DELIVERABLE SUBMISSION SHEET

To: Manuela Speiser (Project Officer)
EUROPEAN COMMISSION
Directorate-General Information Society and Media
EUFO 1165A
L-2920 Luxembourg

From:

Project acronym: ARCOMEM  Project number: 270239
Project manager: Wim Peters
Project coordinator: The University of Sheffield (USFD)

The following deliverable:

Deliverable title: ARCOMEM Parliament Tool V1 Report
Deliverable number: 9.1
Deliverable date: 30 September 2012
Partners responsible: ATC
Status: ☒ Public  ☐ Restricted  ☐ Confidential

is now complete. ☒ It is available for your inspection.
☒ Relevant descriptive documents are attached.

The deliverable is:

☒ a document
☐ a Website (URL: .........................)
☐ software (.........................)
☐ an event
☐ other (.........................)

Sent to Project Officer: manuela.speiser@ec.europa.eu
Sent to functional mail box: INFSO-ICT-270239@ec.europa.eu
On date: 1 October 2012
D9.1 ARCOMEM Parliament Tool V1 Report

Deliverable Co-ordinator: Dimitris Spiliotopoulos

Deliverable Co-ordinating Institution: ATC

Other Authors: Dimitris Koryzis (HEP); Penelope Levantaki (HEP); Katerina Doka (Athena), Efstratios Tzoannos (ATC), Günther Schefbeck (AUP)

Document Identifier: ARCOMEM/2012/D9.1
Date due: August 2012

Class Deliverable: ARCOMEM EU-ICT-2009-270239
Submission date: September 2012

Project start date: January 1, 2011
Version: V1

Project duration: 3 years
State: Final

Distribution: Restricted
## ARCOMEM Consortium

This document is a part of the ARCOMEM research project funded by the ICT Programme of the Commission of the European Communities by the grant number ICT-2009-270239. The following partners are involved in the project:

<table>
<thead>
<tr>
<th>The University of Sheffield (USFD) – Coordinator</th>
<th>Leibniz Universität Hannover (LUH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department of Computer Science</td>
<td>Forschungszentrum L3S</td>
</tr>
<tr>
<td>Regent Court</td>
<td>Appelstrasse 9a</td>
</tr>
<tr>
<td>211 Portobello</td>
<td>30169 Hannover</td>
</tr>
<tr>
<td>Sheffield, S1 4DP</td>
<td>Germany</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>Contact person: Thomas Risse</td>
</tr>
<tr>
<td>Contact person: Hamish Cunningham, Wim Peters</td>
<td>E-mail address: <a href="mailto:risse@L3S.de">risse@L3S.de</a></td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:arcomem-coord@lists.dcs.shef.ac.uk">arcomem-coord@lists.dcs.shef.ac.uk</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Yahoo Iberia SLU (YIS)</th>
<th>Internet Memory Foundation (EA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avinguda Diagonal 177, 8th floor, Barcelona,</td>
<td>45 ter rue de la revolution</td>
</tr>
<tr>
<td>08018, CAT, Spain</td>
<td>93100 Montreuil</td>
</tr>
<tr>
<td>Contact person: Alejandro Jaimes</td>
<td>France</td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:ajaires@yahoo-inc.com">ajaires@yahoo-inc.com</a></td>
<td>Contact person: Julien Masanes</td>
</tr>
<tr>
<td></td>
<td>E-mail address: <a href="mailto:julien@internetmemory.org">julien@internetmemory.org</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>University of Southampton (SOTON)</th>
<th>Athens Technology Center (ATC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room 4011, Building 32</td>
<td>10, Rizariou Street</td>
</tr>
<tr>
<td>Highfield campus</td>
<td>15233, Halandri</td>
</tr>
<tr>
<td>University of Southampton</td>
<td>Athens, Greece</td>
</tr>
<tr>
<td>SO17 1BJ</td>
<td>Contact person: Dimitris Spiliotopoulos</td>
</tr>
<tr>
<td>Contact person: Paul Lewis</td>
<td>E-mail address: <a href="mailto:d.spiliotopoulos@atc.gr">d.spiliotopoulos@atc.gr</a></td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:phil@ecs.soton.ac.uk">phil@ecs.soton.ac.uk</a></td>
<td></td>
</tr>
</tbody>
</table>

