D7.1

Architecture Design

v 1.10 / 2015-04-10

George Mourikas (HWC), Symeon Papadopoulos (CERTH), Adrian Popescu (CEA), Timotheros Kastrinogiannis (VELTI), Theodoros Michalareas (VELTI)

This document defines the architecture design of the USEMP software. The architecture is based on the user scenario role, technical aspects of the implementation and functional aspects of individual elements provided by deliverables D2.1 and D2.2. Deliverable D2.1 is based on task T2.1 Use cases analysis and specification and deliverable D2.2 is based on task T2.2 Requirements analysis.

Project acronym	USEMP
Full title	User Empowerment for Enhanced Online Presence Management
Grant agreement number	611596
Funding scheme	Specific Targeted Research Project (STREP)
Work program topic	Objective ICT-2013.1.7 Future Internet Research Experimentation
Project start date	2013-10-01
Project Duration	36 months

Workpackage	WP7
Deliverable lead org.	HWC
Deliverable type	Report
Authors	George Mourikas (HWC) Timotheros Kastrinogiannis (VELTI) Symeon Papadopoulos (CERTH) Adrian Popescu (CEA) Theodoros Michalareas (VELTI)
Reviewers	Laurence Claeys (iMinds) Theodoros Michalareas (Velti)
Version	
Status	Final
Dissemination level	PU: Public
Due date	2014-05-31
Delivery date	2014-07-24
Revision date	2015-04-10

Version	Changes
0.1	Start of doc
0.2	Template for components build up (section 5)
0.3	After teleconference on 7/3/2014 corrections added to this document rename VALUE DB to PRIVACY DB, Java based code interacting with HADOOP
0.4	Added explanation to 1.3.1, 1.3.2, (small change in 1.3.3, 1.3.4. and 1.3.5.).
0.5	Added a top-level architecture in Section 4.1.1, 4.1.2, 4.1.3 created a separate template-file for subsystem specification (‘Subsystem specifications template.dotx’ utilised when the top-level Technical architecture is agreed)
0.6	Edited the document based on Timos email
1.0	Additions based on TelCo#2
1.1	Added the changes according to feedback from Telco#3 to Section 1 and 4. Included feedback from the consortium members on the TelCo “2.1 and 7.1 Alignment” that is included in Section 1
1.2	Included the Top-level Technical architecture (Section 3), divided the work to achieve the deadline in the comments
1.3	TelCo#5 and #6 agreed table of content included
1.4	From TelCo#7 section 3 big part is added, from v1.3 corrections from Symeon Papadopoulos are added
1.5	From TelCo#8 and includes contribution from CEA, VELTI and CERTH
1.6	TelCo#9 and #10 Clean-up formatting ready for review
1.7	Integrating reviewers’ corrections
1.8	Rename ‘LIO’ to ‘DataBait’ (Annual review recommendation)
1.9	Following up on the Annual review recommendation
1.10	Final changes and restructuring to review comments.

Table of Contents

Table of Figures

List of Tables

1. Concept Architecture

1.1. Introduction

This document defines the architecture and design of the software to meet the requirements and goals of the USEMP project. It provides also the designs of the methods for the collection of user data from online social networks and web browsing (via the use of web browser plugins) which are then provided as feeds to the USEMP platform for the evaluation of privacy and personal data value scores which are then presented to the end-users. For the rest of the document and the duration of the project the consortium has decided to refer to the collection of USEMP developer tools and applications as 'DataBait'.

The work carried out for the present report in order to design and specify the 'DataBait' USEMP set of tools, is based on WP2 and WP6 results and particularly follows the steps described below:

Pillar I – Requirements Analysis

Step 1. **Studying WP2 Use Case and Corresponding USTs,** towards deriving USEMP Tools functional requirements and corresponding expected features that will facilitate their efficient realisation.
Step 2. **Defining Concrete Technical Requirements and Expected Features**, in order to recognise the required DataBait technology components that will enable them.
Step 3. **Defining USEMP System Data & Information Models,** in terms of end users' personal data that need to be collected and stored by USEMP Tools towards facilitating the operation of the envisioned USEMP Services (namely, USEMP back-end System Data Modelling).

Pillar II –Analysis and Designing Principles

Step 4. **Analysing Existing Technology Stacks,** for each of DataBait components/plug-ins and corresponding emerging technical architectures
Step 5. Justifying the Selected Solution, in terms of specifications and requirements fulfilment, as well as expandability and scalability attributes.

Pillar III – Design & Integration

Step 6. **Designing software components (web applications, browser plug-ins, backend services)** in terms of architecture components, interfaces, API and technologies that will be used following a web services based SOA approach (service-oriented architecture)
Step 7. **Specifying Existing Open Source Projects,** that can be used and extended as part of DataBait to implement envisioned features. This would allow us not only to maximize the efficiency of the project's development resources, but also to set the basis for an open source based overall solution.
Step 8. **Integrating Individual software components/plug-ins to an overall USEMP architecture** justifying and reassuring their seamless integration

1.2. Concept Overview

The main function of the system is to offer the Online Social Network(s) User actor (OSN_USER) a set of tools that provide the functionalities described by the use cases presented in WP2. Interactions of the OSN_USER include a group of graphical user interfaces called USEMP_TOOLS, available after installation and registration in the USEMP_SYSTEM. The USEMP_TOOLS are considered 'client side tools' that will interact (via internet connection) with the USEMP server-side (USEMP_SS) 'backend services' part of the USEMP_SYSTEM. The OSN_USER will not have access to USEMP_SS other than credentials authentication, sending seamlessly to USEMP_SS the data collected by the USEMP_TOOLS and the processed result from the USEMP_SS.

