovery, ontology-based techniques, such as OWL-S (Web Ontology Language for Web Services) and WSMO (Web Service Modelling Ontology) are used to solve this problem. Nonetheless, these approaches are heavyweight and require significant efforts to provide their descriptions. SAWSDL (Semantic Annotations for WSDL) [7] is created to cope with semantic annotations in WSDL using arbitrary ontologies. A tag-based description technique proposed in [8] is another lightweight option as well.

For these service description approaches, there exist several initiatives to evaluate different description formalisms and matchmaking algorithms, e.g., the

S3 (Semantic Service Selection) Contest [9]. Since this work aims to integrate arbitrary services, any assumptions regarding the description framework cannot be made. Therefore, the solution is to support several different description formalisms, including OWL-S, SAWSDL, and tag-based.

The major challenges would be to answer the following questions: how to support various service description formalisms; how to make these formalisms configurable without tampering with the code to support new description languages in the future. In the real-time processing, how to improve the matchmaking time; how to guarantee the reliability of the dynamic service discovery during its runtime. And finally, how to choose the suitable information to add to the service request.

3.3 Initial Assumption

In MERCURY, all sensors and actuators are assumed to be available as web services with WSDL descriptions. Optionally, the services may have other descriptions, e.g., OWL-S, SAWSDL or social-tagging, etc. MERCURY itself does not support semantic annotation of services, though it does provide an interface for tagging. This thesis is not proposing new service matchmakers, instead, this work employs existing service matchmaking approaches.

It is compulsory to specify the domain of discoverable services by defining the ontology repository, so that MERCURY can automatically assign the ontology to a request from a user correctly. Otherwise, the request with ambiguous descriptions can lead to wrong analysis results, for example, apple can be defined in either domain of fruit or brand.

3.4 Solution Outline

Figure 1 shows the architecture of the MERCURY service discovery module. First, a user explicitly creates a request via the Request UI. Imagine a user looks up a weather forecast service, she or he needs to specify a city name as a service input and weather forecast as a service output. From these user inputs and, if possible, user context, the request converter module creates a service request. Afterwards, existing service matchmakers, such as [10] and [11] are used. Since each service may have more than one type of description, the user does not have to specify the description type of the service she or he is looking for. The matchmaker(s) will compare the request message with description files, which are already known by matchmakers. If the results from matchmakers contradict each other, the result integrator will assign weight values based on the ranking result from S3 Contest to determine the rankings. Optionally, the user can manually adjust these weight values. Consequently, the result integrator module will merge and rearrange the returned results from all matchmakers. The same methodology is also applied to an implicitly created request, which is automatically created during the modelling and execution process.

Currently, the request converter is supporting SAWSDL-based and OWL-S-based matchmakers. The module is designed to be externally re-configurable,

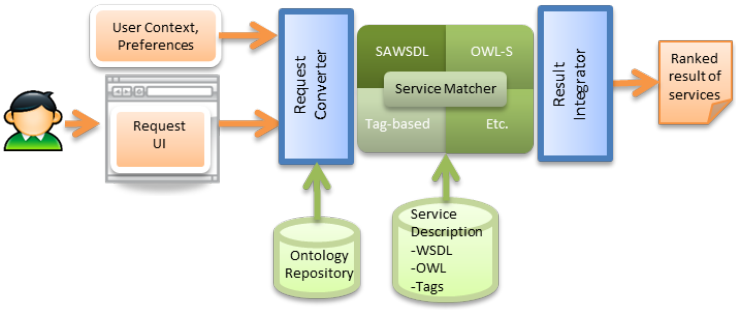

Fig. 1. MERCURY Service Discovery Architecture

so that the new semantic service description language can be introduced to the system without requiring to recompile the code. Before processing the final result, the whole system has to wait for the slowest matchmaker. Therefore, users are recommended not to activate all available matchmakers concurrently.

From S3 contest results, there is always a trade-off between the accuracy of result and the computing time. This weakens the reliability of the automatic discovery process significantly. Therefore, this challenge will be further studied. The source of user preferences can be explicitly gained from questioning each user, or be implicitly gained from activities of an individual; e.g. frequently used sensors, and friend recommended services. Moreover, their activities in social media applications can be analyzed and utilized in the sensor discovery module.

