



NoTube

*Networks and ontologies for the transformation and unification of
broadcasting and the Internet*

FP7 – 231761

D5.1b Semantic TV Broker Requirements

Coordinator: S. Dietze (OU), N. Benn (OU)

With contributions from:

A. Conconi, F. Cattaneo (TXT), H. Qing Yu (OU),
M. Damova, M. Yankova (OT)

Quality Assessor: Ronald Siebes

Quality Controller: Lora Aroyo

Document Identifier:	NoTube/2010/D5.1b
Version:	1.0
Date:	28/02/2010
State:	Final
Distribution:	PU

EXECUTIVE SUMMARY

This deliverable describes the envisaged requirements for the Semantic TV Broker component. These requirements are based on the use cases obtained by the WP7 tasks (which are described in deliverables D7a.1, D7b.1, and D7c.1) and the architectural requirements posed by WP6 Task 6.1 (which are described in deliverable D6.1). In addition, this deliverable presents an overview of current Semantic Web Services (SWS) technology applied to the Semantic TV Broker.

Similar to deliverable D6.1 on NoTube specifications and overall architectural design, this deliverable consists of two releases: D5.1a, which was submitted in M3 and this current document, D5.1b, which refines the earlier release. The goal of the first release was to narrow the requirements for the Semantic TV Broker and to set the basis for the development work scheduled for the first year of the project. The goal of this current release is to report on further refinement of requirements based on further discussions with use case partners and service providers within the project, and based on our early prototyping work.

DOCUMENT INFORMATION

IST Project Number	FP7 - 231761	Acronym	NoTube
Full Title	Networks and ontologies for the transformation and unification of broadcasting and the Internet		
Project URL	http://www.notube.eu/		
Document URL			
EU Project Officer	Leonhard MAQUA		

Deliverable	Number	5.1b	Title	Semantic TV Broker Requirements
Work Package	Number	5	Title	Semantic TV Resource Broker

Date of Delivery	Contractual	M 13	Actual	
Status	version 1.0		final	<input checked="" type="checkbox"/>
Nature	prototype <input type="checkbox"/> report <input checked="" type="checkbox"/> dissemination <input type="checkbox"/>			
Dissemination level	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			

Authors (Partner)	TXT, OU, OT			
Responsible Author	Name	Conconi, Cattaneo, Dietze	E-mail	s.dietze@open.ac.uk
	Partner	TXT, OU	Phone	+44 (0)1908-858217

Abstract (for dissemination)	<p>This deliverable further details the requirements as well as some preliminary design for the Semantic TV Broker component. In that, this deliverable represents an extension of D5.1a. Requirements were derived from the use cases (WP7), the technical WPs (in particular WP1, 3 and 4) and the architectural aspects posed by WP6. In addition, it presents an overview of current SWS technology applied to the Semantic TV Broker in the light of the technologies presented in D6.1 and a preliminary prototypical implementation.</p> <p>While the initial version of this deliverable – D5.1a released in M3 – posed a very preliminary set of requirements and an abstract classification of services, these are elaborated here in greater detail involving also an initial set of services being provided by individual NoTube WPs. In that, this deliverable acts as facilitator for the implementation work within WP5.</p>
Keywords	SWS Broker, SW, SWS, Requirements

Version Log			
Issue Date	Rev. No.	Author	Change
31/10/09	0.1	S. Dietze	Preliminary ToC and adapted content from D5.1a
30/11/09	0.3	H. Yu	Added content to Section 5
04/12/09	0.4	S. Dietze	Added content to Section 6
10/12/10	0.5	N. Benn	Added content to Section 5
13/01/10	0.6	S. Dietze	Changes to Section 3.3 plus minor edits throughout
15/01/10	0.7	F. Cattaneo, A. Conconi	Added content to Section 4 on NoTube services
08/02/10	0.9	S. Dietze, N. Benn, H. Yu	Final grammatical edits for submission to internal reviewer.
19/02/10	1.0	S.Dietze, N.Benn, H. Yu	Amendments according to internal reviewer comments

PROJECT CONSORTIUM INFORMATION














Participants		Contact
Vrije Universiteit Amsterdam		Guus Schreiber Phone: +31 20 598 7739/7718 Email: schreiber@cs.vu.nl
British Broadcasting Corporation		Libby Miller Phone: Email: libby.miller@bbc.co.uk
Asemantics SRL Uninomiale		Alberto Reggiori Phone: +39 0639 7510 78 Email: alberto@asemantics.com
Engin Medya Hizmetleri A.S.		Ron van der Heiden Phone: +31 6 2003 2006 Email: ron@engin.tv
Institut fuer Rundfunktechnik GmbH		Christoph Dosch Phone: +49 89 32399 349 Email: dosch@irt.de
Ontotext AD		Atanas Kiryakov Phone: +35 928 091 565 Email: naso@sirma.bg
Open University		John Domingue Phone: +44 1908 655 014 Email: j.b.domingue@open.ac.uk
RAI Radiotelevisione Italiana SPA		Alberto Morello Phone: +39 011 810 31 07 Email: a.morello@rai.it
Semantic Technology Institute International		Lyndon Nixon Phone: +43 1 23 64 002 Email: lyndon.nixon@sti2.org
Stoneroots B.V.		Annelies Kaptein Phone: +31 35 628 47 22 Email: annelies.kaptein@stoneroots
Thomson Grass Valley France SA		Raoul Monnier Phone: +33 2 99 27 30 57 Email: raoul.monnier@thomson.nett
TXT Polymedia SPA		Sergio Gusmeroli Phone: +39 02 2577 1310 Email: sergio.gusmeroli@txtgroup.com
KT Corporation		Myoung-Wan Koo Phone: +82 2 526 5090 Email: mwkoo@kt.com

TABLE OF CONTENTS

LIST OF FIGURES	7
LIST OF TABLES	8
LIST OF ACRONYMS.....	9
1. INTRODUCTION	10
1.1. SCOPE OF THIS DELIVERABLE.....	10
2. SEMANTIC WEB SERVICES (SWS) OVERVIEW.....	11
2.1. SWS TECHNOLOGIES.....	11
2.2. IRS-III SWS EXECUTION ENVIRONMENT.....	12
3. SWS AS PART OF THE NOTUBE ARCHITECTURE	14
3.1. NoTUBE SERVICE-ORIENTED APPROACH - OVERVIEW.....	14
3.2. SWS BROKER ROLE	15
3.3. SUPPORTING APPLICATIONS AND DEVELOPERS THROUGH SERVICE SEMANTICS	16
<i>Support of distributed NoTube developers with light-weight service annotations</i>	<i>16</i>
<i>Support of application automation with Semantic Web Service brokerage via the WP5 Broker..</i>	<i>17</i>
4. NOTUBE SERVICES	19
4.1. IMPLEMENTED SERVICES.....	19
4.2. SERVICES UNDER DEVELOPMENT.....	22
4.3. SERVICES IN PLANNING.....	25
5. NOTUBE USE CASE SCENARIOS FOR 1ST PROTOTYPES	29
5.1. WP7A SCENARIO – PERSONALISED SEMANTIC NEWS (PSN)	29
<i>Current state of integration of the Broker into 1st prototypes.....</i>	<i>29</i>
5.2. WP7B SCENARIO – PERSONALISED TV GUIDE WITH ADAPTIVE ADVERTISING	30
<i>Current state of integration of the Broker into 1st prototypes.....</i>	<i>31</i>
5.3. WP7C SCENARIO – INTERNET TV IN THE SOCIAL WEB	31
<i>Current state of integration of the Broker into 1st prototypes.....</i>	<i>32</i>
5.4. DERIVED REQUIREMENTS.....	32
6. NOTUBE SEMANTIC TV BROKER PROTOTYPE	36
6.1. OVERALL ARCHITECTURE	36
<i>Video metadata services.....</i>	<i>36</i>
<i>EPG data services.....</i>	<i>37</i>
6.2. API.....	39
<i>Achieve goal.....</i>	<i>39</i>
<i>Simple “achieve-goal” example.....</i>	<i>39</i>
6.3. INTEGRATED SWS GOALS.....	40
<i>Get-video-metadata-goal: selection and invocation of video metadata query services.....</i>	<i>40</i>
<i>Get-EPG-metadata-goal: selection, orchestration, and invocation of EPG query services.....</i>	<i>41</i>
7. CONCLUSION	44
8. REFERENCES	45

List of Figures

Figure 1 – NoTube General Architecture.....	14
Figure 2 – SWS-based Broker as part of the NoTube Architecture.....	17
Figure 3 – Interface to SWS Broker.....	18
Figure 4 - Conceptual diagram of WP5-WP7a integration.....	30
Figure 5 - Conceptual diagram of WP5-WP7b integration.....	31
Figure 6 - WP5 proof-of-concept prototype architecture.	36
Figure 7 - WP5 proof-of-concept prototype AJAX interface presenting a set of matching video metadata records.	38
Figure 8 - AJAX interface for representing the intermediate results of the service matchmaking.	38
Figure 9 - Conceptual overview of Get-Video-Metadata-Goal and corresponding SWS.	40
Figure 10 - Conceptual overview of Get-EPG-Metadata-Goal and corresponding SWS.....	41



List of Tables

Table 1 - Services that are currently implemented and available from NoTube partners.....	19
Table 2 - Services that are currently under development by the NoTube partners.	22
Table 3 - Services being planned by NoTube partners	25
Table 4 - Scenario-driven requirements for the Semantic TV Broker.....	32

List of Acronyms

<u>Acronym</u>	<u>Description</u>
AMI	Ambient Intelligence
EPG	Electronic Program Guide
hRESTs	HTML for RESTful Services
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
RIF	Rule Interchange Format
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
STB	Set-Top Box
SWS	Semantic Web Services
UDDI	Universal Description, Discovery and Integration
WADL	Web Application Description Language
WSDL	Web Services Description Language
WSMO	Web Service Modeling Ontology

1. Introduction

This deliverable aims at gathering functional and architectural requirements for the Semantic TV Broker, which is the component responsible for handling semantic annotations of services in the NoTube architecture. In addition, we report on current SWS technologies that will be used for the purpose of the project, i.e., the unification of broadcasting and the Internet in the context of internet-based TV provisioning, and the context-adaptive delivery of broadcasting content.

In order to facilitate content adaptation – when following service-oriented approaches that are potentially carried out through a sequence of Web Services invocations [7] – the central challenge lies in identifying the most appropriate services for a given request within a certain situation. In this respect, the Semantic TV Resource Broker (henceforth, the “Broker”) will enable the dynamic discovery and orchestration of the most appropriate services for a given context. Particularly with respect to personalised delivery of media content it is an important requirement to deliver the most appropriate media services to the end user.