The USEMP_SYSTEM is separated between the 'client side tools' and the 'backend services/processes'. The 'client side tools' are three (3) applications with main graphical user interfaces that the OSN_USER (end-consumer) will interact with. The client side tools (DataBait-FB, DataBait_BROWSER and DataBait-GUI) are available to users that have registered with Databait and they protected by a logging-in process. The developed USEMP_TOOLS serve the followup purposes in USEMP:

- **DataBait_OSN** is a plug-in made specific for each supported OSN, as a 1st example Facebook OSN is considered, additional social networks can be supported. The plugin is configured specifically to be able to access (with the permission of the OSN_USER) the digital-trail (see mind mapping in APPENDIX E.) generated by the OSN_USER while interacting with the OSN (for example users posts/comments/tweets/pokes/share images or similar share personal data). In the 1st realization of the Databait system DataBait_FB is implemented to support Facebook social network integration.
- **DataBait_BROWSER** is an application in the form of a **web browser plug-in** made specifically to collect the digital-trail of the end-consumer information (see mind mapping in APPENDIX E.) generated while interacting with the OSN_USER web browser where the plug-in is installed.
- **DataBait-GUI** is an application that acts as the graphical user interface through which the USEMP_SYSTEM can convey to the OSN_USER results of the USEMP_SYSTEM processes and services (done by utilising the backend services and processes).

The USEMP_SYSTEM client side tools interface with the backend services/processes by a set of USEMP custom-made Application Programming Interfaces (USEMP_API). All the backend processes occur in a cloud environment. The collection of data temporary stored in the client side will be streamed to the USEMP_SYSTEM backend services using the USEMP_APIs. A local storage strategy will be employed for the USEMP_SYSTEM to temporary store and clean after streaming is complete.

The information acquired by DataBait-FB and DataBait_BROWSER will use the USEMP_APIs to transfer the collected information to be stored in within the USEMP_SYSTEM backend storage area & database identified as 'HISTORICAL_DATA_DB'. Before the communication of the acquired information to the backend can be initiated, a transparent authorisation/authentication of the client credentials is completed. USEMP_APIs implement the interface between the client side and the backend of the USEMP_SYSTEM including transparent authorisation and authentication of

the information credentials step with the ‘Identity Manager’ (USEMP_SYSTEM backend service).

Access to the USEMP_TOOLS by the OSN_USER requires installation and registration steps to the USEMP_SYSTEM. In the registration step authorisation and authentication credentials are created with the unique OSN_USER identity. Authorisation and Authentication steps in the concept Architecture are implemented by the DataBait_AA interface. The DataBait_AA is interfaced via the USEMP_API interface with the backend service “Identity Manager”. The “Identity Manager” is a backend service which verifies/authorises the in/out-going information-credentials between the client side and the backend services of the USEMP_SYSTEM.

Information communicated from the USEMP_TOOLS to the backend regarding the processing elements of Technical Components is the ‘HISTORICAL_DATA_DB’ dataset and a collection of information from open/public Data-Sources. The type/sources of the open/public information is dictated by the input-information-structure of each Technical Component. The Technical Components are a set of processes that analyse aspects of the ‘HISTORICAL_DATA_DB’ against the open/public information datasets. The metadata produced by the Technical Component (TC_metadata) is provided for additional processing by the PRIVACY PROFILING and VALUE ESTIMATE processes.