| ATHENA Research and Innovation Center in         | Telecom ParisTech (IT)           |
| Information Communication & Knowledge Technologies (ATHENA) | 46 rue Barrault                   |
| National Technical University of Athens          | 75634 Paris Cedex 13              |
| School of Electrical and Computer Engineering    | France                           |
| Division of Computer Science                     | Contact person: Pierre Senellart |
| Iroon Polytechniou 9                             | E-mail address: pierre.senellart@telecom-paristech.fr|
| Athens, 15780                                    |                                   |
| Greece                                          |                                   |
| Contact person: Nectarios Koziris                |                                   |
| E-mail address: nkoziris@imis.athena-innovation.gr|                                   |

<table>
<thead>
<tr>
<th>Deutsch Welle (DW)</th>
<th>SUDWESTRUNDFUNK (SWR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neue Medien / Distribution</td>
<td>Hans-Bredow-Strasse,</td>
</tr>
<tr>
<td>Voltastr. 6</td>
<td>D-76522 Baden-Baden</td>
</tr>
<tr>
<td>13355 Berlin, Germany</td>
<td>Germany</td>
</tr>
<tr>
<td>Contact person: Birgit Gray</td>
<td>Contact person: Robert Fischer</td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:birgit.gray@dw-world.de">birgit.gray@dw-world.de</a></td>
<td>E-mail address: <a href="mailto:robert.fischer@swr.de">robert.fischer@swr.de</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HELLENIC PARLIAMENT (HEP)</th>
<th>PARLAMENTSDIREKTION (AUP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amalias 14, 10557, Athens, Greece</td>
<td>Dr. Karl Renner-Ring 3</td>
</tr>
<tr>
<td>Contact person: Dimitris Koryzis</td>
<td>A-1017 Vienna</td>
</tr>
<tr>
<td>E-mail address: <a href="mailto:dkoryzis@parliament.gr">dkoryzis@parliament.gr</a></td>
<td>Contact person: Guenther Schefbeck</td>
</tr>
<tr>
<td></td>
<td>E-mail address: <a href="mailto:guenther.schefbeck@parlament.gv.at">guenther.schefbeck@parlament.gv.at</a></td>
</tr>
</tbody>
</table>
Milestone participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

Change Log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Amended by</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>9-7-2012</td>
<td>Dimitris Spiliotopoulos</td>
<td>Initial structure</td>
</tr>
<tr>
<td>0.2</td>
<td>25-7-2012</td>
<td>Dimitris Spiliotopoulos</td>
<td>Updated</td>
</tr>
<tr>
<td>0.3</td>
<td>6-8-2012</td>
<td>Dimitris Spiliotopoulos</td>
<td>Updated</td>
</tr>
<tr>
<td>0.4</td>
<td>29-8-2012</td>
<td>Dimitris Spiliotopoulos</td>
<td>Major update and formatting</td>
</tr>
<tr>
<td>0.5</td>
<td>30-8-2012</td>
<td>Stratos Tzoannos</td>
<td>updated</td>
</tr>
<tr>
<td>1.0</td>
<td>01-10-2012</td>
<td>ATC</td>
<td>Final based on QA feedback</td>
</tr>
</tbody>
</table>

Executive Summary

This document reports on the V1 of the parliament tool application. The application has been developed according to schedule, while formal usability studies for specific social network related parts have been performed and are under evaluation (results to be reported in D9.2). This deliverable is the accompanying document to the V1 of the prototype that describes the functionalities developed so far.
Table of Contents