However, the definition and discovery of the most appropriate service or resource representation for a given context remains a challenging task for SWS research. Moreover, the domain independence and general-purpose nature of SWS technologies requires us to provide an appropriate formalisation and integration of domain specific vocabularies for describing digital media and TV scenarios.

1.1. Scope of this deliverable

The main service interaction requirements will be defined from the use cases (WP7) and the NoTube architecture (WP6). Furthermore, services defined and provided in WP1, WP2, WP3 and WP4 with respect to TV metadata, context and user modelling or vocabularies alignments will also have an impact on the final set of requirements. Since the overall requirements for the NoTube architecture are elicited within D6.1 / WP6, the architectural requirements of the Broker will be derived from the ones defined in WP6.

2. Semantic Web Services (SWS) overview

Semantic Web Services (SWS) are ontological descriptions of Web services in terms of their capabilities, interfaces and non-functional properties. Tasks such as Web service discovery, composition, and invocation can be automated to a great extent by applying semantic technologies. For example, semantics allow programs to access services through a machine-processable description of the service's capabilities rather than as a direct service endpoint. The use of semantics thus forms a scalable access layer over Web service data and processes. In deliverable D6.1, a review of Semantic Web and Web Service technologies as well as standards are presented, which will be commonly referred to in this deliverable. In this Section, we present a number of SWS technologies, and in particular, the IRS-III SWS execution environment, which implements the role of the Broker.

2.1. SWS technologies

Semantic Web Services technologies enable the automatic discovery, selection and composition of distributed services for a particularly expressed user request. Note that the term service here refers to software functionalities which are exposed to and accessible through the Web (i.e. based on HTTP). In that, a Web service might utilise standard Web service technology such as SOAP [8], UDDI [9] and WSDL [10] but also more light-weight approaches such as REST or XML-RPC. Semantic Web Services are being deployed to facilitate interoperability and to increase the degree of automation in a wide range of applications from different domains, such as eLearning [2] or business process management [6].

Current results of SWS research are available in terms of reference ontologies, such as **OWL-S** [4], **WSMO**¹ [5], and **SAWSDL**² as well as comprehensive frameworks such as those produced by the DIP project³). These reference ontologies and frameworks are intended to enable fully automated service matchmaking based on comprehensive semantic specifications of service capabilities.

Recent derivations of WSMO, enable representation of rather light-weight service descriptions based on RDF and the hREST microformat:

- **WSMO-Lite**⁴: Lightweight Descriptions of Services on the Web – a lightweight set of semantic service descriptions in RDFS that can be used for annotations of various WSDL elements using the SAWSDL annotation mechanism. WSMO-Lite exploits the standard languages of W3C including RDF and RDFS as well as various extensions of those languages such as OWL, WSML and RIF for semantic service descriptions.
- **MicroWSMO**⁵: Semantic Annotations for RESTful Services – a semantic annotation mechanism for RESTful Web services based on the hRESTs microformat.
- **hRESTs**⁶ (HTML for RESTful Services) - microformat for machine-understandable descriptions of Web APIs, backed by a simple service model. The hRESTs microformat describes main aspects of services, such as operations, inputs and outputs. Also available are two extensions of hRESTs: SA-REST, which captures the facets of public APIs important for mashup developers, and MicroWSMO, which provides support for semantic automation.

¹ <http://cms-wg.sti2.org/TR/d1/v1.0/>

² <http://www.w3.org/2002/ws/sawSDL/>

³ DIP Project: <http://dip.semanticweb.org>

⁴ http://cms-wg.sti2.org/TR/d11/v0.2/20090310/d11v02_20090310.pdf

⁵ http://cms-wg.sti2.org/TR/d12/v0.1/20090310/d12v01_20090310.pdf

⁶ <http://knoesis.wright.edu/research/srl/projects/hRESTs>

While the above mentioned lightweight approaches are less costly to apply, they envisage a much lower degree of automation and do not facilitate comprehensive matchmaking scenarios as foreseen by more complex frameworks such as WSMO.

2.2. IRS-III SWS execution environment

IRS-III⁷ [1] is a Semantic Web Service platform developed at KMi, UK. IRS-III adopts the WSMO conceptual model, thus implementing the following top-level knowledge-based components:

- **Ontologies.** Provide the foundation for semantically describing data in order to achieve semantic interoperability and are used by the three other elements;
- **Goal.** Describes the task that a service requester expects a web service to fulfil. In this sense they express the client request;
- **Web Service.** Describes the capability of an existing deployed Web Service. The description also outlines how Web Services communicate (choreography) and how they are composed (orchestration);
- **Mediator.** Describes connections between the components above and represent the type of conceptual mismatches that can occur. In particular, four types of mediators are provided: oo-mediators link and map between heterogeneous ontologies; wg-mediators connect web services to goals; gg-mediators link different goals and ww-mediators connect different web services.

At runtime, based on a SWS library consisting of the above mentioned knowledge entities, IRS-III automatically discovers and invokes Web services suitable for a given request. In that, based on a request which captures a desired outcome (Goal) and is provided by a particular SWS consumer, IRS-III proceeds through the following steps:

1. Discovery of potentially relevant Web services.
2. Selection of set of Web services which best fit the incoming request.
3. Invocation of the selected Web services whilst adhering to any data, control flow and Web service invocation constraints.
4. Mediation of mismatches at the data or process level.

⁷ <http://technologies.kmi.open.ac.uk/irs/>

The following describes the main functionalities provided by IRS-III to build and deploy Semantic Web Services-based applications, i.e. to enable automated SWS discovery and orchestration as described above:

- **Using Domain Ontologies.** The concepts involved in the application scenario which are used to describe client requests and Web Service capability are provided by means of domain ontologies.
- **Describing Client Requests as Goals.** The request for a service can be expressed from a consumer viewpoint and represented as a goal.
- **Semantically Describing Deployed Web Services.** The capability of a Web Service includes the types of inputs and outputs, and logical expressions for applied restrictions. This capability description can also include various other aspects such as orchestration and choreography.
- **Resolving Conceptual Mismatches.** Mediator descriptions can be used to declare a mediation service or mapping rules that will be used to provide alignment between goals, web services and domain ontologies.
- **Publishing Semantically Described Web Services.** Once a semantic description has been created for a deployed Web Service as above, it can be registered into the IRS-III repository for goal-based invocation.

3. SWS as part of the NoTube architecture

In this section, we provide an overview on the NoTube architecture and the particular role of the Broker.

3.1. NoTube service-oriented approach - overview

The general architecture described in D6.1, shown in Figure 1, is composed of two main building blocks: the Service Provider side and the Home Ambient side.

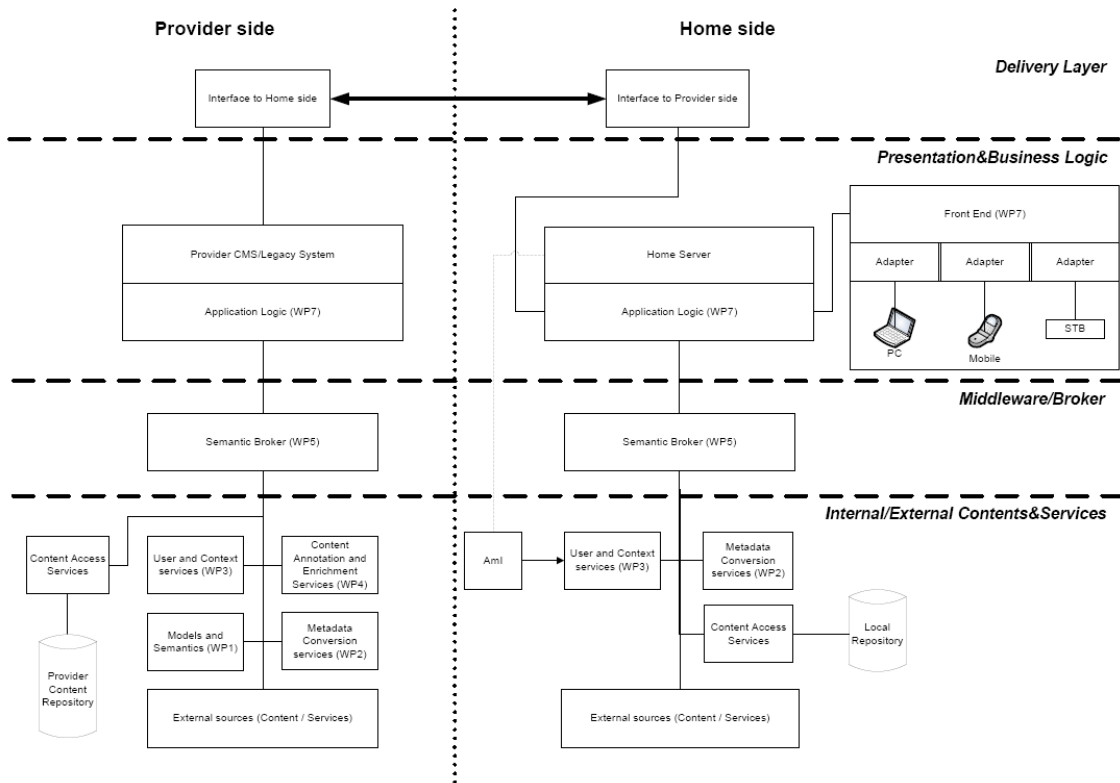


Figure 1 – NoTube General Architecture

The Provider-side block, on the left of the above picture, is aimed at the preparation of TV content to be delivered to the end-user, living at the home side. This preparation includes

- ingestion of TV contents
- production of content-related metadata (annotation and enrichment)
- implementation of providers' business policies
- application of generic user-context profile categories.

The Home side block, on the right of the picture, represents the physical place where the end-user lives and where the ambient intelligence (Aml) is implemented.

The NoTube platform receives content from the provider in order to:

- apply smart filtering basing on user's personal preference and context
- deliver the final output to the chosen channel (i.e.: STB, PDA, etc.).

Please note, that apart from the end-user devices, all components are regarded as server-side components which expose their functionalities. The separation between Provider- and Home side in Figure 1 just highlights the envisaged functionalities rather than the physical location. The defining feature of the NoTube architecture is that it follows a SOA approach –

i.e. all functionalities will be exposed by means of services. In this regard, the Broker will play a significant role with respect to service discovery and orchestration.

3.2. SWS Broker role

As can be seen in Figure 1, the Broker is located in the middleware layer, between the application logic and internal/external services. The Broker will provide common Brokering services such as discovery, mediation, and goal-achievement/invoke to both the provider side and home side application logic and to provide one single entry point.

