Two-staged approach for semantically annotating and brokering TV-related services

Hong Qing Yu, Neil Benn, Stefan Dietze, Carlos Pedrinaci, Dong Liu, John Domingue
Knowledge Media Institute
The Open University
Milton Keynes, United Kingdom
e-mail: {h.q.yu, n.j.l.benn, s.dietze, c.pedrinaci, d.liu, j.b.domingue}@open.ac.uk

Ronald Siebes
Department of Computer Science
Vrije Universiteit Amsterdam
The Netherlands
e-mail: rm.siebes@few.vu.nl

Abstract—**Nowadays, more and more distributed digital TV and TV-related resources are published on the Web, such as Electronic Personal TV Guide (EPG) data. To enable applications to access these resources easily, the TV resource data is commonly provided by Web service technologies. The huge variety of data related to the TV domain and the wide range of services that provide it, raises the need to have a broker to discover, select and orchestrate services to satisfy the runtime requirements of applications that invoke these services. The variety of data and heterogeneous nature of the service capabilities makes it a challenging domain for automated web-service discovery and composition. To overcome these issues, we propose a two-stage service annotation approach, which is resolved by integrating Linked Services and IRS-III semantic web services framework, to complete the lifecycle of service annotating, publishing, deploying, discovering, orchestration and dynamic invocation. This approach satisfies both developer's and application's requirements to use Semantic Web Services (SWS) technologies manually and automatically.**

Semantic Web Services; Linked Services; Semantic Web; Linked Data; Digital TV

I. INTRODUCTION

With high demands of Digital TV and IP TV, more and more TV broadcast organizations provide TV related content as multimedia resources to be accessed through Web technologies [1]. This allows audiences to better interact with the broadcast content and allows them to watch programs via various kinds of media and devices. For example, a user can view EPG data via an iPhone, or to get personal recommendations via a TV Box. Also, a user can search for a TV program that she/he is interested in and retrieve the actors' detailed information along with the broadcast information of the TV program. To enable end-user applications to access these TV resources easily, the TV resource data is commonly provided by Web service technologies. There are many different service message schemas and many different service functionalities available such as video processing services, video transcoding services, EPG related services, etc. Dealing with this level of heterogeneity is a major challenge that raises the need to have a broker to discover, select, mediate, and orchestrate services to satisfy the runtime requirements of the end-user applications. Furthermore, with large-scale availability of TV online resources, such a middleware component is essential for not only sharing the resources but also enabling interoperability between the functionalities that use and process the resources.

In this paper, we propose and implement a Semantic TV Resource Broker (STRB) based on a Service Oriented Architecture (SOA) approach in combination with Semantic Web Services standards. In other words, the functionalities are deployed as Web services with semantic metadata. Here we refer to a Web service as any kind of software functionality that is accessible through HTTP, varying from REST-based APIs to SOAP interfaces.

To achieve this goal, we need to meet some important challenges:

- *Develop an easy approach that allows service developers to annotate semantic metadata for their TV-related services.* Currently, semantic annotations are mainly based on WSMO [6] and OWL-S [5]. However, there is a gap between developer's knowledge about these ontologies and tools for supporting developers. In addition, the complexity of the WSMO and OWL-S standards impedes the adoption by the developers' community to use them in large-scale applications.
- *Enable both the developer and the applications to seamlessly interact and align the available semantic metadata provided by the developers.* This challenge is to bridge the somewhat conflicting requirements. A developer needs an easy-to- understand and human-readable description of the functionalities of the services. In contrast, in order for applications to interact and integrate services, they require a machine-oriented environment to dynamically work with services and their metadata. Therefore, the service metadata annotations should have two different levels of interoperability.

The contributions of our approach in meeting the above challenges are:

- We implement a two-stage approach for semantically annotating the services in the STRB.

The two stages are (1) allowing developers to annotate and publish the services by using the *Linked Services* [2] approach based on lightweight RDF annotations through a Web form; (2) the Linked Services RDF annotations feed into IRS-III [4] semantic execution environment to semantically deploy the service. The term Linked Services is used to describe the fact that the semantic service annotations using this approach are much easier to produce (than say those based on WSMO or OWL-S) and can be populated with references to widely established Linked Data vocabularies. Furthermore, they address a much wider audience and allow even non-SWS experts and lay people to describe and annotate services.