ARCOMEM Consortium ............................................................................................................. 2
Milestone participants .................................................................................................................. 3
Change Log.................................................................................................................................. 3
Executive Summary ...................................................................................................................... 3
Table of Contents .......................................................................................................................... 4
Glossary ........................................................................................................................................ 5
Introduction ................................................................................................................................... 6
Parliament Application: Search and Retrieval (SARA) ................................................................. 6
Database definition ......................................................................................................................... 10
Document/Object Store .................................................................................................................. 11
Knowledge Base ........................................................................................................................... 12
Technical aspects ........................................................................................................................... 13
Conclusion ....................................................................................................................................... 16
List of figures

Figure 1: Main results page 7
Figure 2: Advanced search 7
Figure 3: Item results page 8
Figure 4: Main page item information 9
Figure 5: Example of Document/Object store over HBase 12
Figure 6: Current Parliament Tool Functionality Layout 15

Glossary

Document/Object store: The Document/Object store is used to store the resources fetched by the crawler (HTML or API crawler) and is essentially an HBase.

Knowledge Base: Data base which contains extra attributes annotating web objects that derive either from the crawler (making use of the API of each crawled SMC, see D8.1 and D1.2) or from the analysis of the fetched content.
Introduction

The Parliament Tool prototype V.1 has been delivered in June 2012 and includes the front-end that is used to submit queries and view the results from the ARCOMEM database. It is the user interface to the ARCOMEM system for parliament users for both user groups, archivists and generic users. The users may search and browse the ARCOMEM Web Archive to find relevant content for an event, entity, topic. The Crawler Cockpit V1 that the archivists use to set up an archiving campaign is described in D8.1

Parliament Application: Search and Retrieval (SARA)

Members of Parliament and their scientific associates and assistants (journalists, archivists, political analysts, students and researchers, government officials and policy makers) investigate the fully annotated social media content using the SARA application. Users may use the ARCOMEM metadata and the application GUI to retrieve relevant social media content in a user friendly and efficient way from the web archive.

The V1 of the Parliament Application has implemented the following main functionalities:

Main functionalities

The main functionalities of the first version of the retrieval interface (Fig.1) are:

- Single search (Fig.1 point 3)
- Advanced search (Fig.2)
- Refine search
- Dynamic filtering via faceting (Fig.1 point 4)
- Sorting (by modality, by source SMC) (Fig.1 point 5, 6)
- Item viewing properties and derived functionalities:
  - Modality views (all, text, image, video, sound)
  - Social Media related information per item (current opinion, trending, latest tweets, source SMC)
  - Text/image analysis related information (entities, events)
  - Tag cloud support for major entities for quick refinement
  - SMC and text/image analysis items with linked specific functions (under investigation for T9.4, preliminary functions include search refinement, new search)
- Generic viewing properties:
  - Recent searches
  - Recent campaigns (as linked to the crawler cockpit application)
Latest news (promotional)

Figure 1: Main results page

Figure 2: Advanced search

Also in Figure 1, the Parliament Tool’s SARA interface highlighting the main placeholders and associated functions/visuals

1. User login and ARCOMEM logo
2. Top level menu reserved for high level functions
3. Search box (also toggling single and advanced search)
4. Dynamic filtering via facets
5. Main results page with modality sorting tabs
6. SMC filtering (results are filtered by selecting Social Media icons - toggling)
7. Timeline view button

The item results page is shown in Figure 3

Figure 3: Item results page

The item results page provides all retrieved information about an item. The standard information includes:

- Title
- Details
- Date
- Provider
• Format
• Description

The ARCOMEM specific information includes, so far for V1:
• the identified associated entities
• the identified events
• a tag cloud of entities derived from items that include the current item’s entities/events
• The related events of the current identified event (if any)
• the latest tweets that contributed the major entities for the current item
• the major twitter accounts that contributed to the above
• the timeline that shows the positive-negative social network input regarding the associated entities (currently only for Twitter as of V1)
• the lists of positive and negative Tweets for the above