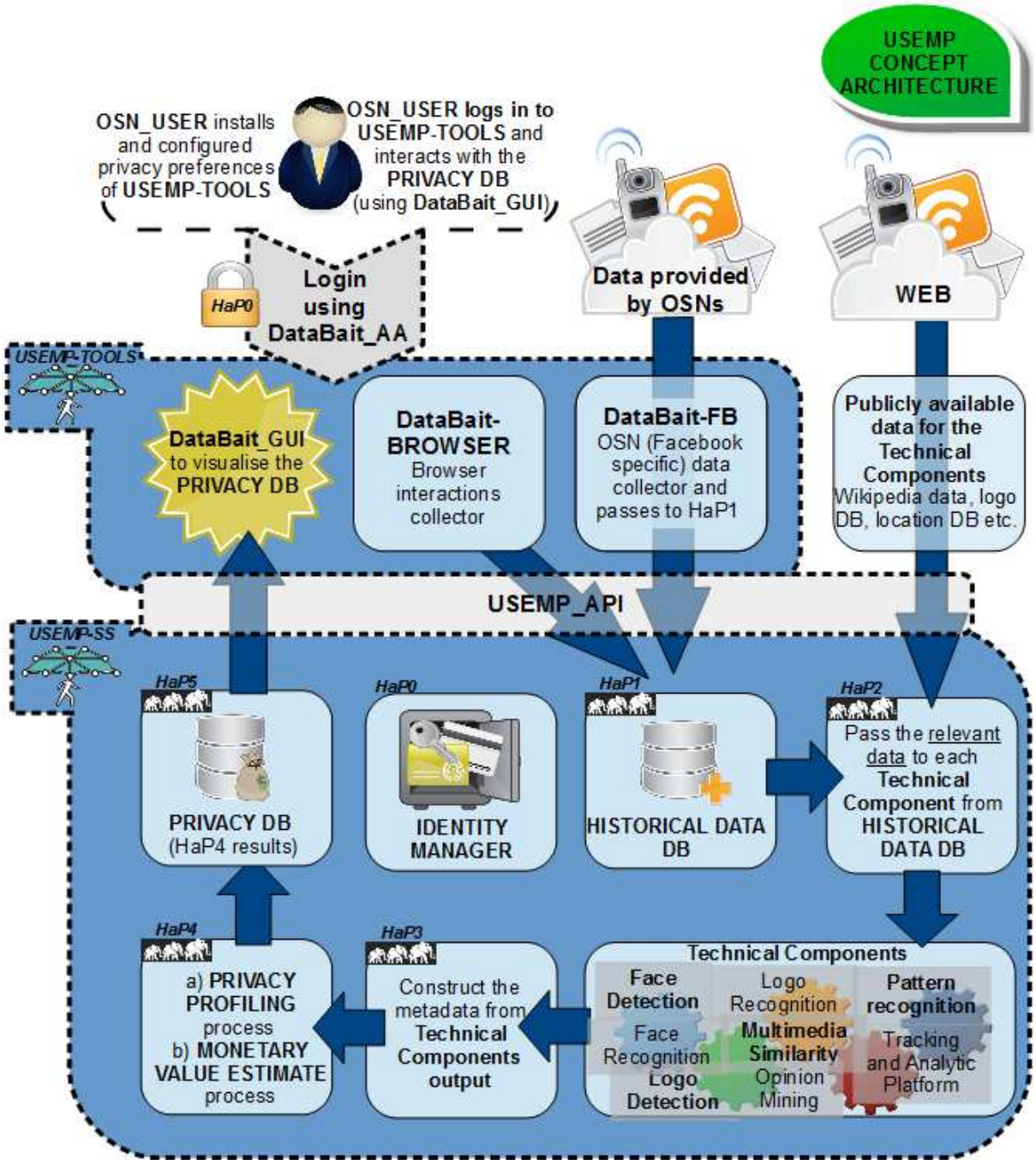


Figure 1. USEMP Concept Architecture depicting the main software components and information flows for Databait tools

1.3. Social Networks data integration

One of the basic requirements of the Databait tool as described in deliverable D2.1 is to provide users with additional information related to their operations in Online Social Networks related to privacy and personal data value. Additionally the tools developed in WP6 require access to OSN user personal data. It is therefore essential for the Databait system to collect such information from OSNs and to make if available for further analysis. In order to integrate user personal data into Databait a modular an open interface architecture has been followed.

Databait architecture allows to develop a OSN-specific application that follows the limitations and facilities provided by each different OSN that collects personal data associates them with a Databait user and makes them available through a common format. For the 1st realization of the Databait system Facebook has been chosen as the primary OSN from which data will be collated and integrated into the system. While in this document the focus is on interactions and specifications for Facebook additional OSNs can be added following the proposed open interface and modular approach. For each additional OSN additional applications added at a later stage in the project, such as Twitter, or newer social networks such as Google Plus, Instagram, Vine, provided they can support an open interface for the collection of user personal data. These can then be developed as DataBait_OSN specific modules and provide data to HISTORICAL_DATA_DB.
Additionally Databait architecture allows for a 2nd method to collect data from OSNs without residing to open interfaces from OSNs but to the use of the DataBait_BROWSER plugin. DataBait_BROWSER plugin is developed as a flexible web browser component that has access to all the web browsing history of an OSN_USER that uses the web browser (for example Chrome browser) to access OSN. DataBait_BROWSER plugin is designed to collect references to multimedia items shared by the OSN_USER and transfer these references to the Databait backend services so that these are collected if they are publically available. Additionally DataBait_BROWSER plugin can be easily modified to seek and collect references to additional public items that are shared through the web browser, for example comments or posts if this is desirable (for example if it is not possible to use an OSN open interface to collect such data). Both supported methods for the collection of OSN data (use of OSNs open interface and use of DataBait_BROWSER) provide a very powerful framework that allow the Databait tools to integrate any of the popular OSNs that users utilize over their web browser.

1.4. User Interactions

There are a number of ways in which the user will interact with the system, whether that be actively or passively. These interactions are with the browser, and via the browser plugin which provides browsing habits and tracker activity, via the OSN directly, from which an interface is provided when a user signs up to the system via the DataBait GUI interface, from which most user activity is expected to take place.

In order DataBait_BROWSER to be used by a USEMP user, the user should:

- Download it from the corresponding browser store that will be published (for example Chrome Web Store or Mozilla addon) or alternatively download the addon from Databait web site;
- Install DataBait_BROWSER into their browsers and then sign-in with the USEMP User account.

In order DataBait_OSN to be used by a USEMP user, the user should:

- Access DataBait_OSN URL via their web/mobile web browsers;
- Login with their corresponding OSN account and approve the required privacy settings;
- The collected data are forwarded to Databait backend services where number of privacy/personal value scores are computed. As a benefit/reward Databait_OSN

USEMP users can experience a visualization of such scores in Databait_GUI and select certain actions that can be applied with respect to their privacy (an example of such a system that demonstrates the overall flow similar to the that that will be developed by USEMP but with different focus/visualizations and scores is available at: http://www.youarewhatyoulike.org)

In order for DataBait_GUI to be used by a USEMP user, the user should:

- *Access DataBait_GUI URL via their web/mobile browsers;*
- *Login with their corresponding USEMP account credentials;*

1.5. Requirements Mapping

This section will map the requirements defined in WP2 to components and modules within the overall system.