- With these two levels of annotations, the developer can manually discover and select services by simply using SPARQL [18] queries to develop applications on top of the STRB. Meanwhile, applications can invoke the atomic service or orchestrated service through the IRS-III semantic execution environment that can automatically discover, orchestrate and invoke the available Web services.

This paper is organized as follows: Section 2 introduces the background information about the NoTube project[1] and related terminologies and technologies. Section 3 shows two realistic use cases that present the main requirements of the STRB. Section 4 illustrates the architecture of STRB and the detailed implementation of the prototype. Finally, the conclusion and future work are discussed in Section 5.

II. BACKGROUND

The STRB is defined as an important middleware component for the NoTube project with the purpose of automatically finding, combining and invoking relevant Web Services based on goals specified by the NoTube application developers. The ultimate goal of this project is to develop a flexible/adaptive end-to-end architecture, based on semantic technologies, for personalized creation, distribution and consumption of TV content. The project takes a user-centric approach by investigating the fundamental aspects of consumers' content-customization needs, interaction requirements and entertainment wishes, which will shape the future of the television experience. Figure 1 shows the overall NoTube environment, containing four conceptual layers of service, broker (or called control), application (or called view) and screen. The broker is centrally located in the architecture because it is responsible for the communication between applications and services.

At this moment, we have collected more than 40 existing services relevant to the TV domain. The functionality of the services contains EPG services, context logging services (e.g. user profiling service), enrichment services (getting richer information about a certain TV resource from multiple resources), social network services (like interfaces to the functionality of. Twitter or Facebook), and recommendation services.

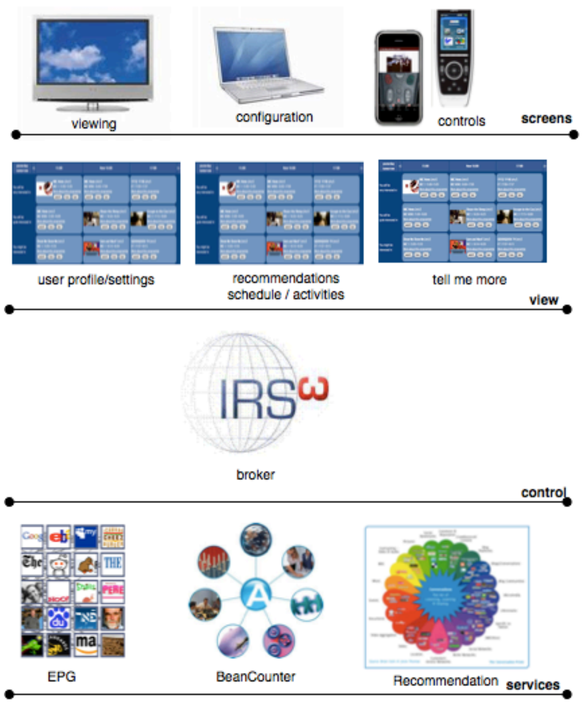


Figure 1. NoTube overall framework from STRB side.

The technology used to develop the broker is based on SWS. The broker uses a repository of SWS in order to perform its functionalities. SWS are Web Services enriched with ontological descriptions of Web services in terms of their capabilities, interfaces and non-functional properties. SWS technologies aim at automatic discovery, selection and orchestration of distributed services for a particularly expressed user's request/goal. The SWS approach utilises both standard Web service technology such as SOAP [8], UDDI [9] and WSDL [10] and more lightweight approaches such as REST or XML-RPC.

The current efforts of the SWS research community resulted in reference ontologies, such as OWL-S, WSMO and SAWSDL[2] as well as comprehensive frameworks to demonstrate the SWS approach (i.e. DIP project[3]). Whereas WSMO is intended to enable fully automated service matchmaking based on comprehensive semantic specifications of service capabilities, recent derivations of WSMO, like WSMO-Lite[4], MicroWSMO[5] and hRESTs[6], enable representation of rather lightweight service descriptions based on RDF.

