Towards Open Data for Personal Web Tasking

Anna Ruokonen

Department of Electronic Engineering
City University of Hong Kong
Kowloon Tong, Hong Kong
Email: aruokone@cityu.edu.hk

Otto Hylli and Samuel Lahtinen

Pervasive Computing
Tampere University of Technology
Tampere, Finland
Email: firstname.lastname@tut.fi

***Abstract*—Open data is targeted for public use and usually released by the public sector, e.g., by the government. Its utilization, however, is currently restricted to programmers with knowledge on programming and processing of different data formats. The actual targeted end-users are simply lacking sufficient knowledge and technical skills. End-user tools for data visualization and browsing are available, but customization of the presented data and interaction patterns are limited. The users are unlikely to be able to exploit their creativity and meet their personal needs. In this paper, an on-going work on enabling open data sets for personal Web tasking is presented. Personal Web Tasking aims at automation of user-centric Web interactions. Work presented in this paper concentrates on automation of open data queries defined by the end-users.**

Keywords–Personal Web Tasking; Open Data Queries; End-User Development.

I. Introduction

The Web is becoming increasingly data-centric, consisting of interconnected data-oriented resources, large data sets and semantic linked data. Unlike HyperText Markup Language (HTML), which is designed to format and present text, EXtensible Markup Language (XML) is designed to structure data. With XML, individual data elements can be put in context of a larger taxonomy in order to turn plain text into data. While the main consumers of HTML documents are humans, the main data consumers are computers.

More and more open data sets are made available for the public, especially Public Sector Information (PSI) provided by the governments. Open means anyone can freely access, use, modify, and share for any purpose [1]. Also, by the definition, open data needs to be available in a machine-readable format, like Comma-Separated Values (CSV), XML, or Application Programming Interfaces (API). However, there is no generic mechanism for citizens to explore and consume the data sets [2] [3]. Thus, they depend on software developers to build proper tool-support for data consumption. These applications, however, are often specific to certain predefined data sets and provide limited possibilities for interaction.

End-user development aims at empowering non-programmer end-users to create their own applications. When building systems for end-user computation, the essential thing is to provide an environment, which supports the end-users' concepts and mental models [3]. Especially when consuming data, abstraction and visualization are an essential step to enable end-user interaction [4] [5].

In our previous work, we have studied description of Web resources such that they support end-user driven development [6]. It means that in addition to technical details, the service descriptions should include also end-user targeted metadata, including descriptions of parameters and a type of operations in a way that the concepts are familiar to the end-users.

Personal Web Tasking (PWT) aims at automation of user-centric repetitive web interactions by assisting users in achieving their personal goals on using internet systems [7]. A simple example of a personal Web task utilizing open data is monitoring of Air Quality Health Index (AQHI). It extracts the recent AQHI value and sends the end user a notification, if the value evaluates unhealthy. The PWT system executes the data query repeatedly on behalf of the user, e.g., every hour.

In this paper, an on-going work on end-user approach for creating personal Web tasks involving structured open data sets, provided as XML format or as an API, is presented. Data tasks are based on indicators, which are data elements selected by the end-user for monitoring. The task engine will automatically execute queries on the indicator values. At the current stage, a simple visualization of the data set is provided to enable the end-user identify certain indicators in the data set.

The rest of the paper is organized as follows. Related work is presented in Section II. In Section III, our approach for personal Web tasking with structured open data is presented. Challenges on building such end-user tasking system are discussed in Section IV. Finally, our plans for future work and conclusions are presented Section V.

II. Related work

There are existing approaches that focus on personal Web tasking, data visualization, or creation of personalized data feeds [5] [8]–[10]. Our approach aims at combining these aspects. Especially, existing approaches of personal Web tasking do not consider utilization of open data.