To understand its importance, we start from the basic assumption that IT partners in the consortium are going to perform research and develop a set of services. Such services have a common knowledge-base identified by the WP1, namely "Models and Semantics". Services from a Provider perspective (Section 3.1) have been grouped in:

- User and Context (general profiling) – WP3
- Content Annotation and Enrichment – WP4
- Metadata conversion – WP2

In addition, it is envisaged to make use of external and publicly available services for the needs of NoTube. It appears reasonable and somewhat obvious that each of these categories requires individual effort in order to develop and expose the corresponding services. This is translated into an initial analysis and design phase, followed by the technical development. In particular, the latter is foreseen to be accomplished leveraging on the individual know-how of each involved partner and technologies that may involve specific hardware and software environments as well.

Finally, a set of different, heterogeneous environments – distributed on a network – is provided, each offering different services that do not necessarily link one to another neither physically nor semantically. Such scenario may recall in the reader with some technical knowledge on the principles of a typical SOA as envisaged by NoTube.

Thus, the envisaged SWS Broker functionality can be summarised through the following key roles:

- R1. Abstraction from actual Web service implementations
- R2. Automated discovery of appropriate services
- R3. Mediation of heterogeneities between distinct services

Given the heterogeneity of available services, these need to be discovered (R2), combined and orchestrated in a smart way regardless of the abovementioned factors. There should be also a mechanism that classifies the underlying services and possibly composes them to reach higher-level goals in the purpose of the considered application scenario. Such complex service composition usually also involves the mediation of heterogeneities (R3). Moreover, by abstracting from underlying service implementations by means of semantic annotations, the SWS Broker provides a single entry point to the upper level, the application layer (R1).

Going through the 3 foreseen use cases (WP7) for the NoTube project we've designed a general high-level architecture. However it seems reasonable to expect different behaviours for the different prototypes with respect to the specific requirements that each one has. This behaviour we've described is of course embedded into the considered application logic together with the legacy systems exploited but it's in general based on top of the services provided at a lower level by the SWS Broker. These services include the underlying low-level services we've already discussed together with higher-level services obtained by combining them using a semantic fashion.

With respect to and from the preliminary overview of the three test cases (see D6.1 Chapter 3 for more details), we've identified as an additional common requirement of the possibility to leverage on external contents and services in order to enrich the locally stored contents and maybe provide smart suggestions based on reasoning algorithms. Here again the SWS Broker could leverage on its natural ability to discover and classify new services in order to enrich the semantic computation with contents potentially coming from sources over the Internet – i.e.: social feeds, advertising, TV guides, etc.

From the Home Ambient perspective, the only difference is that the local low-level services are just related to:

- User and Context (personal profiling and ambient intelligence)
- Metadata conversion

Content Annotation and Enrichment services have not been listed as part of the Home Ambient environment, since they're based on the idea of adding metadata to raw contents (i.e.: multimedia streams, tv guides, etc.) and should not be confused with the retrieval of additional contents/services from external sources. Annotation services, designed and developed in WP4, are expected to be part of the content packaging phase that takes place at the Service Provider side.

Apart from the service collection and classification, from a strictly functional point of view the importance of the Broker, here, is to provide intelligent filtering of the local metadata through the services provided at user level. This is based on the assumption that the SWS Broker is expected to allow the automatic discovery and execution of services for a given goal request.

In the Home Ambient the main request is the customisation of the contents received from the Service Provider in the light of the user personal preferences together with the chosen delivery channel. Again, external services and contents are included as well as potential sources for metadata augmentation.

3.3. Supporting applications and developers through service semantics

Basically, service semantics will be exploited within WP5 to support two major goals:

- G1. Support of distributed NoTube developers with light-weight service annotations
- G2. Support of application automation with Semantic Web Service brokerage via the WP5 Broker

Support of distributed NoTube developers with light-weight service annotations

While the NoTube development takes place in a highly distributed setting and follows service-oriented principles, which involves the reuse of public and WP-specific services, NoTube developers need to be provided with smart means to find and identify appropriate services and data sources for their specific applications. Hence, as an initial step, light-weight service semantics will be produced to support the NoTube SOA developers in finding and re-using appropriate services. Therefore, light-weight service schemas such as WSMO-Light will be utilised and refined for the needs of NoTube to allow service providers to document services in RDF what forms the basis for a more structured search for relevant services by potential service consumers.

In addition, such service annotations can be gradually enriched in order to produce more comprehensive and formal service specifications, which are the basis for automated service brokerage, i.e., discovery and orchestration of services via a service broker (G.2).

Support of application automation with Semantic Web Service brokerage via the WP5 Broker

The overall aim of the SWS Broker is to provide application systems with the ability to invoke, discover, compose and mediate services based on their semantic annotations. Note that the SWS Broker produced in WP5 just stores and deploys semantic annotations of services while the actual Web services are neither part of the SWS Broker nor are these produced within WP5. In that, one assumption of the work within WP5 is, that services (Section 4) are provided by NoTube partners and are accessible through HTTP.

By annotating existing services via standardized ontologies we abstract from the Web Service implementation, ensuring a high level of autonomy and flexibility – that is, the client need not worry about changes in how the underlying service is implemented. It is also possible to semantically describe the requests or goals to be achieved. In Figure 2 we illustrate how the Broker can be used by applications in order to invoke existing Web Services. Goals are achieved by sending so-called goal requests to the Broker platform, which are resolved by automatic discovery and execution of available services based on their semantic annotations.

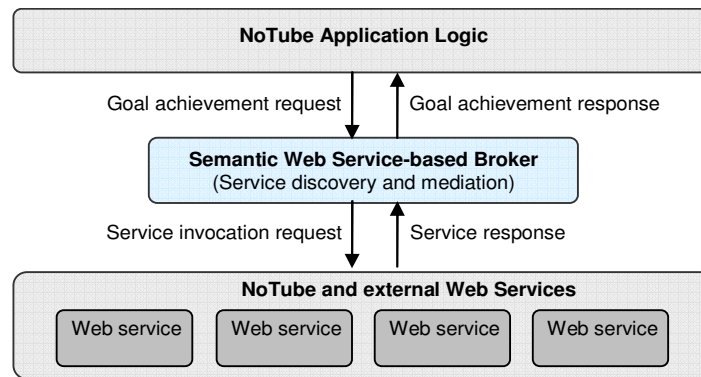


Figure 2 – SWS-based Broker as part of the NoTube Architecture

This highly generic version is particularly supported by the SWS technology introduced in Section 2.3, in that it allows the Broker to discover and orchestrate a set of services which suit the needs of a given goal. Note, while goals, i.e. WSMO Goals, are notions represented semantically, i.e. within the Broker, the implemented Web Services themselves represent external software entities, provided by either other NoTube components/WPs or external providers. In that, the Broker will make use of the services provides by WP1, WP2, WP3 and WP4 and will integrate with the architectural components and interfaces provided by WP6 and WP7.

The upper level application logic can be developed independently from the implementation specifics of the underlying software functionalities by simply requesting particular computations by means of goal-achievement requests. The Broker resolves such a request by discovering the most appropriate services for a given request, handling their execution and if desired, mediating mismatches occurring during execution-time. Note, service orchestrations might be created either within the SWS-based Broker but also the upper layers of the NoTube architecture, such as the application logic, dependent on the actual need and abstraction level.

More specifically, in Figure 3 we illustrate a number of services that can be available from the Broker API to the application layer.

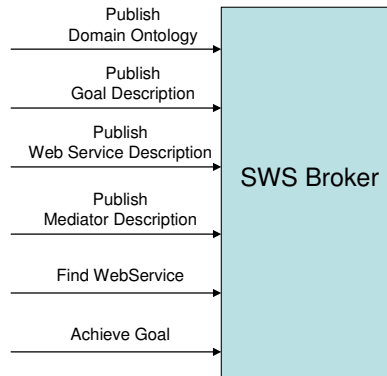


Figure 3 – Interface to SWS Broker

The Broker acts on the semantic descriptions of domain data, Goals, Web Services and Mediators that must be published into the Broker library. Requests for finding Web services or achieving Goals can then be sent to the Broker.

The next section introduces a number of services, which vary in their development status from “implemented” to “planned”, and which are provided by the technical work packages in order to be exposed through the Broker.

4. NoTube services

Currently, the development of NoTube services has three stages, namely, implemented services (services are already available to be tested and invoked), “under development” services (services have started to be implemented, but have not been completed yet) and planned services (services are not in development at the moment, but will be available in the future). NoTube services are also classified by their functionalities, such as EPG service, enrichment services and so on. Detailed information about the classification of services according to functionality can be found in deliverable D6.1b. In this section, NoTube services are listed by their stage of development, as shown in Table 1 – Table 3.

4.1. Implemented services

Table 1 - Services that are currently implemented and available from NoTube partners.

Name	Description	Provider	URI	Interface
ViewAllTVShow Recordings	Returns the list of all scheduled recordings with their state	TXT	http://demos.txt.it/Notube/HA_API/Ingestion_Service.smx?op=ViewAllTVShowRecordings	SOAP
AddTVShowRecording	Permits to schedule a new recording	TXT	http://demos.txt.it/Notube/HA_API/Ingestion_Service.smx?op=AddTVShowRecording	SOAP
ReadTVShowRecordingDetails	Returns the description of a single TVShow recorded or scheduled	TXT	http://demos.txt.it/Notube/HA_API/Ingestion_Service.smx?op=ReadTVShowRecordingDetails	SOAP
DeleteTVShowRecording	Deletes an already scheduled Recording, if the TVShow is already recorded no data will be deleted	TXT	http://demos.txt.it/Notube/HA_API/Ingestion_Service.smx?op=DeleteTVShowRecording	SOAP
UpdateTVShowRecording	Updates the already stored data of a TVShow recording that is not already started or finished.	TXT	http://demos.txt.it/Notube/HA_API/Ingestion_Service.smx?op=UpdateTVShowRecording	SOAP
Add scheduling	Allows to add a scheduling.	AS	http://moth.notube.tv:9998/scheduler/scheduling/add	REST
Remove scheduling	Allows to remove the scheduling for a certain tubelet.	AS	http://moth.notube.tv:9998/scheduler/scheduling/remove/{tubelet}	REST
Get schedulings	Allows to retrieve all schedulings set for all tubelets.	AS	http://moth.notube.tv:9998/scheduler/schedulings	REST
Get scheduling of a specific tubelet	Allows to retrieve the schedulings set for a specific tubelet.	AS	http://moth.notube.tv:9998/scheduler/scheduling/{tubelet}	REST
Start the scheduler	Allows to start the scheduler.	AS	http://moth.notube.tv:9998/scheduler/start	REST
Stop the scheduler	Allows to stop the scheduler.	AS	http://moth.notube.tv:9998/scheduler/stop	REST