Most recently, the Linked Services concept has been proposed based on Linked Data principles. Linked Data is a way to publish data on the Web in order for machines to automatically derive the meaning of the data. The Linked Data cloud contains a rich variety of alignments between external data, which makes it possible to create services that

[1] http://www.NoTube.tv/

[2] http://www.w3.org/2002/ws/sawsdl/

[3] DIP Project: http://dip.semanticweb.org

[4] http://cms-wg.sti2.org/TR/d11/v0.2/

[5] http://cms-wg.sti2.org/TR/d12/v0.1/

[6] http://knoesis.wright.edu/research/srl/projects/hRESTs

make use of relevant combinations [12]. We implement the STRB by integrating the available Linked data with the IRS-III semantic web services framework.

III. Use Cases and Challenges

In order to explain the role and functionalities of the STRB, we select and illustrate two use cases that were driven by the TV broadcast industry partners within the EU NoTube project.

A. *Personalized Semantic News*

The Personalized Semantic News use case describes how a user acquires news items from generic broadcast streams and obtains additional enriched news information by using a set of personalised news related services through the NoTube platform. The NoTube platform understands the meaning of video news items and the physical context in which news items are going to be shown. Based on this, the platform will apply criteria for matching and filtering the user profile and preferences to match the available news items. Figure 2 shows a possible scenario where a user asks his/her context-aware news-agent to search interesting news when he/she is using an iPhone and travelling by bus. He/she registered his/her profile to the agent and he/she prefers to use English and is generally interested in sports. The agent will invoke the STRB to get the interesting news data by discovering, selecting and invoking the suitable news services that match the user's context.

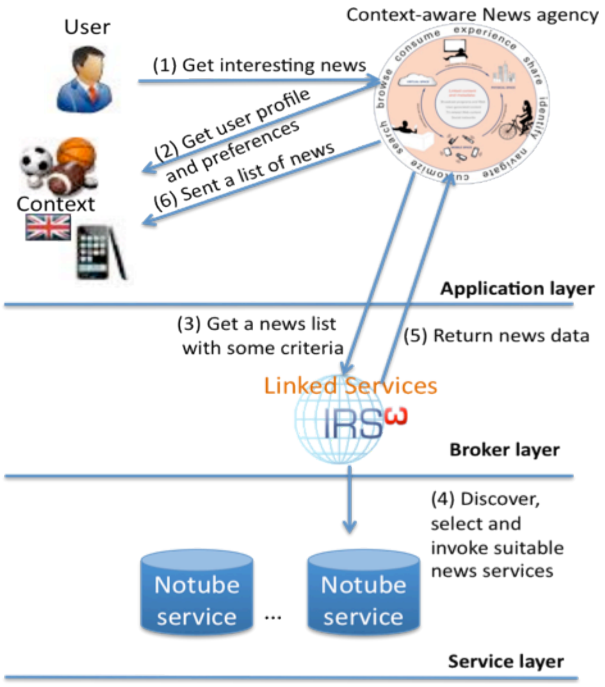


Figure 2. Personalized Semantic News use case.

B. *Personalized TV Guide*

This use case allows a user to send a request for getting EPG data with program recommendations and additional related information gained from Internet resources that are provided by a set of TV program enrichment services. The recommendations should be based on the context of the user, such as user activities, languages and personal interests. This information is stored by a User Profiling service. Thus, the core services that underpin this scenario are User Profiling services, EPG services, TV program recommendation services, and content enrichment services.

Unlike the first use case, the Personalized TV Guide scenario requires the broker to orchestrate a group of services according to an orchestration process shown in Figure 3. The orchestration process does not combine any concrete services at the beginning and only assigns services at runtime, which is the main difference contracting to current WSBPEL[11] technology (e.g. activeBPEL [7] and ODE[8]).