A self-adaptive application aims at predicting users preferences within a dynamic environment. It involves monitoring the user behavior and changes in the context and system infrastructures. It can be used to enable PWT systems that adapt on changing context and users personal goals to provide pleasant user experiences. In [8], Castaneda *et al.* present a self-adaptive approach where task personalization is done based on context information, such as user's personal context, and historical interactions and social network. Whereas these works emphasizes the user experience, we believe that the power of service-oriented systems and open data can be fully exploited when users are able to develop their own personal Web task consisting of service interactions.

 ISBN: 978-1-61208-392-6

In [9], Ng J. presents an approach to extend REST for smarter interactions to support personal Web tasking, called REpresentational Action State Transfer (REAST). A key feature related to personal Web tasking is a meta-model, which can be used to build user interface (UI) widgets to compose tasks without programming. REAST interactions are based on transitions from one action state of a resource to another action state of another resource. REAST actions are *Create*, *Read*, *Update*, *Notify*, and *Share*. Notify and Share can be used to enable social networking among users. In addition, event-based and time-based constraints can be attached to customize action execution by the end-users. It is assumed that information technology (IT) personnel first develop a configurable REAST application with supported UI widgets. The end-users can compose their personal tasks using the predefined UI widgets and resources. Our goal is that the user can select the resources and the system automatically generates the UI widget. Furthermore, the approach presented in [9] does not consider interaction with open data.

Castellani *et al.* present a web application framework for Internet of things (WebIoT) [10]. The framework is implemented based on Google Web Toolkit. It utilizes thing-centric design and communication is done using REpresentational State Transfer (REST) paradigm. It aims at integration of services, devices, sensors and other smart things into the Web 2.0 by allowing users to develop and exploit their own applications. Thus, sharing the same vision of user-centered development as emphasized in our work.

Generic mechanisms for data visualization, a tool not specific to any particular data set, for the end-users has been studied in [5]. The work presented by Mazumdar *et al.* aims at identifying the end-users' needs and preferences. Especially, they provide dashboard alternative visualizations, e.g., different kind diagrams, of data content is provided. The users are allowed to use filters to reason on the data. Filters are implemented as SPARQL Protocol and RDF Query Language (SPARQL) [11] queries. In our approach, a simpler approach for data interaction is taken, the user is only assumed to select indicators for future monitoring.

In our earlier work [6], we have presented an approach of adding metadata on the service descriptions. A similar approach, adding annotations on the open data sets, can be applied to provide useful information for the end-users and help them to develop their personal Web tasks. There is also existing work on providing guidelines for visual overviews of open government data [2].

III. Personal Web tasking with open data

As a part of our earlier work, we have presented tool-support for composing service interactions into personal Web tasks [12]. It is now extended to provide the end-users a simple mechanism to query and interact with open data sets. Sequences of end-user defined Web interactions, like Representational State Transfer (REST) service invocation, data queries are supported as personal Web tasks. Execution of the tasks is handled automatically by a task engine. An overview of the approach is shown in Figure 1.

A. Data abstraction and visualization

To enable end-user interaction, an abstraction and visualization of the data set must be provided. At the current

Figure 1. Personal Web tasking system

stage of our prototype implementation, a simple approach is implemented. The abstraction of the data set is provided by reducing the document structure. It is implemented as a simple algorithm that parses the XML feed. It finds the elements with the higest occurency and visualizes them to the user. For example, Figure 2 presents an air quality report AQHI24HrReport provided by the Hong Kong government as an XML feed. It contains 360 *item* elements presenting the resent measurement entries and some other XML elements presenting some document level information. After abstraction, only the structure of the *item* element and the latest values are presented to the end-user. Visualization the data content is shown in Figure 3.

```
<AQHI24HrReport>
<title>Environmental Protection Department - AQHI</title>
<link>http://www.aqhi.gov.hk</link>
<description>Environmental Protection Department -
   Past 24-hr AQHI</description>
<language>en-us</language>
<copyright>Environmental Protection Department</copyright>
<webMaster>enquiry@epd.gov.hk</webMaster>
<lastBuildDate>Fri, 26 Sep 2014 18:30:00 +0800</lastBuildDate>

<item>
   <type>Roadside Stations</type>
   <StationName>Mong Kok</StationName>
   <DateTime>Fri, 26 Sep 2014 18:00:00 +0800</DateTime>
   <aqhi>5</aqhi>
</item>

..
</AQHI24HrReport>
```