Get the status of the scheduler	Allows to retrieve the status of the scheduler	AS	http://moth.notube.tv:9998/scheduler/status	REST
Get all pipelines	Returns the list of pipelines registered in the container.	AS	http://moth.notube.tv:9998/pipelinecontainer/pipelines	REST
Get the default triple pipeline	Returns the default pipeline that consumes RDF triples exposed by the container.	AS	http://moth.notube.tv:9998/pipelinecontainer/defaultTriplePipeline	REST
Get the default object pipeline	Returns the default pipeline that consumes objects exposed by the container.	AS	http://moth.notube.tv:9998/pipelinecontainer/defaultObjectPipeline	REST
Delete specific pipeline	Deregisters a pipeline from the container.	AS	http://moth.notube.tv:9998/delete/{pipelineName}	REST
Process RDF data with a specific pipeline	Start the processing of a bunch of RDF data with the default triple pipeline.	AS	http://moth.notube.tv:9998/process/{pipelineName}	REST
Process RDF data with the default triple pipeline	Start the processing of a bunch of RDF data with the default triple pipeline.	AS	http://moth.notube.tv:9998/Process	REST
Process a specific tubelet	Processes a tubelet from the container.	AS	http://moth.notube.tv:9998/tubeletcontainer/process/{identifier}	REST
Get all tubelets	Returns the list of tubelets registered in the container.	AS	http://moth.notube.tv:9998/tubeletcontainer/tubelets	REST
Get a specific tubelet	Returns a specific tubelet exposed by the container.	AS	http://moth.notube.tv:9998/tubelets/{identifier}	REST
Delete a specific tubelet	Deregisters a tubelet from the container.	AS	http://moth.notube.tv:9998/tubelets/{identifier}	REST
Upload a tubelet	Uploads a tubelet in the container.	AS	http://moth.notube.tv:9998/upload	REST
Get the delta of a specific user by a specific tubelet	Get the delta of information related to a specific couple (service, user).	AS	http://moth.notube.tv:9998/deltamanager/{tubelet}/{user}	REST
Register a user	Allows to register a user.	AS	http://moth.notube.tv:9998/triplestorage/store	REST
Get all users	Allows to retrieve all registered users.	AS	http://moth.notube.tv:9998/usermanager/users	REST
Get user by id	Allows to retrieve the user with given id.	AS	http://moth.notube.tv:9998/usermanager/{id}	REST
Get users by name and surname	Allows to retrieve the userIDs of the users with given name and surname.	AS	http://moth.notube.tv:9998/usermanager/{name}/{surname}	REST
Delete a user	Allows to deregister a user.	AS	http://moth.notube.tv:9998/usermanager/{userid}	REST

Register a user credential with an HTTP Basic Authenticator	Allows to register a user credential.	AS	http://moth.notube.tv:9998/usermanager/credentials/basicauth/{userid}/{service}/{username}/{password}	REST
Register a user credential with an anonymous authenticator	Allows to register a user credential.	AS	http://moth.notube.tv:9998/usermanager/credentials/anonymous/{userid}/{service}/{username}	REST
Delete a user credential	Allows to deregister a user credential.	AS	http://moth.notube.tv:9998/usermanager/credentials/{userid}/{service}	REST
Get all user credentials	Allows to retrieve all user credentials of registered services of a specific user.	AS	http://moth.notube.tv:9998/usermanager/credentials/{userid}	REST
Get all user services	Allows to retrieve the list of services to which at least one user is registered.	AS	http://moth.notube.tv:9998/usermanager/credentials/services	REST
SELECT	Perform a SPARQL query	AS	http://moth.notube.tv:9998/triplestorage/sparql	REST
CONSTRUCT	Perform a SPARQL query	AS	http://moth.notube.tv:9998/triplestorage/sparql	REST
DESCRIBE	Perform a SPARQL query	AS	http://moth.notube.tv:9998/triplestorage/sparql	REST
ASK	Perform a SPARQL query	AS	http://moth.notube.tv:9998/triplestorage/sparql	REST
Store RDF	Perform a SPARQL query	AS	http://moth.notube.tv:9998/triplestorage/store	REST
BBC-Programmes RDF keyword query	Queries BBC programmes RDF store at TALIS (just entertainment category) via SPARQL for given set of keywords (separator: ";"); used within WP5 and iZapper demos.	OU	http://luisa.open.ac.uk:8080/axis/bbcProgrammesServices.jws?method=getvideosbykeywords	HTTP GET (SOAP enveloped message; switch to REST intended)
BBC-Backstage world news keyword query	Queries BBC backstage World news feed for given set of keywords (separator: ";"); used within WP5 and iZapper demos.	OU	http://luisa.open.ac.uk:8080/axis/bbcServices.jws?method=getvideosbykeywords	HTTP GET (SOAP enveloped message; switch to REST intended)
Youtube-OU-content-metadata-query	Queries youtube metadata of OU eLearning content for given set of keywords (separator: ";"); used within WP5 demo.	OU	http://luisa.open.ac.uk:8080/axis/ouServices.jws?method=getvideosbykeywords	HTTP GET (SOAP enveloped message; switch to REST intended)
Youtube-mobile-content-metadata-query	Queries youtube metadata of mobile suitable content for given set of keywords (separator: ";"); used within WP5 demo.	OU	http://luisa.open.ac.uk:8080/axis/mobileServices.jws?method=getvideosbykeywords	HTTP GET (SOAP enveloped message; switch to REST intended)

Openvideo-metadata-query	Queries openvideo.org video metadata for given set of keywords (separator: ";""); used within WP5 demo.	OU	http://luisa.open.ac.uk:8080/axis/openvideoServices.jws?method=getvideosbykeywords	HTTP GET (SOAP enveloped message; switch to REST intended)
Engine-EPG-metadata-query	Queries engineMedia's EPG feeds metadata for given set of keywords (separator: ";""); used within WP5 demo and to be integrated in 7b demo. Will be exposed as goal via the Broker.	OU	http://luisa.open.ac.uk:8080/axis/engineService*CHANNEL-ID*.jws?method=getvideosbykeywords (different services for each channel available via EnginMedia. For instance http://luisa.open.ac.uk:8080/axis/engineService171.jws?method=getvideosbykeywords queries only metadata for channel id 171 which is Deutsche Welle. More infos on channel IDs and the engine feeds at http://www.notube.tv/wiki/index.php/Link_for_EPG/_I_fanzy)	HTTP GET (SOAP enveloped message; switch to REST intended)

4.2. Services under development

Table 2 - Services that are currently under development by the NoTube partners.

Name	Description	Provider	URI	Interface
Reframing	Reframes a input stream according to the region of interest	TGV	-	SOAP
Ad Insertion	Insert an ad at the good place at the right time in a video stream	TGV	-	SOAP
LUPEDIA2HTML	Takes a plain text input and returns HTML formatted version of the text, enriched with data about DBPedia entries found in it.	OT	http://lupedia.ontotext.com/lookup/text2html	REST
LUPEDIA2XML	Takes a plain text input and returns XML formatted data about DBPedia entries found in it.	OT	http://lupedia.ontotext.com/lookup/text2xml	REST
LUPEDIA2JSON	Takes a plain text input and returns JSON formatted data about DBPedia entries found in it.	OT	http://lupedia.ontotext.com/lookup/text2json	REST

LUPEDIA2RDFa	Takes a plain text input and returns HTML+RDFa formatted version of the text, enriched with data about DBPedia entries found in it.	OT	http://lupedia.ontotext.com/lookup/text2rdfa	REST
Get Content Relevance	The service returns the relevance of the given NIC (a float value between 0 and 1) for the given user	AS	http://moth.notube.tv:9998/newsrecommender/related?user={userid}&content={NIC xml file as body parameter}	REST
User Interests URIs	Returns the list of user interests in terms of Linked Data URIs	AS	http://moth.notube.tv:9998/people/@me/@self?fields=interests or http://moth.notube.tv:9998/people/{guid}/@self?fields=interests	REST
NIC related URI	The service returns URI recommendation for the given NIC. Probably it will be replaced completely.	AS	-	REST
BBC-2-IMDB	Returns for a bbc-program id an RDF graph containing a matching IMDB description	OU	http://filetrace.net/notube/bbc2imdb.rdf	REST, SPARQL-over-REST
IRS-III-SWS-Broker-API-achieve-goal	Achieve goal requests to the SWS Broker: are resolved by the Broker by reasoning on available SWS descriptions and automatically identifying and orchestrating services suitable to achieve the requested goal.	OU	<ul style="list-style-type: none"> http://notube.open.ac.uk:3000/achieve-goal?... http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal-metrics?... http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?... 	REST
Term expansion using DBPedia	If there is resource in DBPedia that matches the concept that is provided in the URL, then the service returns other concepts that are related via the 'owl:sameAs' or 'redirect' relations	OU	-	HTTP GET

<p>Get-video-metadata-goal</p>	<p>Achieve goal request to the SWS Broker for video metadata retrieval. Resolved by the Broker by reasoning on available SWS descriptions and automatically identifying and orchestrating services suitable to achieve the requested goal. Brokers between video metadata retrieval services as described above. See also: http://www.notube.tv/wiki/index.php/Wp5-goals-documentation#get-video-metadata-goal:_Selection.2Finvocation_of_video_metadata_query_services</p>	<p>OU</p>	<p>http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpo-goals&goal=get-video-metadata-Goal&....</p>	<p>REST</p>
<p>Get-EPG-metadata-goal</p>	<p>Achieve goal request to the SWS Broker for EPG metadata retrieval. Resolved by the Broker by reasoning on available SWS descriptions and automatically identifying and orchestrating services suitable to achieve the requested goal. Brokers between EPG metadata retrieval services provided based on Engine metadata feeds and ZapperWarehouse services. See also: http://www.notube.tv/wiki/index.php/Wp5-goals-documentation#get-EPG-metadata-goal:_Selection.2Forchestrating.2Finvocation_of_EPG_query_services_for_a_full_documentation_of_the_goal,_its_parameters_and_the_underlying_Web_services.</p>	<p>OU</p>	<p>http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpo-goals&goal=get-EPG-BY-KEYWORD-AND-DATE-GOAL...</p>	<p>REST</p>

4.3. Services in planning

Table 3 - Services being planned by NoTube partners

Name	Description	Provider	URI	Interface
GetPublicationDate	This method returns the publication date of the video identified by its objectid. The service will parse the correspondent NIC xml file (TvAnytime file) and retrieve this information	IRT	-	SOAP
GetPublicationInfo	This method returns the publication info of the video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
GetGenre	This method returns a list of genre of the video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
GetContentRef	This method returns the mms reference to the entire TvShow video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
GetTimecodeStart	This method returns the start timecode of the single news video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
GetDuration	This method returns the duration in seconds of the single news video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP

GetLowQualityRef	This method returns the low quality mms reference of the single news video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
GetHighQualityRef	This method returns the high quality mms reference of the single news video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
GetSpeechToText	This method returns the speech transcription of the single news video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
GetEnrichmentData	This method returns all the enrichment data (URI i.e. dbpedia links) associated to the single news video identified by its objectid. The service will parse the correspondent NIC xml file and retrieve this information	IRT	-	SOAP
UploadNIC	This method receives as input the NIC.xml of a news video and save the contained information locally in order to accomplish Data Extraction tasks. The example input could change according to the chosen schema (Prestospace - TvAnytime)	IRT	-	SOAP
SetPublicationDate	This method sets the publication date of the video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP

SetPublicationInfo	This method sets the publication info of the video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP
SetGenre	This method sets the list of genres of the video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP
SetContentRef	This method sets the mms reference to the complete TvShow video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP
SetTimecodeStart	This method sets the start timecode value of the video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP
SetDuration	This method sets the duration in seconds of the video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP
SetLowQualityRef	This method sets the mms reference to the low quality video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP

SetHighQualityRef	This method sets the mms reference to the high quality video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP
SetSpeechToText	This method sets the transcription text correspondent to the video identified by its objectid. The service will parse the correspondent NIC xml file and adds or updates this information	IRT	-	SOAP
AddEnrichmentData	This method receives as input a new enrichment data for the video identified by its objectid. The service will parse the correspondent NIC xml file and adds this external link into the dedicated tag	IRT	-	SOAP
DeleteAllEnrichments	This method deletes all the enrichment data in the NIC relative to the video identified by its objectid. The service will parse the correspondent NIC xml file and deletes all the external link from the dedicated tag	IRT	-	SOAP
ConvertPrestospaceToTvAnytime	This method is used to convert a Prestospace file into a TvAnytime. All the data contained in the first schema must be maintained into the new file, no lose of data is allowed	IRT	-	SOAP

5. NoTube use case scenarios for 1st prototypes

5.1. WP7a scenario – personalised semantic news (PSN)

The Personalized Semantic News use case (WP7a) envisages a system that can acquire news items from generic broadcast streams and obtain additional enriched news information by using a set of personalised news related services. Moreover, the system will enable the understanding of the meaning of video news items, the understanding of the physical context in which news items are going to be shown, and will apply criteria for matching the user profile with the available news items in order to filter news items according to given user preferences.

Realising this system will require cooperation with work on metadata conversion services (WP2), enrichment services (WP4), transcription services (WP7.a), Beanconter services (WP3 and WP7.c) and recommendation services (WP3). It is proposed that this and other required functionalities should be exposed through the Broker. Furthermore, the Broker can select the most appropriate service for a given task, and can orchestrate a set of services to achieve more complex tasks.

Current state of integration of the Broker into 1st prototypes

One envisaged proposal for integrating the Broker with the PSN system is that given a user request for content and the current user device, the Broker selects either the content delivery service that provides the appropriate quality content for the device or the appropriate format-conversion service that can transform the generic content feed format. This assumes the existence of a service that can take a News Item Container and returns links to the different content delivery services providing the news content.

Figure 4 shows a conceptual layout for where the Broker is positioned in the proposed WP5-WP7a integration scenario. The figure shows conceptually how the Broker exposes underlying service functionality through its “achieve goal” interface. When these goals are invoked, they can trigger the Broker to orchestrate a set of services – for example, in order to achieve the Service Enrichment Goal, we can envisage that there will be a need to orchestrate the process of first converting the NIC metadata into a format that can be read by the WP4 enrichment service, and then executing the enrichment service on the metadata. The Broker can also be triggered to select from among different services – for example, in order to achieve the Item Selection Goal, we can envisage the need to select between a number of dedicated services that retrieve NIC content for mobile device, or TV device, etc.)

WP5-WP7a integration proposal --- Conceptual Diagram

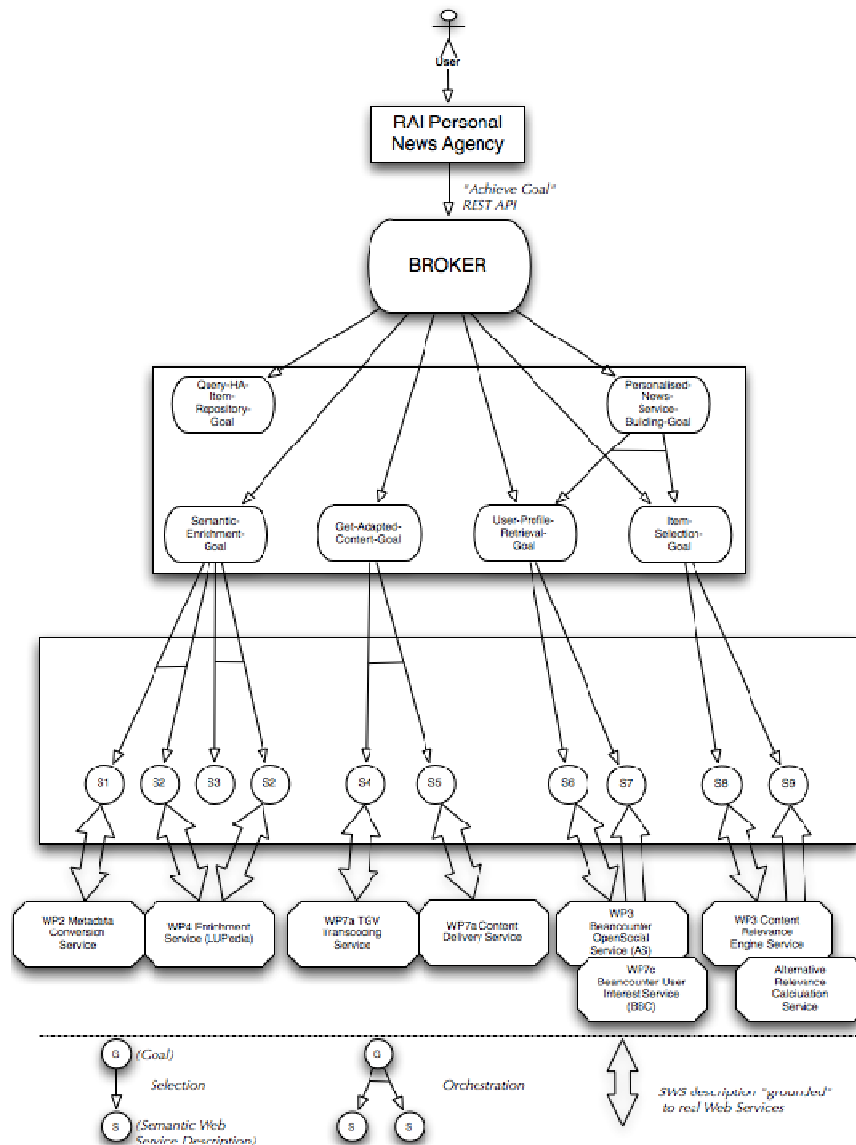


Figure 4 - Conceptual diagram of WP5-WP7a integration

5.2. WP7b scenario – personalised TV guide with adaptive advertising

The Personalised TV Guide use case (WP7b) envisages a suite of iFanzly clients (e.g. Web-based and iPhone-based) that can be used to display TV program recommendations with additional related information gained from Internet resources provided by enrichment services. The recommendations are also based on user’s context information, such as user activities, languages and personal interests in order to provide the suitable recommendations and advertisements in adequate formats. Thus the core services that underpin this scenario are user profiling and recommendation (WP3), and content enrichment (WP4). These services will be semantically described in the Broker, which will then expose the underlying functionality as “goals”, which in turn allows the most appropriate service to be selected to achieve a particular task.

Current state of integration of the Broker into 1st prototypes

Figure 5 shows a conceptual layout for where the Broker is positioned in the proposed WP5-WP7a integration scenario. As in the integration scenario with WP7a, the Broker exposes its functionality through the “achieve goal” interface. In the WP7b integration scenario, when these goals are invoked, they can trigger the Broker to select from among different services. For example, for Get EPG Metadata Goal we can envisage the need to select – based on certain parameters such as language – between a number of dedicated EPG service feeds. Invoking this goal executes functionality that can search the different Engin EPG feeds for programme information that falls within a given time period. The iFanzly client can invoke this functionality via the Broker and display it in the interface. Specifically, the iFanzly client requires a goal, invocable through the Broker, that can "get-epg-by-period" and return an EPG response. Currently, the Broker exposes EPG feeds (through a layer of semantic descriptions and goal definitions), and these feeds can be selected based on user language and filtering keywords.

WP5-WP7b integration proposal --- Conceptual Diagram

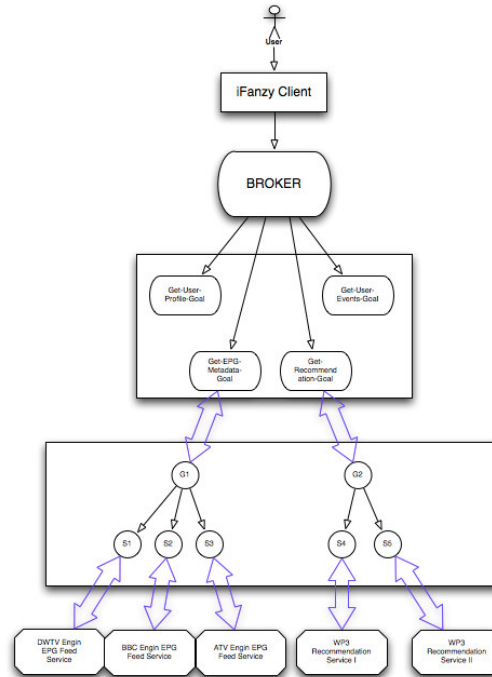


Figure 5 - Conceptual diagram of WP5-WP7b integration.

5.3. WP7c scenario – Internet TV in the social Web

This use case envisages a system and a set of services that allow a user to watch broadcast and on-demand TV, view their personalised recommendations, saying they would like to know more about a programme, using a simple remote. For instance, a user wants to see recommendations based on her/his social activities (such as social communications on twitter and facebook) and watch the recommended programs when she gets home at night. The user wants to see recommendations clearly featured on the user interface of the set top box. Meanwhile the user also would like to obtain more information about a programme. It is proposed that the user is able to view these recommendations in an EPG format.

Current state of integration of the Broker into 1st prototypes

One proposal for integrating the Broker in the WP7c scenario involves orchestrating a service that retrieves a given user's list of interests and services, exposed via the Broker, that return EPG feeds. This proposal is in line with the one above for integrating the Broker in the WP7b scenario, and validates our approach of attempting to derive brokering requirements common across all three main use cases.