Specific ARCOMEM derived information is also used on the main results page in order to enable the users to make informed decisions on items to examine, as shown in Figure 4:

Figure 4: Main page item information

The information presented includes the current state of single item information within the main result panel:

Current opinion associated with the main entities/events of the item

Source SMC, trending (opinion), and button for more details
The item information page will also have a link to the archived content item (document). This functionality will be implemented in V2 when the WARC files for a campaign will be available by the ARCOMEM system.

**Database definition**

The database plays an important role in the architecture of the ARCOMEM Parliament Tool, as it provides the storing, indexing and retrieving functionality for all the data collected by the crawler, utilized by the archivists for the crawling refinement process and searched upon by the generic users. As such, it is expected to support various types of data, handle updates and accommodate many types of queries.

The required functionality of the database is defined by its interaction with the rest of the components of the Parliament Tool architecture.

The database is expected to store: a) the original web content fetched by the crawler during the crawling campaign, b) metadata, meaning some extra attributes that derive from the application aware crawling (e.g. author information, #retweets for a tweet etc.).

The database should be able to answer quantitative as well as qualitative related queries posed by the archivists to further guide the resource harvesting process of the crawler.

General users must be able to navigate through the stored content by posing queries that concern either the annotations of the application aware crawling or the semantic information derived after the analysis of the content.

The above-mentioned functional requirements must be accompanied with some non-functional properties, dictated by the anticipated sheer volume of crawled data and by several operational constraints.

**Scalability:** The volume of the information available on the Internet has surpassed the zettabyte barrier and keeps growing at an astounding rate. This fact combined with the requirement of our system to capture multiple versions of web objects over time creates enormous demands for storage as well as memory. It is thus important for the storage module to be able to scale out not only to accommodate the increasing size of the data but also to provide high availability at low cost.

**Performance:** The module should be able to resolve some queries in near real-time, as some decisions are made at runtime (by the crawling module). High throughput is important even for complex analytics tasks, since they may trigger a large number of queries and thus take hours or even days to complete.
Robustness: A large number of storage and computational resources will be required to cooperate in order to achieve the aforementioned functionality. It is imperative that this functionality is not hindered by node failures and data can be reliably stored and retrieved.

Ease of use: The storage module should be easy to deploy and maintain.

To meet all the requirements of the Parliament Tool application, we have opted for the combination of MapReduce and NoSQL databases, technologies that are widely used to index huge amounts of data that reside in distributed storage facilities and offer scalability and availability at low cost.

Taking all the above observations into account, we propose a storage module consisting of two components: the Document/Object store and the Knowledge Base.

**Document/Object Store**

The Document/Object store is used to store the resources fetched by the crawler (HTML or API crawler) and is essentially an HBase\(^1\) table. HBase is a non-relational, distributed database modeled after Google's BigTable\(^2\). Like all DBMS in this class, it sacrifices some query expressiveness (compared to SQL DBMSs) to achieve extreme scalability. It is also very flexible in the columns definition: each table is a sparse map associating a value to (key, column family, qualifier, date)-tuples.

Storing the output of crawls requires the ability to store a large number of resources (especially when adding individual web objects), representing a large overall volume. In addition, different type of extra data must be attached to different type of resources. HBase covers all these needs.

In order to provide high availability of the data, HBase keeps its data in a distributed filesystem called HDFS. This approach also diminishes the impact of a node failure on the overall database system performance.

An HBase table consists of a large number of sorted rows indexed by a row key and columns indexed by a column key (each row can have multiple different columns). Actual content is stored in HBase cells: an HBase cell is defined by a combination of a row and a column key, in the same way an (x,y) value defines a point in a 2-dimensional space. The primary key of an HBase table is the row key. HBase supports two basic lookup operations on the row key: exact match and range scan.

