REST-based Meta Web Services in Mobile Application Frameworks

Daniel Sonntag
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3,
66123 Saarbruecken, Germany
daniel.sonntag@dfki.de

Daniel Porta
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3,
66123 Saarbruecken, Germany
daniel.porta@dfki.de

Jochen Setz
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3,
66123 Saarbruecken, Germany
jochen.setz@dfki.de

***Abstract*—This paper describes how a multimodal dialogue application framework can be used to implement specific mobile applications and dynamic HTTP-based REST services. REST services are already publicly available and provide useful location-based information for the user on the go. We use a distributed, ontology-based dialogue system architecture where every major component can be run on a different host, thereby increasing the scalability of the overall system with a mobile user interface. The dialogue system provides customised access to the *Google Maps Local Search* and two REST services provided by GeoNames (i.e., the *findNearbyWikipedia* search and the *findNearbyWeather* search).**

***Keywords*-Multimodal Dialogue, Application Backend, REST Services**

I. Introduction

Over the last several years, the market for speech technology has seen significant developments [1] as well as powerful, commercial off-the-shelf solutions for speech recognition (ASR) or speech synthesis (TTS). However, these infrastructures have had only moderate success so far in the entertainment or industrial sector, especially in the mobile user interface (UI) context. This is the case because a dialogue system cannot easily be constructed for a mobile application. Only distributed systems, where the speech input processing is done on a server, allow for real-time dialogue reactions. Additionally, the dialogue engineering task requires many customisations to specific end user applications.

A modern multimodal dialogue system can act as the middleware between the mobile clients and the backend services, which hides complexity from the user. It should present only aggregated information to the user, thereby customising the presentation rules to the specifics and requirements of various output devices. This is possible because these architectures often encapsulate the dialogue proper from the rest of the application. These architectural decisions are often based on usability issues that arise when dealing with end-to-end dialogue-based interaction systems for industrial dissemination. Prominent examples of integration platforms include TRIPS [2], Galaxy Communicator [3], and SmartWeb [4]; these infrastructures mainly address the interconnection of heterogeneous software components. Earlier projects [5], [6] have integrated different sub-components into multimodal interaction systems. Thereby, hub-and-spoke dialogue frameworks played a major role.

We use the encapsulation characteristic for our own benefit. In multiple tier architectures, multiple user interfaces can be used more easily, and the application backend may comprise of several information sources. Accordingly, the multimodal dialogue application framework can be used to connect specific mobile interfaces at the frontend with dynamic HTTP-based REST [7] services at the backend. Works in mobile application frameworks are often concerned with physical issues, e.g., supporting wayfinding with tactile cues [8] or interactive experiences for cyclists [9]. Other work concerns mobile search scenarios and incidental information [10] where contexts such as location and time play major roles in information discovery [11] or the design of Web-based mobile services [12]. Our work addresses available services through an application backend in a three tier architecture and focused on multimodal dialogue for the user interaction with locations on a map.

In what could be a typical application scenario, the user is visiting historic sites in Berlin and wants to get location-based (restaurant) information. In previous projects [13], we used a Web service infrastructure for the backend access. We also developed a semantic representation formalism based on OWL-S and a service composition component to interpret an ontological user query. Although the composition module was able to dynamically compose different Web services for hotel, restaurant, and theater information, the maintenance of private/public WSDL Web services was more difficult and produced a high degree of dependence on external service providers. By contrast, the interaction with HTTP/REST services is much simpler, and these services provide useful location-based information, too. For combined searches with geographical coordinates, we selected the *Google Maps Local Search* REST service and the two services for reverse geocoding provided by GeoNames for inclusion, namely the *findNearbyWikipedia* search, and the *findNearbyWeather* search.