A second proposal for integrating the Broker with the WP7c scenario involves using the Broker to select between different recommendation services, each of which may be based on different recommendation algorithms. The most appropriate algorithm could be based on user feedback: we can imagine a feedback mechanism where the user is shown a series of recommendations generated by different recommendation algorithms and the user can vote which recommendations they 'Like' or 'Dislike'. This feedback information can then be fed to a ranking mechanism, handled by the Broker, than can then be used to determine which recommendation algorithm is chosen for presenting future recommendations to the user.

5.4. Derived requirements

Following the task of describing each use case scenario for each of WP7a, WP7b, and WP7c, and describing how the Broker is expected to be positioned in each scenario, the task is now to derive a list of functional requirements based on analysing the use case scenarios. Table 4 shows this list of requirements, specifically the requirements for the Broker to discovery, select, and/or orchestrate a set of services to achieve a higher-level user goal. The derived Broker requirements listed in the table do not consider the specific implementation requirements of each use case; rather the aim is to provide a list of common requirements that the Broker must meet to enable the application layer to implement the application scenarios that are part of the work performed in WP7.

Table 4 - Scenario-driven requirements for the Semantic TV Broker

	Scenario requirements	Brokering requirements
Personalised Semantic News (WP7a)	To acquire news items from generic broadcast streams and from internal TV archives using scheduling information and semantic filters	To automatically discover repositories and access services/content. To automatically discover metadata enrichment services
	To access external resources from predefined areas of the Web	To automatically discover external Web services/content To orchestrate appropriate services in order to enable the application layer to query for context filtered items To mediate between distinct service I/O messages To abstract from service implementations and communication standards

	To understand the meaning of video news items, based on criteria such as subject, location, time, etc.	To automatically discover repositories and access services/content. To automatically discover metadata conversion services To automatically discover context services To orchestrate appropriate services in order to enable the application layer to query for context filtered items
	To enrich news items with metadata and related content	To automatically discover repositories and access services/content. To automatically discover metadata enrichment services
	To produce services based on available content and information filtered based on semantic filters (e.g. User rules, Device and Environment rules, or Service Provider rules)	To automatically discover repositories and access services/content. To automatically discover metadata conversion services To automatically discover context services To orchestrate appropriate services in order to enable the application layer to query for context filtered items
Personalised TV Guide with Adaptive Advertising	To provide additional WWW-based information to enrich TV guide content	To automatically discover external Web services/content To mediate between distinct service I/O messages
	To provide multiple user identification	To automatically discover user-profiling services
	To integrate user context information	To automatically discover context services
	To integrate social aspects	To automatically discover external Web services/content
	To provide personalised advertisement delivery in iFanzzy	To automatically discover repositories and access services/content. To automatically discover metadata conversion services To automatically discover user-profiling services To orchestrate appropriate services in order to enable the application layer to query for context filtered items

	To provide personalised product placement in video streams	To automatically discover repositories and access services/content. To automatically discover metadata conversion services To automatically discover context services To automatically discover recommendation services
Internet TV in Social Web	To acquire programme information, schedules, and metadata from BBC archives and RESTful BBC services	To automatically discover repositories and access services/content. To automatically discover metadata enrichment services To mediate between distinct service I/O messages To abstract from service implementations and communication standards
	To be able to uniquely identify programmes, brands, broadcast events, people (users) with dereferenceable identifiers	To automatically discover repositories and access services/content. To automatically discover context and user-profiling services To discover services for classifying TV-related contents (programmes, broadcast events, brands) and user categories
	To use a vocabulary (ontology) for programmes that allows navigation between related programmes and other related items (for example concerts)	To automatically discover repositories and access services/content. To automatically discover external Web services/content To mediate between distinct service I/O messages To automatically discover recommendation services
	To be able to access classification resources, including information derived from the Web ("External Sources") and other specific classification systems	To automatically discover external Web services/content To mediate between distinct service I/O messages To abstract from service implementations and communication standards
	To be able to access and use RESTful access and update services for user profiles	To automatically discover context and user-profiling services To abstract from service implementations and communication standards

In addition to the requirements listed in the table, there is an underlying but rather important requirement of the Broker that will influence the development within WP5 for the remainder of the project. That requirement is for the Broker to play the role of a repository of service annotations, where both functional and non-functional properties of available services are semantically described so the services can be discovered, selected, and orchestrated to

meet specific use-case requirements. These project-driven, cross-scenario requirements as well as the individual scenario-driven requirements in the above table are summarised and enumerated in Table 5.

Table 5 - Summarised, enumerated list of requirements for the Semantic TV Broker.

Requirement Number	Requirement description
R1	To automatically discover repositories and access services/content.
R2	To automatically discover metadata enrichment services
R3	To automatically discover external Web services/content
R4	To orchestrate appropriate services in order to enable the application layer to query for context filtered items
R5	To mediate between distinct service I/O messages
R6	To abstract from service implementations and communication standards
R7	To automatically discover metadata conversion services
R8	To automatically discover context services
R9	To automatically discover user-profiling services
R10	To automatically discover recommendation services
R11	To discover services for classifying TV-related contents (programmes, broadcast events, brands) and user categories
R12	To store structured documentation of services used within NoTube
R13	To provide project-wide access to service documentation in order to facilitate development
R14	To provide search facilities to allow NoTube developers to find and reuse available services

The next section describes the work done in WP5 on developing a first prototype of the Semantic TV Broker. The main rationale for developing a Broker prototype this early in the process is to better understand how the requirements previously listed can be met using an approach based on Semantic Web Services. The prototype implements a subset of the required functionality based on the subset of services that are currently available within the project, namely video metadata retrieval services and electronic programme guide (EPG) services.

6. NoTube Semantic TV Broker prototype

Based on the requirements described in the previous section, WP5 has developed an initial proof-of-concept prototype, which forms the basis for future implementation work. Hence, this prototype incorporates some of the services, which had been developed to be integrated into the WP7 use case scenarios in order to showcase the WP5 Broker capabilities and to develop future scenarios of usage. This section describes the architecture of the prototype, the API for clients to interact with the Broker, and the semantic annotation of some of the key services currently available, namely video metadata retrieval services and EPG data services.

6.1. Overall architecture

Figure 6 depicts the overall architecture of the current WP5 prototype:

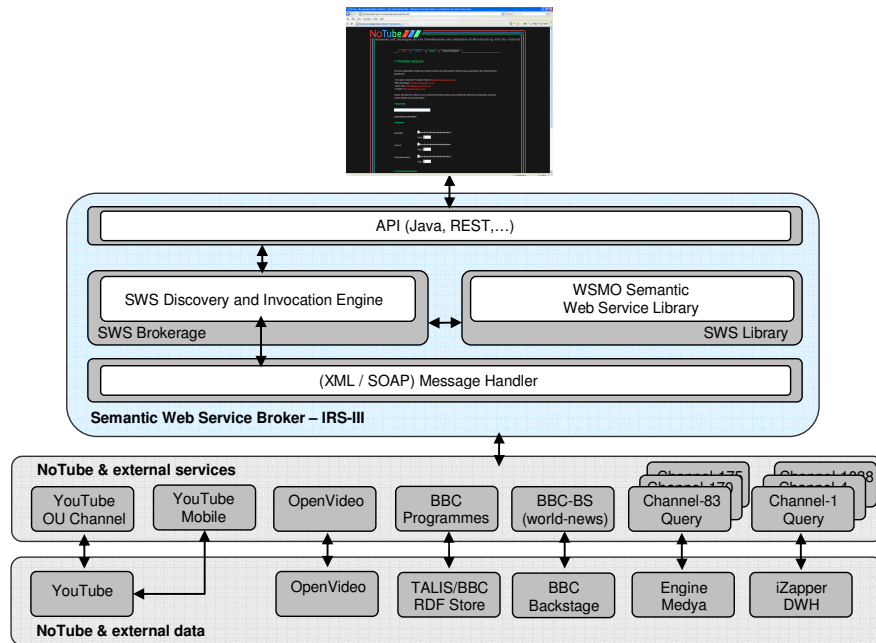


Figure 6 - WP5 proof-of-concept prototype architecture.

As depicted above, a large set of Web services is currently being used and annotated based on a range of distributed data repositories. These services can be classified as (i) Video metadata retrieval services and (ii) Electronic Program Guide (EPG) metadata retrieval services.

Video metadata services

These services make use of the Youtube-API⁸, the BBC (WP7c) TALIS SPARQL endpoint⁹ as well as data feeds provided by BBC- Backstage¹⁰ and Open Video¹¹ and provide capabilities to query and filter several distributed video repositories:

⁸ <http://code.google.com/intl/en/apis/youtube/>

⁹ <http://api.talis.com/stores/bbc-backstage/services/sparql>

¹⁰ <http://backstage.bbc.co.uk/>

¹¹ <http://www.open-video.org/>

- Open-Video¹¹
- BBC Programmes RDF Store¹²
- The Open University's Youtube channel¹³
- BBC Backstage World News feed¹⁴
- Youtube's mobile suitable content

EPG data services

In addition, a set of EPG retrieving and filtering services was provided, which was basically based on two sets of services:

- (a) EPG metadata provided by Engin Medya as part of WP7b
- (b) "ZapperWarehouse" services provided for prototyping purposes by VU

The services (a) allow queries for EPG metadata based on XML feeds¹⁵ which provide the opportunity to query for EPG metadata for 9 different channels, including several Turkish, German and English ones. In addition, (b) consists of REST based services¹⁶ allowing to query an additional set of channels in a wide variety of languages (including amongst others Italian, Dutch, English, French, German, Belgian ones).

Since the services themselves just provide unfiltered EPG metadata, we have added a set of REST-based services on top of these which allow performing keyword-based queries on the retrieved metadata.

Video/EPG metadata retrieval via goal-based service discovery

Based on the semantic annotations of these services, we provided a set of SWS Goals which allow to query and filter (a) video metadata and (b) EPG metadata. A detailed description of these goals can be found in Section 6.3. Hence, instead of developing a user interface which directly queries a multiplicity of heterogeneous services, the IRS-III serves as single entry point for the application layer and abstracts from underlying service heterogeneities, e.g. heterogeneous service interfaces, message formats and capabilities. Based on a given set of contextual parameters, such as the user language, the user environment (e.g. device resolution, connection bandwidth), a set of matching services is being orchestrated and executed. Different metadata schemas are being aligned and a shared XML-schema for all service responses is being applied on the fly.

As a major benefit of this approach, we ensure that only services are being executed which actually are relevant to the user. For instance, if a user is looking for educational video material, only services which provide educational content/metadata will be executed while, to introduce another example, an English-speaking user looking for EPG metadata will only be provided with metadata from English channels. Hence, performance can be improved while ensuring the highest suitability of the retrieved results for the user.