Figure 2. AQHI24HrReport

The example in Figure 3 shows the creation of a data task. First, the user selects a data set by defining the URL or selecting one of the data sets stored in the PWT system. The system extracts the data sets and presents an abstraction of its' contents to the user as a table. The users can select which parts of the data they want to monitor. Selected elements are included in the data task as data indicators. In the presented scenario, the user wants to connect to her Twitter account and send an automated tweet, if the air quality is unhealthy. Thus, she defines an interaction sequence, connecting the result of the data query, *if* control node, and Twitter service.

Figure 3 shows the creation of a simple control structure

with a condition. The user can select the desired operator from the operator list and specify the operand value. For numeric values <, >, and *equals* operators can be used. For strings, *equals*, *contains substring*, *prefix*, and *postfix* operators can be used. On the first column the user can select which indicators she wants to be included in the constraint. The first column presents the name of the data element. The third column shows supported operators, which can be used to create constraints on the indicator value. *ANY* means no constraint is used. The desired constraint value can be specified on the fourth column. The default values are the latest data values. In the given example, the user wants to monitor air quality of the Central/Western district of Hong Kong. If the air quality reaches unhealthy (*aqhi* > *6*), the user wants to send a tweet.

Before completing the task and sending it to the task engine, the user specifies the task execution frequency. For example, if execution frequency is specified to be one hour, the data query is automatically executed every hour by the task engine.

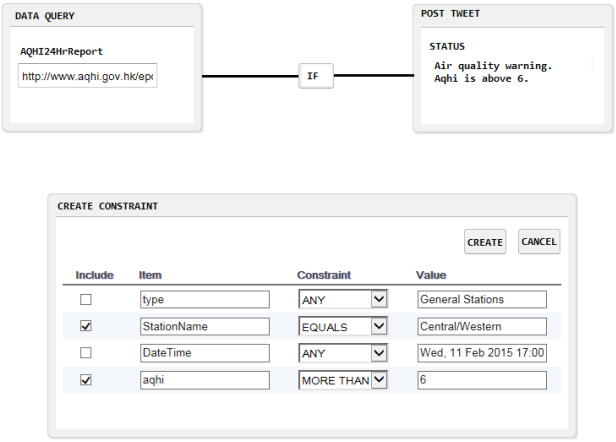

Figure 3. Designing a data task

B. Task creation with a data set

In the proposed Web Tasking system, tasks are composed of a sequence of activities. Basic activities are the CRUD operations to be used to interact with RESTful services: *POST*, *GET*, *PUT*, and *DELETE*. In addition, there are two activies *getDATA* and *NOTIFY* activity. *getDATA* activity can be attached to a data feed to simply get the latest value of a certain indicator or data feed. A *NOTIFY* activity is used to create a custom user notification. User defined constraints are implemented as variables and *if* statements in our task descriptions. Each activity takes a set of input parameters, which are required for its execution. For example, *getDATA* activity has two input parameters, the data URL and an indicator identifier. The values are extracted from the end-user input.

For example, as a result of scenario presented in Figure 3, an instance of *getDATA* activity is created. The parameters are the data *url* http://www.aqhi.gov.hk/epd/ddata/html/out/24aqhi_Eng.xml, and indicator identifiers *StationName* and *aqhi*. The output of the *getDATA* activity is connected to a *if* control node with the created condition (*StationName* equals *CentralWestern* and *aqhi* > 6). If the condition evaluates true, the task execution proceeds to invoking Twitter API to send a status update with a warning message. This is done with Twitter's *POST* method. *status* parameter defines the warning message. When a new task is created, the user specifies global parameter related to task execution, related to timing and repetition.