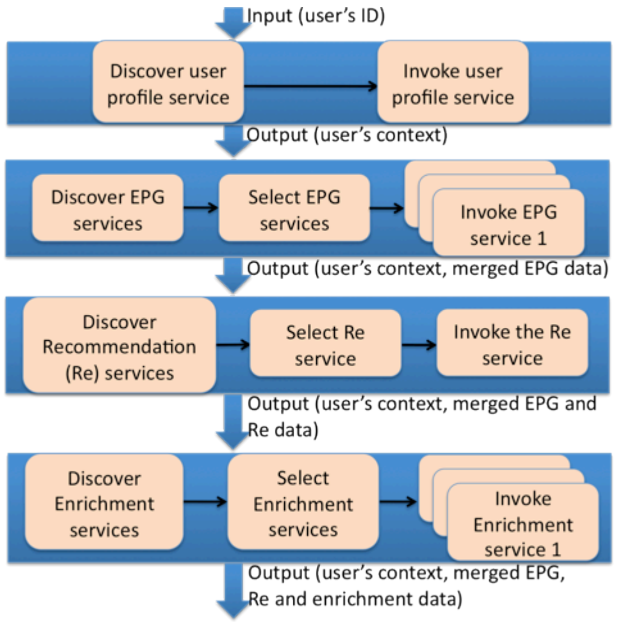

Input (user's ID)
Discover user profile service
Invoke user profile service
Output (user's context)
Discover EPG services
Select EPG services
Invoke EPG service 1
Output (user's context, merged EPG data)
Discover Recommendation (Re) services
Select Re service
Invoke the Re service
Output (user's context, merged EPG and Re data)
Discover Enrichment services
Select Enrichment services
Invoke Enrichment service 1
Output (user's context, merged EPG, Re and enrichment data)

Figure 3. Personalized TV Guide use case orchestration.

Thus, the additional challenge raised by this use case is how to dynamically orchestrate different services throughout discovery, selecting and invoking steps.

IV. Semantic TV Resources Broker Architecture

Our STRB solution integrates two semantic web technologies although they are currently still under development for improvement, namely, Linked Services and the IRS-III framework. Figure 4 shows the core development workflow in which the STRB operates. That is:

- Service providers annotate, register and publish their services into the Linked Services RDF repository that adapts Linked Data principles for linking services to their functional and non-functional annotations and other services in order to dynamically discover services. It is as simple as

[7] http://www.activepbel.org/

[8] http://www.ode.org/

searching a web page and automatically selecting services based on required properties described as SPARQL query.

- After service registration, the services repository will feed the new functional annotations into the IRS-III framework. The registered service can be discovered, selected, orchestrated by the IRS-III semantic execution environment.
- The Application Developer can request invocation URIs from the STRB for the required services to develop NoTube applications.

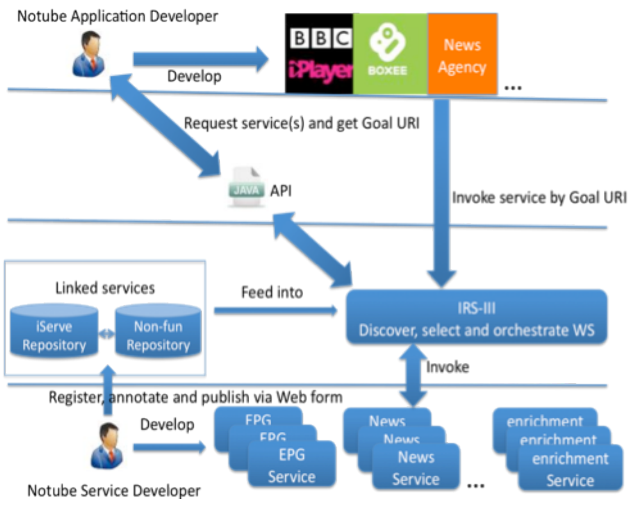

Notube Application Developer
BBC iPlayer
BOXEE
News Agency
Develop
Request service(s) and get Goal URI
API
Invoke service by Goal URI
Linked services
iServe Repository
Non-fun Repository
Feed into
IRS-III Discover, select and orchestrate WS
Invoke
Register, annotate and publish via Web form
Develop
EPG Service
News Service
enrichment Service
Notube Service Developer

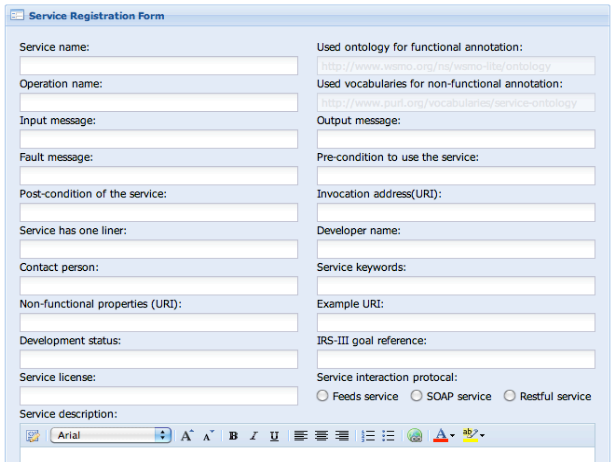
Figure 4. NoTube services development workflow.