A sample of the Document/Object store is shown in Figure 5. The URL of the fetched web object, whether it is text, image or video, will serve as the row key ([www.cnn.com](http://www.cnn.com) in our case), while the

---

actual binary content will be stored under “content”. Extra attributes can be stored in other table columns; however, since the row key is the primary one, queries upon them require the scanning of the whole table and might be inefficient. Moreover, it is possible to store many timestamped versions of a fetched web object under the same row key. This is important, since the crawling procedure is likely to discover altered version of already stored web objects in different crawls.

It is important to note that the Document/Object store allows a posteriori addition of attributes as well. In the meanwhile, we are considering storing for each crawled resource besides the original data, for instance, extracted links pointing out from the page, or the list of pointers to resources embedded in the page, e.g. images. In general, such additional information may be stored at crawl time if available or might constitute a result of a post processing done arbitrarily upon the crawled data. Therefore, the data model can be enriched on demand.

The APIs for communication with the database are very similar to those known from the SQL/ODBC world. For instance, inserting into the database, a data object is created, the values for the particular column qualifiers for the before mentioned column families are set as raw byte values and sent to the server. The inserts are cumulated in a buffer of a predefined size to minimize the network communication overhead.

The querying is done using either an approach of a point query – one particular row is retrieved from the Document/Object store using the URL and API Get or a range of URLs is sequentially read using the Scanner API. In both cases the user can control which versions of the datum – a web resource is to be retrieved.

The Document/Object store can be accessed through a java client API.

![Figure 5: Example of Document/Object store over HBase](image)

Knowledge Base

Extra attributes annotating web objects that derive either from the crawler (making use of the API of each crawled SMC) or from the analysis of the fetched content play an important role in the refinement of the crawling process, performed by the archivists, as well as the search of available
resources, performed by the general users. Such attributes are either related to the contributors of web objects, e.g., nicknames of twitter users, their location, the number of their followers etc. or to the web objects themselves, e.g., the date of a tweet, its number of retweets, etc. It is thus essential to store and efficiently index these annotation attributes in a way that guarantees their fast retrieval.

In general, such a data store should provide store and retrieve functionality. Using a plain HBase to store this information would prove inadequate. As mentioned before, HBase provides efficient searching functionality only on row key queries. We propose a solution that allows reasoning over RDF triples that are stored on top of HBase. Such a solution can be used to store the semantic information derived from the content analysis and will be used to enhance the Parliament Tool with semantic functionality at a later stage. In a similar manner, api-crawler annotations can also be stored if expressed as RDF triples.

The idea behind our solution is to allow the resolution of complex SPARQL queries using the scalability and high performance of HBase. Our approach is based on the centralized Hexastore indexing scheme. Hexastore is an RDF store that creates 6 indices (for all possible permutations of subject, predicate and object – SPO, PSO, POS, OPS, SOP, OSP) in main memory, thus offering the ability to retrieve any triple pattern with minimal cost. Taking this one step further, our distributed Knowledge Base creates 3 indices ultimately stored in an HBase table. These indices correspond to the combinations of S, P and O, namely SP_O, PO_P and OS_P. The Knowledge Base provides native SPARQL query functionality, with joins being executed as MapReduce jobs.

The above functionality is exposed through a java and a python client API.

**Technical aspects**

The graphical interface of the solution has been implemented in Java, following a Model-View-Controller (MVC) design pattern. The used framework is Java Server Faces (JSF), a java framework that implements the MVC pattern. On top of the framework we have used Facelets, an open source web template system as well as richfaces, an off-the-shelf ajax component library for JSF. Any additional effects and ajax functionality has been added with jQuery, a popular javascript library used in most of modern web interfaces. Finally, there is a plan to use Modernizr library together with CSS3/HTML5 to provide an overall solution for multiple clients (e.g. smartphones, tablets etc). For security and authentication purposes the Spring-security framework has been added. For now, the number of users is significantly low, so the credentials are maintained in local

---

configuration files. In the future, if the number of users increases, the plan is to implement a sign-up functionality with the backup of a separate Database server like MySQL.