IV. Requirements for personal Web tasking system

The most important components of the PWT system are the task editor and the task engine. The overall architecture including the core components is shown in Figure 4. The task editor is a client side application, which provides a graphical interface for task creation. It can be used with a standard Web browser and it is targeted for the end-users. When the user has finished, the task editor generates an exportable task description given in XML-based language. The task description is stored and executed by the task engine. The task engine also provides a registry for services and data sets. The metadata editor, shown in Figure 4, is designed for augmenting service descriptions.

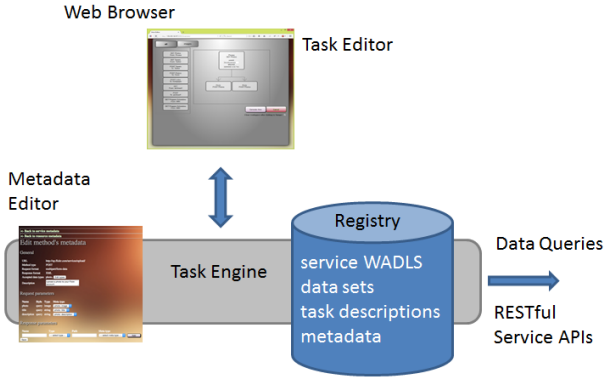

Figure 4. High level architecture

The overall personal Web tasking system must support the following activities: (1) task design, (2) Web interaction sequencing, and (3) task execution.

A. Task design

The main requirement for our end-user driven approach is to enable the user interaction with the data sets and RESTful APIs. Support for data interaction is provided thought data abstraction and data visualization as presented in the example in Section III. In the presented approach, only structured XML documents are considered as the data format. To provide a simple mechanism to interact with the data, our approach is based on monitoring single values, so called indicators. The selected indicators can be attached with simple constraints. For evaluating the indicator values, basic XML processing tools are used.

In our previous work, we have presented an approach for adding meta-data in RESTful service descriptions to support end-user driven service configurations. It assumes no modification on the existing services, but it requires, however, pre-registration of available service descriptions given in Web Application Description Language (WADL) [13]. A similar approach by annotating XML-based data sets can be used. However, this study presents a generic approach with no

 ISBN: 978-1-61208-392-6

advance registration of the annotated data sets. Thus, the use of metadata is limited to the XML tag names and the XML document structure.

B. Web interaction sequencing

Sequencing of Web interaction, such as data queries and RESTful service invocation, is supported by an XML-based task description language. An example of sequencing with the task description language is shown in Figure 5. It describes the interaction sequence of the air quality monitor task, containing two activities, the data query (*getDATA* activity) and posting a tweet on Twitter (*POST* activity).

```
<sequence>
   <getDATA url="http://www.aqhi.gov.hk/epd/
   ddata/html/out/24aqhi_Eng.xml" indicator=
   "/AQHI24HrReport/item[StationName='CentralWestern']
   [last()]/aqhi/text()">
   aqhi</getDATA>
         <if variable="aqhi" operator="bigger_than" value="6">
               <POST>
                  <request>status</request>
                  <response/>
                  <resource_id>twitter</resource_id>
               </POST>
         </if>
</sequence>
```

Figure 5. Task description

Variable *aqhi* is used to store the air quality index extracted from the data set and *status* contains the message that will be posted on Twitter. *getDATA* element defines the data set URL and how to get the indicator value. The *if* element takes a variable (the air quality), an operator (bigger than) and a value (6) and evaluates the expression (aqhi >6). If the condition evaluates true the *POST* activity is executed and and the message is posted to Twitter.