A. *Linked Services*

The idea of Linked Services is inspired by the Linked Data movement. Linked Data is a way to publish data on the Web in order for machines to understand the explicit meaning of the data. The data is linked to other external data sets, and can in turn be linked from external data sets [2]. In this way, the data can be found and operated directly by machines. In another words, Linked data is an implementation standard of the Semantic Web.

Based on a similar idea, we publish "linked services" with their semantic descriptions on the Web. The basic principles are:

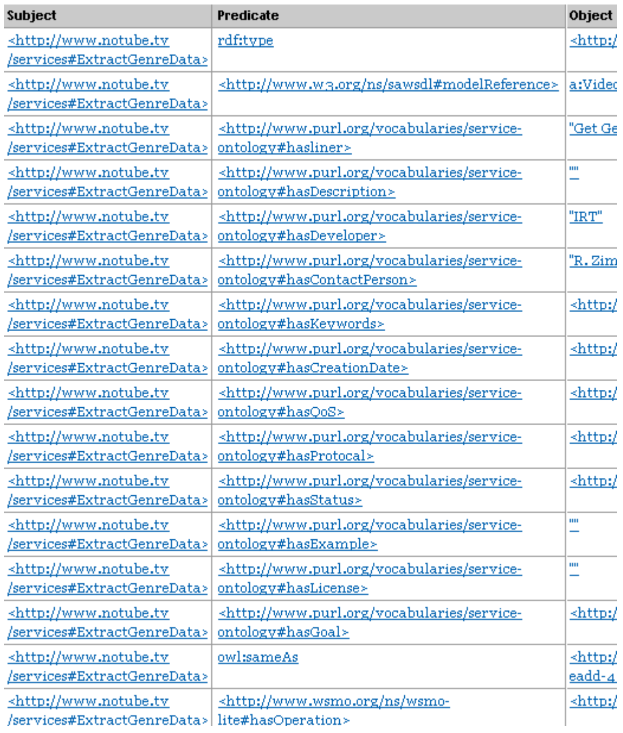
- Using WSMO-Lite and MicroWSMO as functional semantic description schema and using a number of domain ontologies as non-functional semantic description schema.
- Representing and persisting the semantic descriptions as RDF data stored in a Sesame RDF database. The database includes two divided RDF repositories: (1) iServe [3] for storing functional descriptions, such as invocation endpoint, input message and output message and so on, and (2) non-funServe for storing non-functional descriptions, such as IRS-III goal URI, QoS and keywords and so on.
- Allowing service providers to annotate publish their services using a web-based form UI (see Figure 5). Figure 6 shows an example of linked data for semantically describing a service after annotated and published by provider via the web-based form.
- Using a Similarity-based Conceptual Space approach proposed in [14] for semantic web service discovery and selection.



Service Registration Form

Service name:	Used ontology for functional annotation: http://www.wsmo.org/ns/wsmo-lite/ontology
Operation name:	Used vocabularies for non-functional annotation: http://www.purl.org/vocabularies/service-ontology
Input message:	Output message:
Fault message:	Pre-condition to use the service:
Post-condition of the service:	Invocation address(URI):
Service has one liner:	Developer name:
Contact person:	Service keywords:
Non-functional properties (URI):	Example URI:
Development status:	IRS-III goal reference:
Service license:	Service interaction protocal: ○ Feeds service ○ SOAP service ○ Restful service
Service description:	

Figure 5. Linked Services publishing form UI.



Subject	Predicate	Object
<http://www.notube.tv /services#ExtractGenreData>	rdf:type	<http:/
<http://www.notube.tv /services#ExtractGenreData>	<http://www.w3.org/ns/sawsdl#modelReference>	a:Videc
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasliner>	"Get Ge
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasDescription>	""
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasDeveloper>	"IRT"
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasContactPerson>	"R. Zim
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasKeywords>	<http:/
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasCreationDate>	<http:/
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasQoS>	<http:/
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasProtocal>	<http:/
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasStatus>	<http:/
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasExample>	""
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasLicense>	""
<http://www.notube.tv /services#ExtractGenreData>	<http://www.purl.org/vocabularies/service-ontology#hasGoal>	<http:/
<http://www.notube.tv /services#ExtractGenreData>	owl:sameAs	<http:/ eadd-4
<http://www.notube.tv /services#ExtractGenreData>	<http://www.wsmo.org/ns/wsmo-lite#hasOperation>	<http:/

Figure 6. An RDF data example of a linked service annotation.