This paper is structured as follows: in Section II, we will

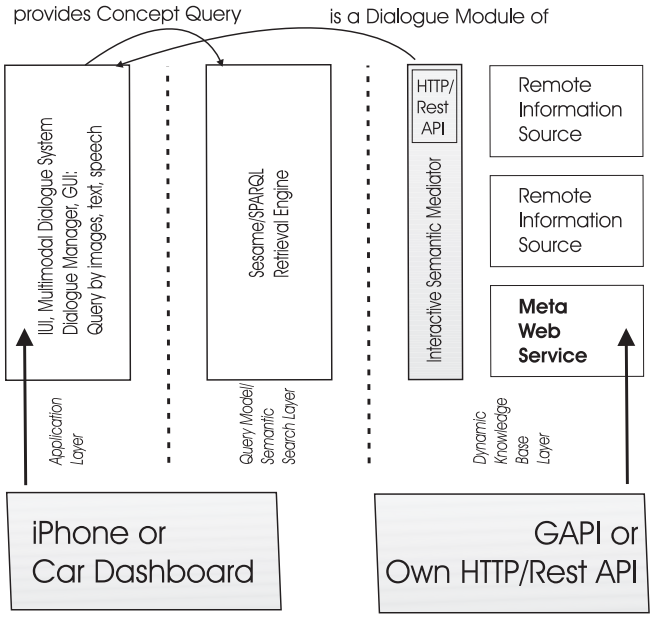

Figure 1. Three Tier Querying Architecture

present the application architecture, followed by the detailed application scenario (Section III). Section IV provides a conclusion and future work.

II. Application Architecture

Figure 1 outlines the three tier architecture. It consists of an application layer (the dialogue system), a query model and semantic search layer (which, e.g., extracts keywords from the result of the natural language understanding, NLU), and a dynamic knowledge base layer which addresses information sources in general. The knowledge layer hosts the interactive semantic mediator (providing information-source-specific queries) and remote information sources. The most important characteristic is the three tier distribution [4]. Multiple input/output devices and multiple heterogeneous information sources can be integrated into the respective technical layer. We already evaluated the architecture for a customised iPhone application [14] and a car dashboard [4] while addressing SOAP-based Web services as information sources. In this paper, we present a completely new technical implementation: the REST API in combination with a generic graphical UI concept for the iPhone.

A. Meta Web Services

Many Web services (that can be found on the Web) can be directly used as key components in a Web service composition process. For example, we have been experimenting with services that provide keyword-to-SPARQL query functionality (for the basic idea, see [15]).

Apart from services which help the composition process, other standard Web services such as *Amazon Web Services* exist that are often part of such composition chains. Preferably, SOAP services are considered for composition due to their WSDL [16]-formalised technical interface. We also consider services based on REST, JSON, XML-RPC, and the like.

Whenever we provide a custom interface to a compound of those services, we speak of *Meta Web Services*. A Meta Web Service provides a complex processing service. For example, one such Meta Web Service maps the GAPI [17] results on a ranked result table containing several YouTube videos with metadata (title, list of comments, etc.). The ranking of the videos is done by computing a string-based distance between the (keyword) query terms and the textual description of videos.

One prominent graphical framework for building Meta Web Services by aggregating different HTTP/REST-based information sources is Yahoo Pipes [18]. However, the aggregation processes implemented in such frameworks are hard-wired and do not allow the conditional execution of sub-processes. Additionally, information extraction from unstructured textual information sources is (apart from the extraction of location information) not supported in the framework. In contrast, the Business Process Execution Language (BPEL) [19] allows for more degrees of freedom. It defines a standard and complete declarative language for the composition of Web services. Graphical editors for modelling such processes are also available. Unfortunately, services in the composition chain have to be formally described in terms of WSDL (which is often not the case for HTTP/REST-based services). The Posr framework [20] provides a holistic approach for Web service composition not only on the technical but also on the UI layer. Although Posr is able to transform browser-based Web forms into a Meta Web Service, they only consider WSDL-formalised SOAP Web services and leave the inclusion of HTTP/REST-based services as future work. Hence, one of our main goals is the inclusion of HTTP/REST-based services into a mobile dialogue architecture. Our solution is a custom HTTP/REST Meta Web Service for mobile location-based scenarios.