4 Research Plan

In the first year of the project, started in September, 2011, I presented the core architecture and the prototype in [5]. During the second year, I focused my research on the topic of sensor and service discovery. In April, 2013, the automatic service discovery was deployed in the registration process, and the idea how to utilize it in the context of MERCURY was submitted in [12].

To evaluate service matchmaking process, I set up the semantic service description repository from two sources: one is the testing collection from the S3 contest, which is used to measure the quality of service matchmaking; and the real service descriptions from www.programmableweb.com. The first prototype of automatic service discovery in the scenario modelling module was demonstrated in June, 2013.

Since the scenario modelling module is still in the process of development, the service discovery in this part is expected to be completed by October, 2013. The runtime service discovery will be integrated into execution process and is expected to be completed around April, 2014. The user evaluation should be established by May, 2014, and finally, the service discovery should be improved due to feedbacks from users, which is expected to be finished in October, 2014.

5 Conclusion & Future Directions

The focus of this thesis is to leverage the semantic web services technology to improve the automatic service discovery in the Internet of Things. The service discovery module should be aware of user context and personalized information to choose the most appropriate device or service for each user in any circumstance. It is assumed that the service description contains at least a WSDL file. On top of the existing works and the initial assumption, I presented the architecture of the service discovery. The current challenge is to find the right balance between the service matchmaking computing time and the accuracy of results. In the near future, I will implement the dynamic service discovery during the runtime. The evaluation of service recommendation can be made following the guidelines in the S3 Contest.

References

1. Mattern, F., & Floerkemeier, C.: From the Internet of Computers to the Internet of Things. LNCS Vol. 6462, Berlin, Heidelberg : Springer (2010)
2. Kovatsch, M., Lanter, M., & Duquennoy, S.: Actinium: A RESTful Runtime Container for Scriptable Internet of Things Applications. In Proceedings of the 3rd International Conference on the Internet of Things, pp. 135- 142 (2012)
3. IFTTT - If this then that, https://ifttt.com
4. Cloud Business Apps Integration CloudWork, https://cloudwork.com/
5. Opasjumruskit, K., Exposito, J., Koenig-Ries, B., Nauerz, A., & Welsch, M.: MERCURY: User Centric Device and Service Processing Demo paper. Paper presented at 19th Intl. workshop on Personalization and Recommendation on the Web and Beyond, Mensch & Computer, Konstanz, Germany (2012)
6. Maleshkova, M., Kopeck, J., & Pedrinaci, C.: Adapting SAWSDL for Semantic Annotations of RESTful Services. In Proceedings of the Confederated International Workshops and Posters on "On the Move to Meaningful Internet Systems", pp. 917-926. Berlin, Heidelberg : Springer (2009)
7. Semantic Annotations for WSDL, http://www.w3.org/TR/sawsdl/
8. Gawinecki, M., Cabri, G., Paprzycki, M., & Ganzha, M. Evaluation of Structured Collaborative Tagging for Web Service Matchmaking. In: Semantic Web Services Advancement through Evaluation, pp. 173-189. Berlin, Heidelberg : Springer (2012)
9. S3 Contest, http://www-ags.dfki.uni-sb.de/ klusch/s3/index.html.
10. Wei, D., Wang, T., Wang, J., & Bernstein, A.: SAWSDL-iMatcher: A customizable and effective Semantic Web Service matchmaker. Web Semantics: Science, Services and Agents on the WWW, 9(4), 402-417 (2011)
11. Klusch, M., Kapahnke, P., & Zinnikus, I.: SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants. In IEEE International Conference on Web Service, pp. 335-342 (2009)
12. Opasjumruskit, K., Exposito, J., Koenig-Ries, B., Nauerz, A., & Welsch, M. Service discovery with personal awareness in smart environments. Book chapter in Creating Personal, Social, and Urban Awareness through Pervasive Computing : IGI Global (2013)