B. *IRS-III framework*

IRS-III is a Semantic Web Service execution environment. It acts as "broker" – mediating between the goals of a client and relevant services that are deployed on the Web. IRS-III adopts the WSMO conceptual model of services, the ultimate aim of which is to be able to provide unambiguous models of services with a well-defined semantics, which can then be interpreted by a reasoner to enable automatic discovery, selection, composition, mediation, execution, and monitoring of services.

At runtime, IRS-III automatically discovers and invokes Web services suitable for a given client request, formulated as a goal instance. Thus, the first important part of the IRS-III for STRB is to define a service goal that allow user to automatically consume the service by invoking the goal URI which is stored as one piece of data in Linked service database. Listing 1 shows a simple example of EPG service goal and related semantic annotations. From the definition we can see that BBC-ZAPPER-EPG-BY-KEYWORD-AND-DATE-GOAL service has four input parameters (line 4: HAS-IPUT-ROLE) and all parameters are binding to SOAP string type (see from line 5 to 9). The output message only includes one parameter of has-epg-data, which is a SOAP string type as well (see line 11). We also can find that the service address is located at luisa.open.ac.uk:8080/axis/engineService171.jws (see from line 26 to 28).

The second important part is that IRS-III can orchestrate services based on services' WSMO semantic annotations. Listing 2 shows an example of orchestration of all English EPG services that are deployed in IRS-III. The orchestration process is assigned to a HAS-PROBLEM-SOLVING-PATTERN (line 2). The pattern (defined in from line 3 to the end) has an orchestration sequence ("orch-seq" in the list) and each individual sequence step assigns a concrete goal with input semantics to complete the task of the step (e.g. from line 6 to 10). Finally, all the services' output will be collected and integrated to become one single output (see line from 18 to 20).

```
1 (DEF-CLASS get-BBC-ZAPPER-EPG-BY-KEYWORD-AND-DATE-GOAL
2        (GOAL)
3         ?GOAL
4        ((HAS-INPUT-ROLE :VALUE has-method :VALUE has-keywords
:VALUE has-start-date :VALUE has-end-date)
5        (HAS-INPUT-SOAP-BINDING
          :VALUE
6         (has-method "string")
          :VALUE
7         (has-keywords "string")
          :VALUE
8         (has-start-date "string")
          :VALUE
9          (has-end-date "string"))
10          (HAS-OUTPUT-ROLE :VALUE has-epg-data)
11         (HAS-OUTPUT-SOAP-BINDING
12           :VALUE
13          (has-epg-data "string"))
14          (has-method :TYPE String)
15          (has-keywords :TYPE String)
16         (has-start-date :TYPE String)
17          (has-end-date :TYPE String)
18          (has-epg-data :TYPE String)
19          (HAS-NON-FUNCTIONAL-PROPERTIES
20           :VALUE
21           get-BBC-ZAPPER-EPG-BY-KEYWORD-AND-DATE-GOAL-NON-
FUNCTIONAL-PROPERTIES)))
…
…
…
22 (DEF-CLASS get-BBC-ENGIN-EPG-BY-KEYWORD-AND-DATE-WS-
PUBLISHER-INFORMATION
23    (PUBLISHER-INFORMATION)
24    ((HAS-ASSOCIATED-WEB-SERVICE-INTERFACE: VALUE
25   get-BBC-ENGIN-EPG-BY-KEYWORD-AND-DATE-INTERFACE)
26   (HAS-WEB-SERVICE-HOST :VALUE "luisa.open.ac.uk")
27   (HAS-WEB-SERVICE-PORT :VALUE 8080)
28   (HAS-WEB-SERVICE-LOCATION :VALUE    "/axis/enginService171.jws"
)))
```