The back-end of the solution is also based on Java programming language. The JSF backing beans are directly communicating with the data storage by means of HTTP messages. So, the solution follows a SOA approach and relies on loosely coupled distributed services that expose a RESTful HTTP interface and can be consumed by the frontend. In the case of Arcomem Parliament Tool, the most important backend service is that of Solr Search Engine, which acts as the full-text-search repository in the platform. Apache Solr Server is a very efficient and stable search server developed on top of Lucene search libraries. The processed archive resides in Solr Server and is fully searchable through the graphical interface. The configuration file of Solr has been tailored in order to meet the needs of ARCOMEM Parliament Tool.

More specifically, the schema.xml configuration file of Solr is modified with adding the following "arcomem" fields:

```xml
<!-- ARCOMEM FIELDS -->

<field name="title" type="text_general" indexed="true" stored="true" multiValued="false"/>
<field name="url" type="string" indexed="true" stored="true" multiValued="false"/>
<field name="details" type="text_general" indexed="true" stored="true" multiValued="false"/>
<field name="type" type="string" indexed="true" stored="true" multiValued="false"/>
<field name="provider" type="string" indexed="true" stored="true" multiValued="false"/>
<field name="location" type="string" indexed="true" stored="true" multiValued="false"/>
<field name="date" type="date" indexed="true" stored="true" multiValued="false"/>
<field name="source" type="string" indexed="false" stored="true" multiValued="false"/>
<field name="entities" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="places" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="published" type="date" indexed="true" stored="true" multiValued="false" default="NOW" multiValued="false"/>
<field name="network" type="string" indexed="true" stored="true" multiValued="false"/>
<field name="opinion" type="string" indexed="true" stored="true" multiValued="false"/>
<field name="trending" type="string" indexed="true" stored="true" multiValued="false"/>
```

The search can be performed on any of these fields while the facets returned are defined by the current query. Normally, the following facets are returned after every query submitted by the user, as seen in the related Java source code. These fields can be easily modified even on the fly by changing these few lines of code.

```java
solrQuery.addFacetField("network");
solrQuery.addFacetField("type");
solrQuery.addFacetField("location");
solrQuery.addFacetField("provider");
solrQuery.addFacetField("entities");
solrQuery.addFacetField("events");
```
Since the solution is based on open-source cross platform technologies, there is no restriction about the Operating System. In the current instance, we have selected Ubuntu Linux. Finally, the whole web application (Solr, UI and Java Back-end) are hosted in an Apache Tomcat server, acting as the application server of the system.

Solr Server index is populated with the output of the crawler. For this purpose, we have added a final step to the crawling workflow, adding the “indexing” module. The indexing module is triggered after a bunch of crawled items is analysed and stored in the HBase and RDF Repository. The role of the “indexing” module is to traverse these repositories and retrieve all the information related to each crawled item. In this way, it converts the RDF triplets and the raw data into a Solr-compatible document which contains all the indexable fields supported by Solr. In the current release of the platform, the indexing module is triggered manually. This means that the migration of the collected archive into the Solr Engine is performed on demand and is searchable after this migration finishes. The detailed functionality and the design of the entire solution is depicted in the following diagram:

![Figure 6: Current Parliament Tool Functionality Layout](image-url)
Conclusion

This document presented the V1 of the Parliament Tool, presenting the Search and Retrieval Application functionalities and user interface. This version was already used for targeted usability evaluation and will be used for full formal evaluation by the Parliament users in order to determine ways to discover and process the next set of the functional and non-functional requirements that will lead the implementation into V2. D9.2 will elaborate on the use of the SARA application both for Broadcasters and for Parliament as well as discuss the feedback from the end users.