B. Custom HTTP/REST Meta Web Service

As shown in Figure 2, our own HTTP/REST API service comprises of three modules: the query builder, the query/retrieval module and a set of presenters as the result aggregation module. The query builder takes keywords as input and creates the HTTP/REST URL query with respective arguments. The query/retrieval module handles the information retrieval step. An incoming result triggers a state change in the result aggregation module. The module informs all subscribed presenters of the new result. The presenters can serialise the XML results into files for logging purposes, or parse the results using a SAX parser to provide data tables or platform/ontology-specific exchange objects (TFS).

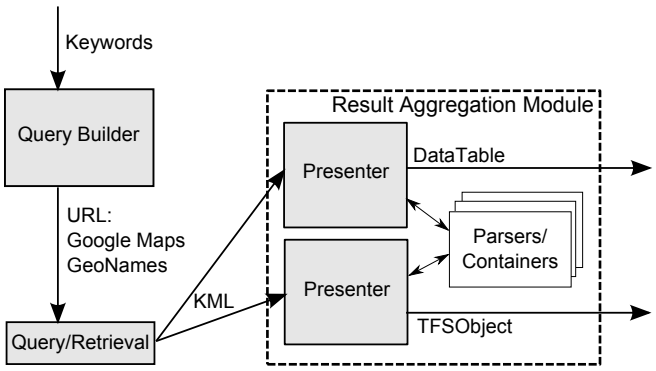

Figure 2. Own HTTP/REST API

Name	Rating
VAU	4.0
Fernsehturm	4.1
Letzte Instanz	3,4
Pergammonkeller	
Brecht	3,8

Name	Fernsehturm
Rating	4.1
Street	Panormastaße 1a
Phone	030 24757537
WWW	http://tv-turm.de
Pic	http://...

http://maps.google.com?
q=Restaurant&near=Berlin+M
&output=kml&mrt=all
&oe=UTF8&v=2.0

Figure 3. Results of a Google Maps Local Search

1) Google Maps Local Search: Currently, Google only provides a public JavaScript API for Google Maps; therefore, we had to rely on publicly available REST API specifications as, e.g., provided by Mapki [21]. Mapki provides an almost complete parameter description for accessing Google Maps functionalities. We selected a suitable subset of these parameters (Table I) for our mobile scenario. The `near` parameter is of particular importance as it parametrises a local search with a point-of-interest (POI) as starting point. The `output`, `oe`, and `v` parameters specify the result format. In our case, an *UFT-8* encoded *KML* [22] file is requested. The `mrt` can be used to narrow down the

Parameter	Value	Description
q	category	Keywords for the search
near	a location	Starts the local search around the POI location
mrt	all	Get as much information as possible
output	kml	Get the results as a kml file
oe	UTF8	Encoding of the output
v	2.0	Version of the KML schema

Table I
PARAMETERS USED FOR QUERYING GOOGLE MAPS

Name	Distance
Fernsehturm	0.009
Alexanderplatz	0.13
Karl-Liebknecht-Straße	0.1982
St. Mary's Church	0.239
Neptunbrunnen	0.2758
Rotes Rathaus	0.2973

Title	Rotes Rathaus
Summary	The Rotes Rathaus is the town hall of ...
Distance	0.2973
Wikipedia Article	http://en.wikipedia.org/...

Figure 4. Results of a Wikipedia NearBy Search

results at query time according to categories, e.g., locations, businesses, places annotated by users, and the links to Wikipedia. For example, search results in the HTTP response for the query “Restaurant near Berlin Mitte” are depicted in Figure 3.

Google Maps’ answers contain detailed pieces of information about restaurants or hotels including the names, user ratings, address data, the URL, a link to a thumbnail, and the geographical coordinates. The vast bulk of this information is well-structured in the KML output. Some details, however, e.g., phone numbers and user ratings, are only available as HTML fragments within CDATA text blocks in the KML structure. Hence, Java classes for regular expression String matches are also necessary. Similar customisation requirements are expected also for new Meta Web Services. This is clearly a drawback for the general scalability of HTTP/REST service integration. As Google Maps does not provide any Wikipedia information in the desired KML output (Google Maps only links to Wikipedia’s HTML) and since we are also interested in additional POIs and/or Wikipedia results, we access additional services. In the context of our mobile scenario, we decided to use the GeoNames [23] REST API, i.e., the *findNearbyWikipedia* search.