Listing 1. Atomic Service goal definition code.

```
1 (DEF-CLASS get-AGGREGATED-ENGLISH-LANGUAGE-EPG-BY-KEYWORD-
AND-DATE-INTERFACE-ORCHESTRATION
        (ORCHESTRATION)
2         ((HAS-PROBLEM-SOLVING-PATTERN
          :VALUE
          get-AGGREGATED-ENGLISH-LANGUAGE-EPG-BY-KEYWORD-AND-
DATE-INTERFACE-ORCHESTRATION-PROBLEM-SOLVING-PATTERN)))


3 (DEF-CLASS get-AGGREGATED-ENGLISH-LANGUAGE-EPG-BY-KEYWORD-
AND-DATE-INTERFACE-ORCHESTRATION-PROBLEM-SOLVING-PATTERN
4         (PROBLEM-SOLVING-PATTERN)
5    ((has-body
     :value
6     ((orch-seq
      (achieve-goal 'ocml::get-BBC-ENGIN-EPG-BY-KEYWORD-AND-DATE-
GOAL
7                 (orch-get-input-role 'ocml::has-method)
8                (orch-get-input-role 'ocml::has-keywords)
9                 (orch-get-input-role 'ocml::has-start-date)
10                 (orch-get-input-role 'ocml::has-end-date)))
11      (orch-seq
12      (achieve-goal 'ocml::get-BBC-ZAPPER-EPG-BY-KEYWORD-AND-DATE-
GOAL
13                 (orch-get-input-role 'ocml::has-method)
14                 (orch-get-input-role 'ocml::has-keywords)
15                (orch-get-input-role 'ocml::has-start-date)
16                 (orch-get-input-role 'ocml::has-end-date)))
17     (orch-seq

18     (string-concatenate 'string
19                      (remove-close-metadata-tag (orch-get-goal-value get-
BBC-ENGIN-EPG-BY-KEYWORD-AND-DATE-GOAL))
20                      (remove-open-metadata-tag (orch-get-goal-value get-
BBC-ZAPPER-EPG-BY-KEYWORD-AND-DATE-GOAL))))

))))
```

Listing 2. Service orchestration code.

The Broker's functionalities are exposed through an HTTP-based REST API, which applications use to interact with the Broker. Primary among the functionalities of the Broker is the 'achieve-goal' method. In the IRS approach, Web services are semantically described and associated to "goal" representations. Goals are then exposed through the Broker's REST interface to allow them to be "achieved" by the client application. We have also implemented a Java API to the IRS-III Broker.

C. *IRS-III Java API*

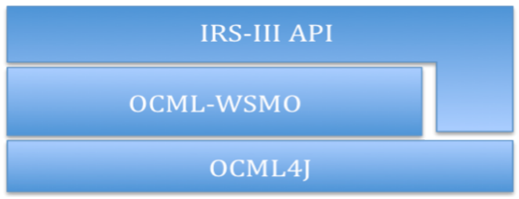


Figure 7. IRS-III Java API implementation blocks.

As Figure 7 shows, IRS-III Java API is implemented based on OCML-WSMO ontology and communicates to IRS-III via OCML4J [9] interface. Thus, the NoTube application developer can define a goal of using service through this IRS-III Java API. On runtime, the NoTube application, firstly, connects to IRS III via the API and executes the application goal definition for mapping a goal defined in IRS-III and retrieving it. Secondly, the application sends the goal to IRS III by invoking the function of achieving goal in IRS API. IRS III performs the discovery, selection, orchestration and invocation of Web services, and sending back the result to the application as a HTTP response.

- IRS-III Java API defines the programming interfaces of the IRS-III knowledge base. IRS-API supports a cache per concept, In other words, it allows using a caching mechanism per kind of instances. The API can retrieve, WSMO entities stored in IRS III, which are independent from the underlying communication mechanism.
- As WSMO is the ontology built for modelling Web services, OCML-WSMO provides a specific interface on top of OCML4J to facilitate the processing of the entities defined in WSMO, i.e. goals, mediators, Web services and non-functional properties.
- Since IRS-III is implemented using OCML, OCML4j helps manipulating OCML entities in IRS-III, such as class, function, instance, relation, ontology, as well as slot and logical expressions. In addition, XMLBeans3 is utilized for processing XML messages exchanged during communications between Java applications and the IRS server.

V. Conclusion and Future Work

In this paper, we described the implementation of a Semantic TV Resources Broker. The broker allows application developers within the TV domain to express application goals in human-readable format, and discovers, integrates and invokes the relevant services in order to fulfill these goals.

[9] http://technologies.kmi.open.ac.uk/ocml/

The technologies that we used are: (1) Linked service repositories for publishing services and annotating functional and non-functional properties; (2) the IRS-III framework including the Java API for service logical annotation based on WSMO, service invocation and orchestration through achieving-goal interface.