1.5.1. Functionality Requirements Analysis

In this section, an overview of fundamental USEMP tools functionality requirements is provided (in line with WP2 use cases and USTs), in terms of user-centric features and functionalities that should be supported by the latter towards a) enabling USEMP services, b) optimising end-users experience and finally, c) reassuring end-users privacy. Moreover, the latter are projected to the main three DataBait plug-in towards identifying the key features supported by each USEMP tool.

The first three features (as presented in Table 1 USEMP Tools Functionality Requirements) are related to USEMP users' personal volunteered and behavioural data collection, in order USEMP envisioned architecture and services to operate. Thus, this is not applicable for DataBait_GUI but mainly concerns the expected functionality of DataBait_OSN and DataBait_BROWSER. The only emerging limitation is related to users' historical data (i.e. personal data created prior to user's registration in DataBait (USEMP System)) collection via their browser, due to the existence of limited historical data stored locally by a the commonly used browser.

The next four features are related to USEMP users' ability to explicitly control the operation of DataBait, either in terms of activating/deactivating their overall operation, or in terms of allowing/forbidding their access to users' personal data generated or disseminated via specific domains (or even specific URLs). The only emerging limitations are related to DataBait_OSN and changes in policy with respect to consumers' privacy settings, since the same privacy settings are applied by default to all the pages of an OSN application. OSN users can either accept all of the privacy rules at once, upon Logging to the OSN (e.g., FB log in) or deny them. Due to the latter limitation, fine grained do not track policies cannot be applied in the case of DataBait_OSN. The user can logout from an OSN and thus, forbid the latter from accessing the created personal data.

Feature eight from Table 1 USEMP Tools Functionality Requirements is related to DataBait ability to enable users to prevent online tracking services from tracking their digital/social personal data, while feature nine is related to the support of intuitive and non-intrusive notifications towards informing USEMP users in real-time on major privacy leaks. The final feature is related to users' ability to use USEMP services from multiple devices/browsers while having a single USEMP account.

Feature	Features Name	Feature Description	DataBait_OSN	DataBait_BROWSER	DataBait_GUI
1	**Real Time Data Collection**	User's personal/behavioural data should be collected in real-time.	YES	YES	Not Applicable
2	**Historical Data Collection**	Users historical (created in the past, prior registering in DataBait) personal/behavioural data should be collected.	YES	Not Feasible (or Limited)	Not Applicable
3	**Real-time Data Visualization**	Users should be able to visualize the outcome of USEMP System in real-time.	Not Applicable	Not Applicable	YES
4	**Terms of Use (ToU)**	Users should be able to clearly identify and read USEMP ToU.	YES	YES	YES
5	**Plug-in Deactivation**	Users should be able to pause/stop USEMP plug-in's operation.	YES	YES	YES
6	**Access Policy Management**	Users should be able to clearly define domains from which no data are collected by the USEMP plug-in. Forbidden domain preferences should be stored locally in order to reassure privacy.	Not Applicable	YES	Not Applicable
7	**Forbid Plug-in Operation For Specific URLs**	Users should be able to clearly define URLs from which no data are collected by the USEMP plug-in. Forbidden URLs should be stored locally in order to reassure privacy).	Not Applicable	YES	Not Applicable
8	**Do Not Track Enable**	Enable users to forbid Specific Online Tracking Services from tracking their personal data.	Not Feasible	YES	Not Applicable
9	**Support Pop-Up Notifications**	Support Pop-Up Notification towards informing users on potential personal data privacy leaks and compromises	YES	YES (For specific types of actions e.g., upload an image).	YES
10	**Unique User Identification**	- Users should be able to create their own USEMP account and uniquely identified across all user plug-ins.	YES	YES	YES

Table 1 USEMP Tools Functionality Requirements

1.5.2. Technical Requirements Analysis

In this section, a list of key fundamental USEMP Tools technical requirements is provided (in line with WP2 use cases and USTs), in terms of cross platforms and devices compatibility, easiness of use and operation's efficacy.

Feature	Technical Requirements	DataBait_OSN	DataBait_BROWSER	DataBait_GUI
1	**Web Browser Compatibility**	Chrome, Safari, Firefox	Chrome, Safari (and/or Firefox)	Chrome, Safari (and/or Firefox)
2	**Mobile Browser Compatibility**	Chrome, Safari, Firefox	NO (Low Priority)	Chrome, Safari (and/or Firefox)
3	**Deployment / Installation**	No Installation Required	Installation is Required (per browser per device/pc)	No Installation Required
4	**Tracking/Operation Traffic Management** (towards QoE maximisation, minimizing browser overhead)	YES	YES	Not Applicable
5	**Avoid Denial of Service from the target 3rd Party OSN Service (e.g. FB)**	YES	Not Applicable	Not Applicable

Table 2 USEMP Tools Technical Requirements

The first two requirements are related to DataBait's compatibility across devices, browsers and operating systems. Due the plethora of existing options/solutions in all three categories, emphasis will be placed to the most popular ones. Specifically, regarding OS, Android and iOS have been selected for mobile devices, while MS Windows and Apple MAC for desktop. Consequently, browsers compatibility options for the DataBait plug-in/add-ons and web apps are limited to Chrome, Firefox and Safari (as the most commonly used per OS[1]). Finally, given overall project's effort limitations and in order to properly proof the cross-platforms feasibility of the proposed solutions DataBait web apps i.e. (DataBait_OSN and DataBait_GUI) will target on Chrome, Safari and Firefox latest versions both for desktop and mobile, while DataBait add-ons (DataBait_BROWSER) will manly target Firefox, Chrome and Safari primarily regarding desktop browsers (mobile add-ons are still within the scope of the project but as a secondary goal).