2) GeoNames: GeoNames provides two Web services for reverse geocoding which fit in our application scenario, namely the *findNearbyWikipedia* and *findNearbyWeather* services. (Both services take latitude, longitude, and radius as parameters.) The Wikipedia search provides links to articles about memorials, landmarks, or other interesting places. The weather search provides the current weather conditions. Wikipedia results contain summaries, distance, and the link to the full article, and other information (Figure 4). As the distance is just the beeline between the starting point and the place, we used Google Maps again to get the route planning

and actual walking distance as an additional result for the mobile user. The *findNearbyWeather* service obtains, apart from temperature, details about the weather conditions such as wind speed or humidity. Certainly, weather conditions are not really meaningful when planning a city walking tour (San Francisco's micro climate might be an exception) but can become very important when going hiking [24].

C. Mobile Client Application

Nowadays, many different mobile device platforms exist. The majority is equipped with a full-fledged Web browser that enables us to provide platform-independent graphical user interfaces (GUIs) by means of DHTML-based Rich Internet Applications (RIA). On the code basis (in our case a declarative XML-based language), we make use of the OpenLaszlo Rich Internet Framework [25], which turned out to be very suitable (in contrast to the findings presented in [26]) for Web applications on modern mobile devices without the need for a Flash player. Since we need to send and receive optional audio data for speech-based interaction, we implemented a lightweight native application that embeds a full-screen Web browser and additionally provides a platform-dependent audio streaming functionality (a similar approach is pursued by [27]). The communication with the dialogue system is implemented by a uniform client-side JavaScript API using long polling AJAX (Asynchronous JavaScript and XML) requests to a server-side HTTP/REST-based endpoint. Currently, client applications exist for the iPhone and the Android platform. Desktop Web browsers can (in combination with an optional Java applet for audio streaming) also render the DHTML-based GUIs, which eases the development process of such multimodal UIs.

III. Application Scenario

First, Google Maps supplies the geographical coordinates of a POI such as a hotel or restaurant. We use the coordinates to get a list of other interesting places close by. Then, we display further POIs on the current map. Second, the user can ask for additional POIs in the vicinity according to a manual POI selection on the map which initiates a GeoNames search. The geographical coordinates from Google Maps or GeoNames can, third, be used to obtain weather conditions or Wikipedia hits around the resulting POI.

The following dialogue between a user on the go and our mobile interface (which is connected to the dialogue system proper via UMTS or WiFi) is an example of a typical user interaction sequence, which combines the *Google Maps Local Search*, GeoNames *findNearbyWikipedia* search, and GeoNames *findNearbyWeather* search.

1 **U:** *"Where can I find good restaurants (in Berlin Mitte)?"* (If no named entity for location is mentioned, we use the built-in GPS locator. Good restaurants are selected according to a heuristics that takes user rankings into account.)
2 **S:** Shows a map and list of restaurants. See Figure 3, left.
3 **U:** *"Give me more information about the second one / the Fernsehturm."*
4 **S:** Shows corresponding detailed pieces of information according to referral restaurant data record. See Figure 3, right.
5 **U:** *"What else can I visit here?"*
6 **S:** Provides Wikipedia results of nearby POIs and the walking route. See Figure 4, left.
7 **U:** *"What do you know about this POI?"*
8 **S:** Provides the Wikipedia article (and current weather information for the respective district). See Figure 4, right.

Please note that the user input is paraphrased for illustration purposes. In a multimodal dialogue system, the user should be able to switch between spoken, written, or clicked input. This also means that a summary of the Wikipedia article could be synthesised. However, the speech option is not necessary in this scenario and does not seem to provide a good trade-off between system complexity and user experience since additional third-party components for ASR, NLU, and TTS would become necessary.