V. Conclusions and future work

More and more open data is made available for the public, especially in the public sector (e.g., European Union and PSI Directive). However, there is no generic mechanism for citizens to explore and consume the data. The users should be provided a simple mechanism to query and interact with the data sets. Our aim is to develop a personal Web tasking system, which supports the use of open XML data composed of a sequence of Web interactions. We present a generic approach, which automatically generates data abstraction and visualization of the data content enabling the end users to interact with the data set. The idea of personal Web tasking is to automatize repetitive Web interactions. For example, do a data query on the data source every hour to get the latest value of a certain indicator the user is interested in. To customize the data task, the user can define simple control structures with constraints. The end-user can, e.g., choose to receive notifications based on the latest data values.

The presented approach will be integrated with our previous work, which provides a platform for personal Web tasking with RESTful services. Interaction with the RESTful services is implemented via augmented WADL descriptions. RESTful service invocations and data queries along with simple control structures can be used as constituents of personal Web tasks, which are executed by the task engine. The ultimate goal of the presented approach is to empower end-users to create their own small Web applications and to benefit from the Internet of resources available. Especially, we want to enable non-programmers to explore and benefit from available public sector data.

References

[1] The Open Definition, Open Knowledge Source Code, http://opendefinition.org/od/ [accessed: 2015-01-02].

[2] A. Graves and J. Bustos-Jimnez, "Towards visual overviews for open government data," in DataWiz2014: Data Visualisation Workshop, ACM Hypertext 2014 conference, September 2014, pp. 1–6.

[3] J. Rode, M. Rosson, and M. P. Quiñones, "End user development of web applications," in End User Development, ser. Human-Computer Interaction Series, H. Lieberman, F. Patern, and V. Wulf, Eds. Springer Netherlands, 2006, vol. 9, pp. 161–182.

[4] A. Dadzie and M. Rowe, "Approaches to visualising linked data: A survey," Semantic Web, vol. 2, no. 2, Apr. 2011, pp. 89–124.

[5] S. Mazumdar, D. Petrelli, and F. Ciravegna, "Exploring user and system requirements of linked data visualization through a visual dashboard approach," Semantic Web, vol. 5, no. 3, 2012, pp. 203–220.

[6] O. Hylli, S. Lahtinen, A. Ruokonen, , and K. Systä, "Resource description for end-user driven service compositions," in IEEE Services 2014, 2nd International Workshop on Personalized Web Tasking (PWT 2014), July 2014, pp. 11–17.

[7] L. Castaneda, H. Muller, and N. Villegas, "Towards personalized web-tasking: Task simplification challenges," in 2013 IEEE Ninth World Congress on Services (SERVICES), June 2013, pp. 147–153.

[8] L. Castañeda, N. M. Villegas, and H. A. Müller, "Self-adaptive applications: On the development of personalized web-tasking systems," in Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, ser. SEAMS 2014. New York, NY, USA: ACM, 2014, pp. 49–54. [Online]. Available: http://doi.acm.org/10.1145/2593929.2593942/ [accessed:2015-01-02]

[9] J. W. Ng, "Adapting rest to reast building smarter interactions for personal web tasking," in IEEE Services 2014, 2nd International Workshop on Personalized Web Tasking (PWT 2014), July 2014, pp. 38–46.

[10] A. Castellani, M. Dissegna, N. Bui, and M. Zorzi, "Webiot: A web application framework for the internet of things," in Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE, April 2012, pp. 202–207.

[11] SPARQL Query Language for RDF, W3C, http://www.w3.org/TR/rdf-sparql-query/ [accessed: 2015-01-02].

[12] O. Hylli, S. Lahtinen, A. Ruokonen, and K. Systä, "Service composition for end-users," Acta Cybernetica, vol. 21, August 2014, pp. 383–399.

[13] Web Application Description Language (WADL), W3C, http://www.w3.org/Submission/wadl/ [accessed: 2015-01-02], 2009.

 ISBN: 978-1-61208-392-6