An AJAX-based user interface was developed as part of the WP5 proof-of-concept prototype which allows to (a) gather a set of contextual parameters from the user (e.g., user language, technical environment, kind of content to be queried, etc.) and (b) to send a matching achieve-goal request to the Broker and to present the consolidated responses retrieved from the Broker.

¹² <http://api.talis.com/stores/bbc-backstage>

¹³ <http://www.youtube.com/user/TheOpenUniversity>

¹⁴ http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/world/rss.xml

¹⁵ E.g., <http://tumdata.triplinq.com/phpscripts/xmlfeed4allifanz.php?d=17&m=9&y=2009&id=83>

¹⁶ E.g.,

<http://services.notube.tv/notube/zapper/datawarehouse.php?service=getprogrammesperiod&channelid=1&start=2009-11-21%2010:00&stop=2009-11-21%2023:55>

Figure 7 depicts the user interface representing a set of metadata records which match the parameters of a given query and which were retrieved by the IRS-III via service discovery, orchestration and execution.

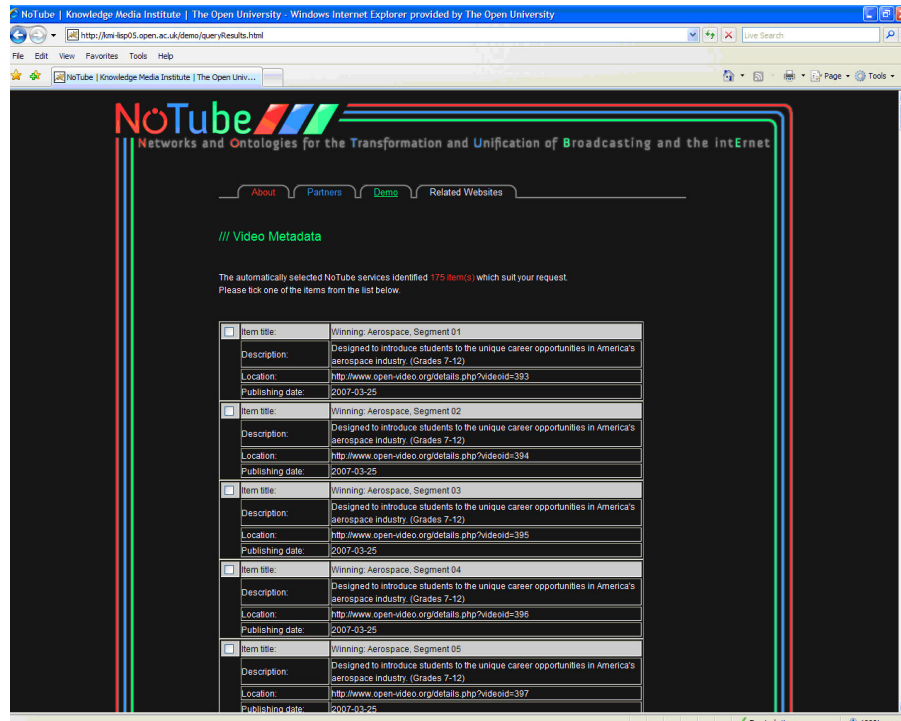


Figure 7 - WP5 proof-of-concept prototype AJAX interface presenting a set of matching video metadata records.

In order to better visualize the service matchmaking procedure, which is processed following an achieve-goal request within the Broker, an additional API function had been added to the Broker, which allows representing the intermediate results of the service matchmaking as some sort of service suitability ranking, as displayed in Figure 8.

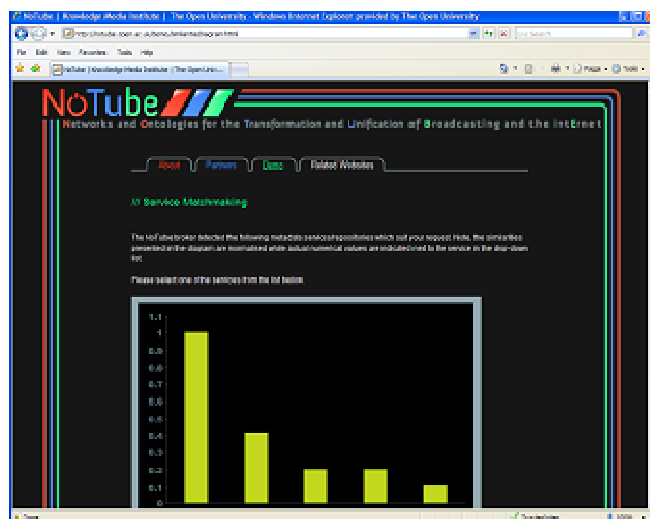


Figure 8 - AJAX interface for representing the intermediate results of the service matchmaking.

6.2. API

In this section we explain the Broker API functionality that we expose through a number of HTTP requests. The HTTP based API allows the client to request the core functionalities of the Broker. Here, we currently focus on the IRS 'achieve-goal' API as this is the main client interaction that we envisage with the Broker.

Achieve goal

Primary among the functionalities of Broker is the 'achieve-goal' HTTP request. In our approach, services are semantically annotated and associated to so-called 'goal' representations. While achieving a goal, the Broker automatically matches the requested goal and all its contextual parameters to the available SWS, selects the most suitable ones and executes these. This process of selection and invocation can also involve the mediation of mismatches between the I/O messages of requester and services.

Note, different 'achieve-goal' functions are available, each realising different matchmaking algorithms:

- <http://notube.open.ac.uk:3000/achieve-goal?...>
- <http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal-metrics?...>
- <http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?...>

In addition, we intend to provide additional specialised 'achieve-goal' functions if required for the needs of the NoTube scenarios.

Simple “achieve-goal” example

Following is a simple example that demonstrates the 'achieve-goal' request. In this example, the 'achieve-goal' directly leads to execution of an underlying service¹⁷, which retrieves metadata from the BBC Programmes RDF repository for a given set of keywords.

<http://notube.open.ac.uk:3000/achieve-goal?ontology=mmpo-goals&goal=get-bbc-entertainment-metadata-goal&has-method=%22getvideosbykeywords%22&has-keywords=%22space::andy%22>

The general parameters of any 'achieve-goal' request are:

- ontology=NAME-OF-ONTOLOGY-WHERE-GOAL-IS-DEFINED (In the example above the name of the ontology is 'mmpo-goals')
- goal=NAME-OF-THE-GOAL (In the example above the name of the goal is 'get-bbc-entertainment-metadata-goal')
- a set of input parameters that are mapped to the inputs of the underlying Web service (In the example above the input methods are 'has-method' and 'has-keywords')

Additional parameters are used by other 'achieve-goal' functions to describe the contextual parameters of the request (e.g. the user language or device). These are submitted as part of a so-called assumption parameter (see below).

In the case of the above example, the Broker invokes the goal, and returns the result as the HTTP response.

¹⁷ The underlying service can be directly accessed at the following URL: [http://luisa.open.ac.uk:8080/axis/bbcProgrammesServices.jws?method=getvideosbykeywords&keywords="space::andy"](http://luisa.open.ac.uk:8080/axis/bbcProgrammesServices.jws?method=getvideosbykeywords&keywords=)

6.3. Integrated SWS goals

In this section, we introduce a preliminary set of goals, which are being used within the current prototype and illustrate their functionality.

Get-video-metadata-goal: selection and invocation of video metadata query services

One key benefit of invoking goals rather than directly executing services is that the Broker can select from among a set of services the most appropriate service for a particular context (e.g. a particular user language or environment). For example, Figure 9 shows an 'achieve-goal' request where the named goal selects between 5 metadata search services, each querying and filtering metadata from a different repository (i.e. BBC programmes RDF store @ TALIS, Open Video, BBC Backstage news feeds, Open University's Youtube channel, Youtube mobile suitable content).

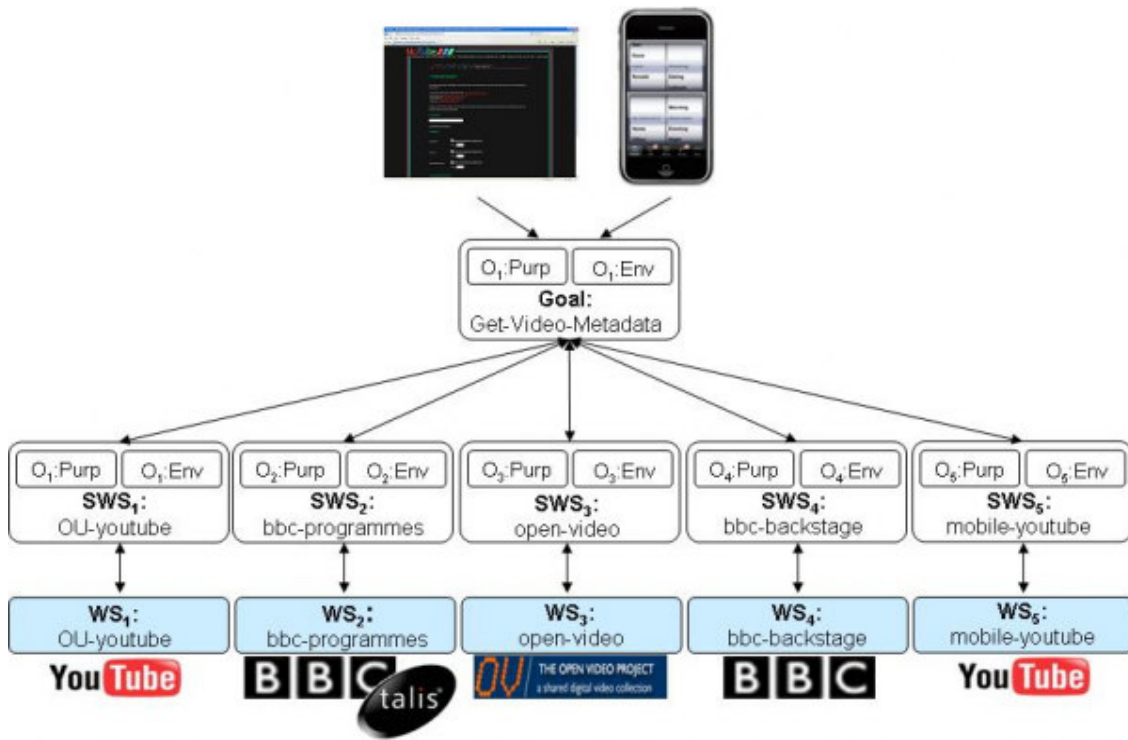


Figure 9 - Conceptual overview of Get-Video-Metadata-Goal and corresponding SWS.

Given a set of assumption parameters defining, for instance, the user language, the nature of the requested content (i.e., "purpose" such as educational or entertainment) or the technical "environment" (e.g. resolution and bandwidth), the Broker automatically selects and executes the most suitable service. At execution time, the services query the repositories on the fly and filter the metadata according to the keywords provided with the "has-keywords" parameter.