Figure 5 depicts the described dialogue and interaction sequence. In (1), the user searches for restaurants by typing an appropriate query into the text field and pressing the search button. Then the result list (2) pops up. The locations of the restaurants (the respective POIs) are shown on the map in the background. The user selects the entry about the restaurant "Fernsehturm" and presses the information button (indicated by *i-icon*) at the top of the screen. The resulting detail screen is shown in (3). After reading the information, the user intends to have dinner there, but he wants to combine it with a little sightseeing nearby. So, the user presses the "What's nearby?" button (indicated by the *circle-icon*). An appropriate result list sorted by the distance from the restaurant appears in which the user selects in (4) the last entry ("Rotes Rathaus"). By pressing the information button again, details to the selected sight including current weather information and a short Wikipedia summary are presented (5). Furthermore, the route from the restaurant to the sight is displayed on the map. The user can minimise the previous view in order to inspect the proposed route (6).

IV. Conclusions and Future Work

Multimodal dialogue application frameworks can be used to implement specific mobile applications and dynamic HTTP-based REST services. These infrastructures can overcome the technical limitations imposed by current mobile device hardware and software. In addition, new services of independent providers can be added easily.

We created a custom HTTP/REST Meta Web Service for mobile location-based scenarios and explained how this

Figure 5. Mobile GUI Screenshots

service integrates into a multimodal dialogue framework. In addition, we provided a real-world application scenario and explained our generic dialogue framework and a specific implementation of a new Meta Web Service.

With the new Meta Web Service, we can provide meaningful information for travellers or tourists. Currently, we are restricted by the information from the Google Maps and GeoNames services. We think that it will soon be possible to find new REST services to be integrated.

The keyword/term based input possibilities of Google Maps is quite suitable for dynamic POI searches. However, the POI classes (e.g., hotel, pharmacy) are not documented so that a user has to search for those in a trial-and-error fashion. This relevance-feedback interaction style could bring speech-based interaction into the fore if the speech grammar accepts the relevant portion of open-domain POI classes.

A distributed dialogue infrastructure overcomes the technical limitations imposed by current mobile device hardware and software (which severely hinders the potential benefits of mobile speech-based applications and, e.g., augmented reality applications such as 3D egocentric views of the user's surrounding [28]). We hypothesise that these new interaction and visualisation techniques, which demand for distributed applications, will become very beneficial in the future.

V. Acknowledgments

This research has been supported by the THESEUS Research Programme in the Core Technology Cluster WP4 and the TEXO use case, which was funded by the German Federal Ministry of Economy and Technology under the promotional reference “01MQ07012“. The authors take the responsibility for the contents.

References

[1] R. Pieraccini and J. Huerta, “Where do we go from here? research and commercial spoken dialog systems.” in *Proc. 6th SIGDdial Workshop on Discourse and Dialogue*, 2005, pp. 1–10.

[2] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent, “An Architecture for a Generic Dialogue Shell,” *Natural Language Engineering*, vol. 6, no. 3, pp. 1–16, 2000.

[3] S. Seneff, R. Lau, and J. Polifroni, “Organization, Communication, and Control in the Galaxy-II Conversational System,” in *Proc. of Eurospeech*, 1999, pp. 1271–1274.

[4] D. Sonntag, *Ontologies and Adaptivity in Dialogue for Question Answering*. AKA and IOS Press, Heidelberg, 2010.

[5] W. Wahlster, “SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell,” in *Proc. Human Computer Interaction Status Conf.*, R. Krahl and D. Günther, Eds. DLR, 2003, pp. 47–62.

[6] N. Reithinger, D. Fedeler, A. Kumar, C. Lauer, E. Pecourt, and L. Romary, “MIAMM - A Multimodal Dialogue System Using Haptics,” in *Advances in Natural Multimodal Dialogue Systems*. Springer, 2005.

[7] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Ph.D. dissertation, University of California, Irvine, 2000.

[8] M. Pielot, N. Henze, and S. Boll, “Supporting map-based wayfinding with tactile cues,” in *Proc. 11th Int’l Conf. Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*. ACM, 2009, pp. 170–179.