The third requirement is related to USEMP users' easiness in using a DataBait. To that end, the less required actions for a user to access/use a USEMP tool the more efficient is for her/him to access and use it. Thus, for DataBait web apps i.e. (DataBait_OSN and DataBait_GUI) no installation process is required, since the user just uses a URL via her/his browser to access them, while DataBait add-ons (DataBait_BROWSER) should be installed by the user in every browser she/he is using (via a simple process[2]).

The final two requirements aim at reassuring that the use of DataBait's is not compromising (decreasing) USEMP users overall QoE from the devices and/or browsers that the latter operate. Specifically, requirement four is mainly related to the operation of DataBait's that track, collect and feed USEMP back-end with end-users personal/behavioural data. Thus, special attention has been placed on attributing DataBait with features that minimise the imposed network traffic overhead in order not to affect the performance of the related devices/browsers. Finally, the fifth requirement is related to the operation of OSNs and especially, to the limitations that the latter impose in the way and frequency that OSN users' data (stored at OSN's back-end) are accessed via the exposed OSN API. In simple words, if a DataBait_OSN is accessing a user's OSN (e.g., FB) personal data in a ratio greater than the defined API calls ratio threshold, then the OSN (in most

[1] http://www.w3schools.com/browsers/browsers_stats.asp
[2] http://en.wikipedia.org/wiki/Plug-in_(computing)

cases) will forbid further access to OSNs API. To avoid the latter, DataBait_OSN should be attributing with proper throttling mechanisms in order to reassure their efficient operation.

1.5.3. Information Modelling Requirements

Prior concluding this sections, and proceeding with the analysis of each individual USEMP Tool, i.e., DataBait_BROWSER, DataBait_OSN and DataBait_GUI, in this section we provide a thorough analysis of **USEMP Tools Data and Information Modelling**. The latter model assembles all the information and data types the USEMP System and the envisioned novel algorithms and sub-components need in order to properly and efficiently operate. Therefore, since the information required by USEMP System is related mainly to USEMP end users' online digital personal and behavioural data, the proficient collection of the latter consist a fundamental requirement that drives the design and implementation of DataBait's, especially DataBait_BROWSER and DataBait_OSN. To that end, the aggregation of data/information requirements founded on WP2 scenarios and USTs, on WP6 algorithms data requirements and WP7 USEMP_SS (services and back-end) has been performed, presented in detail in **APPENDIX D.**

Overall, more than **forty data types** have been identified with more than **one hundred distinct data objects** consisting of a plural of DataBait's data model with large variety. Moreover, a concrete taxonomy of the latter has been performed, based on the source that a personal data is generated (i.e., web browser, OSN, third party service), since DataBait must a track and acquire a user's personal data and the time of creation in order to properly and in real time inform the user for potential privacy leaks. Thus, the following user personal data categories have been identified:

- USEMP Users' Real-time Personal volunteered and Behavioural Browsers Data
  - **Definition:** Personal, volunteered and behavioural data generated and/or disseminated by the end user via her/his web browser either explicitly (e.g., the upload of an image, the post of a news' post etc.) or implicitly (e.g., the URL of a page that she/he visited, the time she/he spent in a blog).
  - **Collector:** DataBait_BROWSER
  - **Data Model Attributes:** Name, Data Type (e.g., URL), Metadata (e.g., Event Time Stamp, Source URL, etc.) and Collection Frequency (On Click, On User Action, etc.).
  - **Detail Data Definition:** APPENDIX D., Table A.
- Internet Services that Track user's Data
  - **Definition:** Personal and behavioural data implicitly created by a user via her/his web browser due to the existence of 3rd party services that monitor her/his digital trail (e.g., advertising services, analytics services, ad networks, brands also see mind mapping of the digital trail data-access level structure in APPENDIX E.).
  - **Collector:** DataBait_BROWSER
  - **Data Model Attributes:** Name (e.g., tracker's name), Data Type (e.g., URL), Metadata (e.g., Tracker ID, Tracker Type etc.) and Collection Frequency (On Click, On User Action, etc.).
  - **Detail Data Definition:** APPENDIX D., Table B.
- User Historical and Real-time OSN Data
  - **Definition:** Personal, volunteered and behavioural data generated and/or disseminated by the end user via her/his OSN (for example Facebook) either historical (i.e., data generated by the user prior creating a USEMP user

account, stored in the OSN system) or real-time (i.e., data generated by the user upon creating a USEMP account). The latter data will be collected via OSN's exposed API.
    - **Collector:** DataBait_OSN
    - **Data Model Attributes:** Name, OSN Object Name, Data Type (e.g., JSON, URL, etc.), Metadata (e.g., Event Time Stamp, Source URL, etc.) and Collection Frequency (Periodically, On User Action, etc.).
    - **Detail Data Definition:** APPENDIX D., Table C.
- User Personal Data for Training USEMP System Algorithms
    - **Definition:** USEMP Users volunteered and behavioural data that need to be collected in order USEMP algorithms to be trained (automated fine tuning) towards maximising the accuracy of the derived outcomes. These data are a subset of the ones in the previous three categories, but what differentiates them is the frequency that they are collected.
    - **Collector:** DataBait_BROWSER and DataBait_OSN
    - **Detail Data Definition:** APPENDIX D. , Table D.