- [http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpp-goals&goal=get-video-metadata-Goal&has-method=getvideosbykeywords&has-keywords=physics&assumption=\(%22education-entertainment-purpose%22\)](http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpp-goals&goal=get-video-metadata-Goal&has-method=getvideosbykeywords&has-keywords=physics&assumption=(%22education-entertainment-purpose%22))

The request above achieves a goal, which is associated with 5 metadata query services (the BBC one being one of them) and selects/invokes the most suitable one depending on the assumption parameters. The latter are in this case strings, which refer to instance-IDs in the user-language ontology within the Broker.

- <http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal-metrics?ontology=mmpo-goals&goal=get-video-metadata-Goal&has-method=getvideosbykeywords&has-keywords=the&assumption=purpose-type:80;0;0::environment-type:80;100>

The request above achieves a goal, which is associated with the same 5 metadata query services and selects the most suitable one depending on the numbers which are part of the assumption parameter. Here, the assumption parameter is represented through measurements (for instance defining a particular location), which are matched to predefined instances (and their measurements) as part of SWS descriptions within the Broker.

Get-EPG-metadata-goal: selection, orchestration, and invocation of EPG query services

Similar to the video-metadata-retrieval goal, another goal was being provided which allows to query EPG metadata from a wide variety of EPG feeds. This goal was being developed based on discussions with use case leaders of WP7a/b/c as well as VU as leader of the overall scenario development effort. It is intended to integrate the goal into some of the respective prototypes. The current goal provides a single entry point to query EPG metadata across different metadata feeds. Therefore we have annotated services provided by VU (<http://www.notube.tv/wiki/index.php/Services/iZapperdatawarehouse>) and have provided additional services on top of the metadata feeds provided by Engyn Media (http://www.notube.tv/wiki/index.php/Link_for_EPG_/I_fanzy). Each service allows to query EPG of a specific channel, while query parameters are the time period (start and end date) and a set of keywords which is used to filter the metadata. Note the keywords are optional and could be left blank. While querying each service individually is time consuming and requires some mechanism of selecting the most appropriate service for a given context or user, our goal-based approach provides a single entry point to submit a query while the Broker automatically selects the most suitable EPG services for a given context and orchestrates and executes these. This EPG invocation scenario is depicted conceptually in Figure 10.

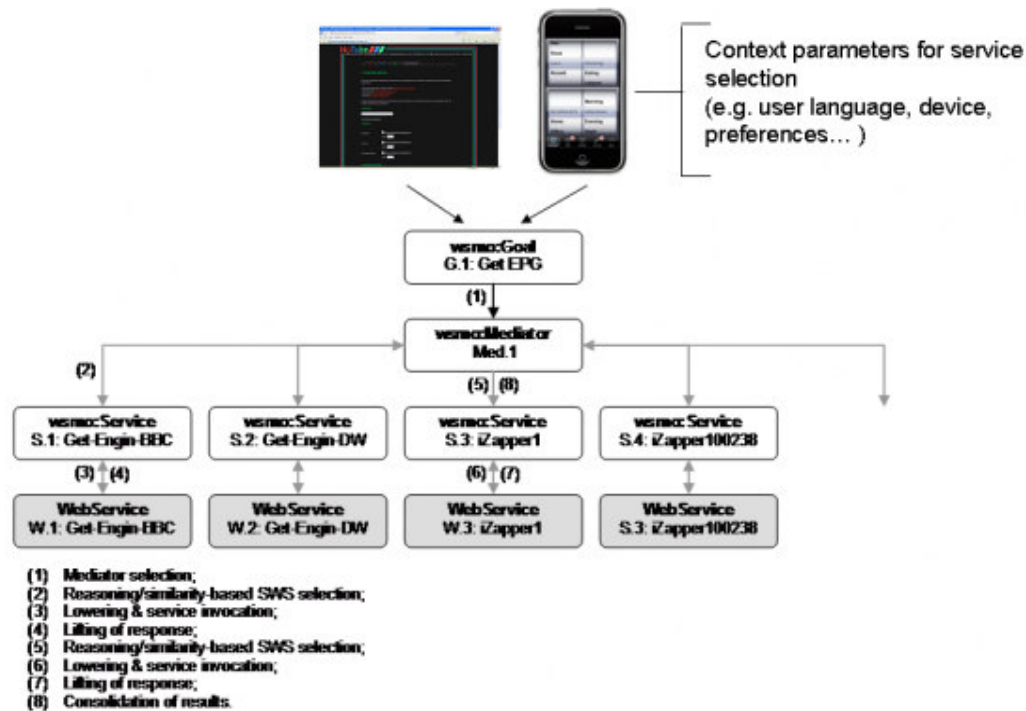


Figure 10 - Conceptual overview of Get-EPG-Metadata-Goal and corresponding SWS.

The following example URI sends a request via the Broker REST API to achieve the get-epg-metadata goal:

- [http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpo-goals&goal=get-EPG-BY-KEYWORD-AND-DATE-GOAL&has-method=getvideosbykeywords&has-keywords=%22week%22&has-start-date=%2219-11-2009%2000:00%22&has-end-date=%2225-11-2009%2023:59%22&assumption=\(%22english-language%22\)](http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpo-goals&goal=get-EPG-BY-KEYWORD-AND-DATE-GOAL&has-method=getvideosbykeywords&has-keywords=%22week%22&has-start-date=%2219-11-2009%2000:00%22&has-end-date=%2225-11-2009%2023:59%22&assumption=(%22english-language%22))

For instance, the request above sends a request to the Broker for EPG metadata within the specified period ("has-start-date", "has-end-date") and which match the keyword "week" (specified via the "has-keywords" parameter). While these parameters are used as inputs for the actually selected services, the "assumption" parameter refers to instances within the Broker which are used for the service matchmaking. Within the query above, the user asks for "english-language" leading to the Broker selecting only services which query English language channels for execution. Other assumption parameters would be, for instance, "german-language", "dutch-language", "turkish-language" which would all lead to the orchestration of different service queries. In the current implementation, all services were annotated with the channel they are providing metadata for and an ontology provides information about the language supported by each channel.

The result is a consolidated set of XML metadata of EPG programmes which match the criteria described above. Since different services respond distinct metadata following different XML schemas, the Broker automatically applies a common schema and returns the respective results.

For instance the following would be part of the response to the query above:

```
<metadata>
  <item>
    <title>The National Lottery: Midweek Draws</title>
    <description></description>
    <url>b00pldxt</url>
    <pubDate>2009-11-18 22:35:00</pubDate>
    <channel>BBC 1</channel>
  </item>
  <item>
    <title>This Week</title>
    <description>A weekly showcase for reports from the BBC's network of
over 250 global correspondents.</description>
    <url></url>
    <pubDate>2009-11-22 02:30:00</pubDate>
    <endDate>2009-11-22 03:00:00</endDate>
    <channel>BBC World</channel>
  </item>
</metadata>
```

Also, please note that in principal any kind of selection criteria could be applied, for instance "genre" for selecting only genre-specific channels. Also, the XML schema applied to all responses currently could be modified according to individual requests.

Another example request of the same goal (but here querying only German language channels/services would be:

- [http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpo-goals&goal=get-EPG-BY-KEYWORD-AND-DATE-GOAL&has-method=getvideosbykeywords&has-keywords=%22wetter%22&has-start-date=%2219-11-2009%2000:00%22&has-end-date=%2225-11-2009%2023:59%22&assumption=\(%22german-language%22\)](http://notube.open.ac.uk:8080/cs-invocation/achieve-cs-goal?ontology=mmpo-goals&goal=get-EPG-BY-KEYWORD-AND-DATE-GOAL&has-method=getvideosbykeywords&has-keywords=%22wetter%22&has-start-date=%2219-11-2009%2000:00%22&has-end-date=%2225-11-2009%2023:59%22&assumption=(%22german-language%22))

Please note, that the current goal just exposes a subset of all available EPG channels from <http://www.notube.tv/wiki/index.php/Services/iZapperdatawarehouse> but the complete list of channels from Engyn Media (http://www.notube.tv/wiki/index.php/Lnk_for_EPG_/I_fanzy). Further services/channels will be added incrementally.

As a major advantage of our approach, the goal above represents a single URI which can be exposed to application layers of specific prototypes (e.g. 7a/b/c, iZapper) and which abstracts from implementation heterogeneities of underlying services. Also, services could be changed/removed/added without affecting the application layers while the interface modifications would be handled within the Broker.

7. Conclusion

This deliverable has proposed a set of requirements for the SWS Broker envisaged within NoTube and discussed some state of the art technologies which will be considered during the design and implementation stage of the the SWS Broker. While the SWS Broker is meant to abstract from underlying service implementations by providing a single entry point to the NoTube application logic, it will enable the automatic orchestration of distributed services by means of semantics.

Similar to deliverable D6.1 on NoTube specifications and overall architectural design, this deliverable consists of two releases: a preliminary which was submitted in M3 and this current document which refines the earlier release. The goal of the first release was to narrow the requirements for the SWS Broker and to set the basis for the development work scheduled for the first year of the project. The goal of this current release has been to report on the refinement of requirements in light of more thorough discussions with use case work package (WP7). The deliverable described how the functional requirements for the Semantic TV Broker component are based on the use cases obtained by the WP7 tasks, the architectural requirements posed by WP6 task 6.1, and taking into account the work to be done in the technical WPs (i.e. WP1, WP2, WP3, and WP4).

While the initial version of this deliverable – D5.1a released in M3 – posed a very preliminary set of requirements and an abstract classification of services, D5.1b now acts as facilitator for the remaining implementation work within WP5.

8. References

- [1] 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.
- [2] Dietze, S., Gugliotta, A., Domingue, J., (2007) A Semantic Web Service oriented Framework for adaptive Learning Environments. European Semantic Web Conference (ESWC) 2007, Innsbruck, Austria, June 2007.
- [3] Dietze, S., Gugliotta, A., Domingue, J., (2008) Conceptual Situation Spaces for Situation-Driven Processes. 5th European Semantic Web Conference (ESWC), Tenerife, Spain, June 2008.
- [4] OWL-S Coalition: OWL-S 1.1 release. (2004). <http://www.daml.org/services/owl-s/1.1/>
- [5] WSMO Working Group (2004), D2v1.0: Web service Modeling Ontology (WSMO). WSMO Working Draft, (2004). (<http://www.wsmo.org/2004/d2/v1.0/>).
- [6] Hepp, M., Leymann, F., Domingue, J., Wahler, A. and Fensel, D.: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. Proceedings of IEEE Intl. Conf. on e-Business Engineering (ICEBE 2005) pp. 535-540. Beijing, China (2005).
- [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-wsdl-20010315>)