[9] D. Rowland, M. Flintham, L. Oppermann, J. Marshall, A. Chamberlain, B. Koleva, S. Benford, and C. Perez, “Ubikequitous computing: designing interactive experiences for cyclists,” in *Proc. 11th Int’l Conf. Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*. ACM, 2009, pp. 151–159.

[10] D. Arter, G. Buchanan, M. Jones, and R. Harper, “Incidental information and mobile search,” in *Proc. 9th Int’l Conf. Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*. ACM, 2007, pp. 129–144.

[11] K. Church, J. Neumann, M. Cherubini, and N. Oliver, “SocialSearchBrowser: a novel mobile search and information discovery tool,” in *Proc. 14th Int’l Conf. Intelligent User Interfaces (IUI)*. ACM, 2010, pp. 101–110.

[12] C. Riva and M. Laitkorpi, “Designing Web-Based Mobile Services with REST,” in *Service-Oriented Computing - ICSOC 2007 Int’l Workshops, Revised Selected Papers*. Springer, 2009, pp. 439–450.

[13] D. Sonntag, R. Engel, G. Herzog, A. Pfalzgraf, N. Pfleger, M. Romanelli, and N. Reithinger, *Artifical Intelligence for Human Computing*. Springer, 2007, ch. SmartWeb Handheld—Multimodal Interaction with Ontological Knowledge Bases and Semantic Web Services, pp. 272–295.

[14] D. Porta, D. Sonntag, and R. Neßelrath, “A Multimodal Mobile B2B Dialogue Interface on the iPhone,” in *Proc. 4th Workshop on Speech in Mobile and Pervasive Environments (SiMPE)*, 2009.

[15] S. R. Duc Thanh Tran, Haofen Wang and P. Cimiano, “Top-k exploration of query graph candidates for efficient keyword search on rdf,” in *Proc. 25th Int’l Conf. Data Engineering (ICDE)*, 2009.

[16] W3C. (2010, Jul.) Web Service Description Language. [Online]. Available: http://www.w3.org/TR/wsdl20/

[17] Google. (2010, Jul.) Google APIs. [Online]. Available: http://code.google.com/intl/en-EN/more/

[18] Yahoo. (2010, Jul.) Yahoo Pipes. [Online]. Available: http://pipes.yahoo.com/

[19] OASIS. (2010, Jul.) Business Process Execution Language. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[20] M. AbuJarour, M. Craculeac, F. Menge, T. Vogel, and J.-F. Schwarz, “Posr: A Comprehensive System for Aggregating and Using Web Services,” *IEEE Congress on Services*, vol. 0, pp. 139–146, 2009.

[21] Mapki. (2010, Jul.). [Online]. Available: http://mapki.com/

[22] OGC. (2010, Jul.) KML. [Online]. Available: http://www.opengeospatial.org/standards/kml/

[23] GeoNames. (2010, Jul.) WebServices overview. [Online]. Available: http://www.geonames.org/export/ws-overview.html

[24] D. Porta, “A Novel, Community-Enabled Mobile Information System for Hikers,” in *Proc. 2nd Int’l Conf. Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*. IEEE Computer Society, 2008, pp. 438–444.

[25] LaszloSystems. (2010, Jul.) OpenLaszlo. [Online]. Available: http://www.openlaszlo.org/

[26] M. Annett and E. Stroulia, “Building highly-interactive, data-intensive, REST applications: the Invenio experience,” in *Proc. Conf. Advanced Studies on Collaborative Research (CASCON)*. ACM, 2008, pp. 192–206.

[27] A. Gruenstein, I. McGraw, and I. Badr, “The WAMI toolkit for developing, deploying, and evaluating web-accessible multimodal interfaces,” in *Proc. 10th Int’l Conf. Multimodal Interfaces (IMCI)*. ACM, 2008, pp. 141–148.

[28] Y. Tokusho and S. Feiner, “Prototyping an Outdoor Mobile Augmented Reality Street View Application,” in *Int’l Symp. Mixed and Augmented Reality (ISMAR)*, 2009.