Concluding this sections analysis, let us underline that the presented USEMP Tools Data Model is a constantly evolving schema based on the need and features added in USEMP System.

2. Component Services

2.1. Client Side Tools

USEMP Client-side Architecture components presented in Figure 1, denoted as USEMP_TOOLS, main objective is to enable USEMP users to directly or indirectly interact with the overall USEMP System (i.e., USEMP_SS).

Individual USEMP Tools are referred to with the term “DataBait Plug-in(s)” (denoted as DataBait). The notion “plug-in” is deliberately selected since all DataBait components are software that add a specific feature to an existing software application (e.g., a web browser), commonly denoted as plugins, extensions or add-ons[3]. USEMP Tools (i.e. DataBait(s)) will serve a two-fold goal.

- Will serve as USEMP users’ graphical interfaces enabling them to access, visualise, exploit and interact with USEMP services and features via their computers and/or mobile devices (supporting various device and browser types (e.g., desktop pc, tablets and mobile devices)
- Will serve as end-users’ personal/behavioural/OSN personal data tracking/collection mechanisms. The collected data via DataBait will be then provided to USEMP back-end for further processing and analysis via USEMP_API (application programming interface[4]). In order to support USEMP use cases, there are two types of API’s that will be developed for collecting end-users’ data:
    - DataBait_OSN serving as end-users’ Historical & Real-Time OSN Data Collectors
    - DataBait_BROWSER serving as end-users’ Real-Time Behavioural/Personal Browser Data Collectors

2.1.1. USEMP-TOOLS: DataBait_BROWSER

DataBait_BROWSER is browser plug-in supporting the most popular web browsers i.e., Chrome, Firefox and Safari, and will serve as users’ digital trail tracker (see mind mapping in APPENDIX E.) and aggregator, specific to the browser utilised by the data-subject (USEMP User).

Key Expected Features

- Track USEMP users’ personal and behavioural data generated via their web browser(s) (either explicitly or implicitly) and feed the latter into USEMP back-end system via USEMP API, denoted as **USEMP_API.**
- Identify the USEMP users’ trackers, in terms of 3rd party online tracking and analytics services, that monitor and collect end-users digital trail (see mind mapping in APPENDIX E.) on their web browser(s).
- Enable USEMP users to identify their web browser tracker(s) and thus, define fine-grained do not track (**DNT**) rules (in a flexible and intuitive manner).

[3] http://en.wikipedia.org/wiki/Plug-in_(computing)
[4] http://en.wikipedia.org/wiki/Application_programming_interface

2.1.2. USEMP-TOOLS: DataBait_OSN

DataBait_OSN is an OSN-enabled web/mobile web application (e.g., a Facebook application), which will function cross-browser and across user's device. DataBait_OSN will enable USEMP users to login to the OSN account and thus, provide USEMP System access to their OSN stored personal data. Specifically, DataBait_OSN:

- Will be developed as an OSN-enabled web application supporting both mobile and desktop browsers.
- Will efficiently manage the process of obtaining, storing and updating OSN API generated access tokens for the purpose of retrieving consumer digital train from OSN.

Key Expected Features

- Track USEMP users' personal and behavioural data generated and/or disseminated via the OSN account provider (e.g., Facebook). Such data can be either historical (i.e., data generated by the user prior creating a USEMP user account, stored in the OSN system) or real-time (i.e., data generated by the user upon creating a USEMP account) and should be fed into USEMP back-end system via USEMP API (**USEMP_API)**. The latter data will be collected via OSN's exposed APIs (OSN Graph API interspace[5]), denoted as **OSN_API**.
- To enable USEMP users to visually and graphically interact with DataBait_OSN in an intuitive and Quality of experience (QoE) optimised way
- To support mechanisms that minimise the imposed network traffic overhead (due to DataBait operation) in order not to affect the performance of the related devices/browsers.

2.1.3. USEMP-TOOLS: DataBait-GUI

DataBait_GUI will serve as USEMP graphical users' interface, enabling USEMP users to access, visualise, exploit and interact with USEMP services and features via their computers and/or mobile devices. Specifically, **DataBait_GUI** will be a developed as web/mobile-wed application, supporting various device and browser types (e.g., desktop pc, tablet and mobile device browsers) towards reassuring cross-platform, cross-device and cross-browsers compatibility and rendering optimization

2.1.3.1. Expected Features

DataBait_GUI will enable USEMP users to:

- *Manage their USEMP Account for example*
  - Log-in (via DataBait_AA Server Communication)
  - OSN Login (OSN Auth.)
  - User Registration
  - USEMP Privacy Setting Configuration
- *Visualisation of USEMP Services for example*:
  - User Profile Indicators
  - User Privacy Leaks
  - User Trackers Identification, filtering and Do Not Track Policies Creations

[5] http://en.wikipedia.org/wiki/Social_graph

- Personal Data Value Insights
- Audience / Influence Management
- USEMP Privacy Notifications (pop-up)
- Compare with OSN friends data

2.2. Backend Services

The USEMP Server Side (USEMP_SS) includes the backend services presented in Figure 1 (with a polygon shape annotated as USEMP_SS). A short list of the services are described below:

- Part of the backend services is to interface information between the client-side and backend
- Part of the backend services is to interface information between open/public Data-Sources like Wikipedia, logo DB, location DB etc. and the backend
- Part of the backend services is to process the client side information-provided within the backend system
- Part of the backend services is to cross-checking, Authentication and Authorisation of credentials collected provided by the USEMP_TOOLS
- Part of the backend services is to provide a USEMP_WORLD_VIEW to the OSN_USER via client-side USEMP_GUI

2.2.1. USEMP_API (client side and server side communication)

The communication of information-streams (bidirectional) between the USEMP Client-side (USEMP_TOOLS) and the USEMP Server-side (USEMP_SS) are instances of Application Programming Interfaces (called USEMP_API).