At present our work is at an early stage and we still need to evaluate the entire two-stage approach. In the future, we will conduct this evaluation by comparing our work to other related frameworks, for example, the Dino framework of Dynamic and Adaptive Composition of Autonomous Services [17], to analyze the efficiency, adaptability, scalability and other aspects.

Other future work includes elaboration on more advanced selection mechanisms, e.g. LSP-based selection method [15], into the implementation to test and compare the selection and composition performances.

References

[1] Dietze, S., and Domingue, J. (2009) Towards Context-aware Multimedia Processing through Semantic Web Services, in Proceedings of the seventh european conference on European interactive television conference (EuroITV 2009), ACM, pp. 129-132, Leuven, Belgium.

[2] Pedrinaci, C., Domingue, J., and Reto Krummenacher (2010) Services and the Web of Data: An Unexploited Symbiosis, Workshop: Linked AI: AAAI Spring Symposium "Linked data Meets Artificial Intelligence".

[3] Carlos Pedrinaci, Dave Lambert, Maria Maleshkova, Dong Liu, John Domingue, and Reto Krummenacher. Adaptive Service Binding with Lightweight Semantic Web Services. In Service Engineering: European Research Results (S. Dustdar and F. Li eds.). Springer (2010).

[4] Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C. (2006): IRS-III: A Broker for Semantic Web Services based Applications. Proceedings of the 5th International Semantic Web Conference (ISWC), Athens, USA.

[5] OWL-S Coalition: OWL-S 1.1 release. (2004). http://www.daml.org/services/owl-s/1.1/

[6] WSMO Working Group (2004), D2v1.0: Web service Modeling Ontology (WSMO). WSMO Working Draft, (2004). (http://www.wsmo.org/2004/d2/v1.0/).

[7] Wagner, M., Kellerer, W. 2004. Web services selection for distributed composition of multimedia content, Proceedings of the 12th annual ACM international conference on Multimedia, October 10-16, 2004, New York, NY, USA.

[8] World Wide Web Consortium, W3C: Simple Object Access Protocol, SOAP, Version 1.2 Part 0: Primer, (2003). (http://www.w3.org/TR/soap12-part0/).

[9] World Wide Web Consortium, W3C: Universal Description, Discovery and Integration: UDDI Spec Technical Committee Specification v. 3.0, (2003). http://uddi.org/pubs/uddi-v3.0.1-20031014.htm).

[10] World Wide Web Consortium, W3C: WSDL: Web services Description Language (WSDL) 1.1, (2001). (http://www.w3.org/TR/2001/NOTE-wsdl20010315)

[11] Advanced Open Standards for the Information Society (OASIS), WSBPEL: Web Service Business Process Execution Language 2.0, (2007). (http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html)

[12] Bizer, C., T. Heath, et al. (2009). "Linked data - The Story So Far." Special Issue on Linked data, International Journal on Semantic Web and Information Systems (IJSWIS).

[13] Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B., Pedrinaci, C.: “IRS-III: A broker-based approach to semantic Web services”, Jounal of Web Semant, pp. 109-132. Elsevier Science Publishers B. V, 2008.

[14] Dietze, S., Benn, N., Domingue, J., Conconi, A., and Cattaneo, F.: "Interoperable Multimedia Metadata through Similarity-based Semantic Web Service Discovery". In Proceedings of 4th International Conference on Semantic and Digital Media Technologies (SAMT '09), 2--4 December 2009, Graz, Austria.

[15] Yu, H.Q., Reiff-Marganiec, S., "A Method for Automated Web Service Selection", services, pp. 513-520, 2008 IEEE Congress on Services - Part I, 2008

[16] E. Motta, Reusable Components For Knowledge Modelling: Case Studies in Parametric Design Problem Solving. IOS Press, ISBN I 58603 003 5, 1999.

[17] A. Mukhija, A. Dingwall-Smith, and D. S. Rosenblum, 2007. QoS-Aware Service Composition in Dino. In Proceedings of the Fifth European Conference on Web Services (November 26 - 28, 2007). ECOWS. IEEE Computer Society, pp. 3-12.

[18] W3C Recommendation, SPARQL query language for RDF, 2008, (http://www.w3.org/TR/rdf-sparql-query/)