2.2.1.1. Expected Features

Instances of USEMP_API describe the communication interfaces in the USEMP_SYSTEM explained bellow:

- A USEMP_API instance enables the channel of datasets from PRIVACY_DB (Figure 1 HaP5) to the DataBait_GUI part of the USEMP_TOOLS.
- A USEMP_API instance enables the channel of datasets from DataBait_BROWSER to the USEMP_SS service ‘HISTORICAL_DATA_DB’ (Figure 1 HaP1).
- A USEMP_API instance is used when information datasets are channelled from DataBait_OSN to the USEMP_SS service ‘HISTORICAL_DATA_DB’ (Figure 1 HaP1).
- Instances of USEMP_API interfaces are utilised where USEMP_SYSTEM backend services (like some of the Technical Components) require information from open/public Data-Sources like Wikipedia, logo/images DB, location DB etc. these instances of the USEMP_API have direct access to the World Wide Web (Figure 1 HaP2).

2.2.2. USEMP_SS: 'HISTORICAL_DATA_DB'

The HISTORICAL_DATA_DB (Figure 1 HaP0) is a collection of data provided by the USEMP_TOOLS based on the OSN_USER credential (OSN accounts/profiles). The HISTORICAL_DATA_DB collected data is not only live-OSN interaction but also complete historical digital-trail (see mind mapping in APPENDIX E.) available by the OSN provider. The OSN_USER data (historical/old digital-trail) are collected and stored in order to be processed further by other USEMP_SS process.

2.2.3. USEMP_SS: DataBait_AA

The authorisation/authentication of credentials and information communication between the USEMP_TOOLS and the OSN_USER is necessary and very important to the system. All instances of authorisation/authentication in USEMP_SYSTEM are signified by DataBait_AA and assume all DataBait_AA instances interact with the 'Identity Manager' USEMP_SS service (Figure 1 HaP0).

2.2.3.1. Expected Features

DataBait_AA has multiple instantiations in the USEMP_SYSTEM and are explained bellow:

- The GUI instance of DataBait_AA exists when the USEMP_SYSTEM is installed and the OSN_USER creates a USEMP account. In this instance the DataBait_AA creates the credentials of the OSN_USER and conveys (using a USEMP_API) the information to the Identity Manager' (Figure 1 HaP0) backend service in the USEMP_SS.
- The GUI instance of DataBait_AA that enables the OSN_USER to access to USEMP_TOOLS. In this instance the Authorisation and Authentication of credentials (collected by DataBait_AA) utilise a USEMP_API interface to access and cross-check with the credentials that exist in the backend 'Identity Manager' service (Figure 1 HaP0). Once the credentials are cross-checked then access to USEMP_TOOLS by OSN_USER is approved.
- A transparent instance of DataBait_AA exists when data is acquired from the OSN on behalf of the OSN_USER to the backend services in the instances of DataBait_FB communicating with the 'HISTORICAL_DATA_DB' (Figure 1 HaP0).
- Additional instances of the DataBait_AA process are utilised when the DataBait_BROWSER is communicating data via the USEMP_API to the 'HISTORICAL_DATA_DB' (Figure 1 HaP0).
- A logging process performed by the OSN_USER using the USEMP credentials is also associated with the use of the DataBait_AA process for example in order to access the information on the 'PRIVACY DB' (Figure 1 HaP5) with OSN_USER requires to access and login to the DataBait_GUI.

2.2.4. USEMP_SS: Technical_Components

The collection of close and public data (Figure 1 HaP1 and HaP2) is utilised by the Technical Components to produce a dataset (Figure 1 HaP3) that will be evaluated by the Privacy/Profiling and Value Estimate processes. As the Technical components have different requirements in terms of: processing speed capability, inputs/outputs types of data (images, text, public-source data etc.), processing iterations of the component with different data-parameters there is a need for an Extract, Transform, and Load Orchestrator (ETL_Orchestrator). The ETL_Orchestrator is not present in the concept architecture for simplicity. In general terms the ETL_Orchestrator manages the Technical Components block input, output and workflow of data within the Technical Components block (Figure 1 between HaP2 and HaP3).

2.2.5. USEMP_SS: PRIVACY_DB

PRIVACY_DB is a database of information based on the post-processing content and metadata from the back-end processing elements. The back-end processing elements of USEMP_SYSTEM are the Technical Components (section 2.2.4), the 'Privacy Profiling'

(section 2.2.6) and 'Value Estimate' (section 2.2.7) processes. In addition to the backend functionality the PRIVACY_DB requires to be accessible to form the USEMP client-side (USEMP_TOOLS).

2.2.5.1. Expected Features

PRIVACY_DB is shown in Figure 1. USEMP Concept Architecture with the annotation HaP5 and the features are summarised below:

- Storage of information i.e. read and write privileges from the 'Privacy Profiling' (section 2.2.6) and 'Value Estimate' (section 2.2.7) processes as suggested by the arrow from Figure 1 step HaP4 to HaP5.
- Accessibility of the content of PRIVACY_DB from the DataBait_GUI is made by using the USEMP_API as suggested by the arrow from Figure 1 step HaP5 to the USEMP_TOOLS DataBait_GUI.
- The accessibility to PRIVACY_DB is web-based, as mentioned in section **Erreur ! Source du renvoi introuvable.**. PRIVACY_DB includes data that enables visualisations of DataBait_GUI described in the subsection section 2.1.3.1

2.2.6. USEMP_SS: PRIVACY PROFILING

The Privacy/Profiling process is one of the back-end services tightly coupled with the account information of OSN_USER (the main actor of USEMP identified in the top of Figure 1).

2.2.6.1. Expected Features

The expected features of this process are described in details in section 3.6.12 (and the main focus of WP6 D6.1) and summarised below:

- One of the expected features of privacy/profiling process is to detect/advise the OSN_USER provided information the known but abstract effects of profiling.
    - The information datasets are the HISTORICAL_DATA_DB, Technical components post processing information and repetition on demand of Technical components processes.
    - Advise on the detected profiling-factors through visualisation using the DataBait_GUI (i.e. after storing them in PRIVACY_DB by this process and reading the PRIVACY_DB by DataBait_GUI)

2.2.7. USEMP_SS: VALUE ESTIMATE

This process is located in the concept architecture in the same area as the Privacy/Profiling process because both of these processes have identical input datasets but different contribution datasets to the PRIVACY_DB (as indicated by the location in Figure 1 HaP4).

2.2.7.1. Expected Features

Value Estimate is a process where the personal data (also used in the Privacy/Profiling) can be analysed to depict an alternative insight to the Economic value of the data. Details of this process are available in section 3.6.13 and the implementation details are included in the task D6.2 of WP6. The expected features are summarised below:

- Access to datasets
    - The OSN_USER profile and associated data to the OSN_USER
    - HISTORICAL_DATA_DB

    - Technical components post processing information and repetition on demand of Technical components processes.
- Guidance on the detected Value is visualised using the DataBait_GUI (i.e. after storing them in PRIVACY_DB by this process and reading the PRIVACY_DB by DataBait_GUI)

3. Component Architecture

3.1. Client Side Tools

In this section, USEMP architecture client-side components are presented and analysed. Emphasis is placed on detailing USEMP Tools technical designing principles followed towards addressing the project's envisioned use case requirements, on justifying the selected technology stacks with respect to current state-of-the-art solutions as well as, on detailing the overall emerging USEMP client side architecture.

In the following, we refer to USEMP client-side architecture components as **USEMP Tools** since their main objective is to enable **USEMP users** to directly or indirectly interact with the overall **USEMP System (**often called **DataBait**). Therefore, many USEMP Tools' emerging design challenges are strongly correlated to end users' quality of experience (QoE) optimization, since they serve as the interface of USEMP System. Finally, with the term **"DataBait Plug-in(s)"** (denoted as **DataBait**), we refer to one or more of client-side USEMP Tools. The notion "plug-in" was deliberately selected since all USEMP Tool i.e., DataBait's, are software components that add a specific feature to an existing software application (e.g., a web browser), commonly denoted as plugins, extensions or add-ons[6].

USEMP Tools (DataBait(s)) will serve a twofold goal. On one hand, DataBait's will serve as USEMP graphical users' interface (denoted as **DataBait_GUI**), enabling them to access, visualise, exploit and interact with USEMP services and features via their computers and/or mobile devices. Specifically, **DataBait_GUI** *will be a web/mobile-wed application*, supporting various device and browser types (e.g., desktop pc, tablets and mobile devices) towards reassuring cross-platform and cross-device compatibility.

On the other hand, DataBait's will serve as end-users' personal/behavioural/OSN personal data tracking/collection mechanisms. The collected data via DataBait's will be then provided to USEMP back-end for further processing and analysis via **USEMP API** (application programming interface[7]). In order to support USEMP use cases, there are two types of DataBait's that will be developed for collecting end-users' data:

- ***DataBait_OSN*** *(e.g.,* ***DataBait_FB****) serving as end-users' Historical & Real-Time OSN Data Collectors (e.g., from Facebook)*

DataBait_OSN *will be an OSN-enabled web/mobile web application* (e.g., a FB application), which will function cross-browser and across user's device. DataBait_OSN will enable end-users to log-in to the OSN (e.g., FB log-in) account and thus, provide USEMP System access to their OSN Graph API personal data.

- ***DataBait_BROWSER*** *serving as end-users' Real-Time Behavioural/Personal Browser Data Collectors*

DataBait_BROWSER *will be a browser plug-in (or add-on)* and serve as users' digital trail (see mind mapping in APPENDIX E.) tracker and aggregator, specific to the browser utilized by the data-subject, towards collecting users' personal data generated or

[6] http://en.wikipedia.org/wiki/Plug-in_(computing)
[7] http://en.wikipedia.org/wiki/Application_programming_interface

distributed via their web browsers. In addition to the above data, the DataBait_BROWSER will also collect information related to the Internet services that track online users' browser behaviour (e.g., online analytics services, ad networks, etc.). The rest of this section is organised as follows:

First, the methodology followed towards designing and justifying USEMP Tools are detailed. Then, DataBait plug-ins' technical, functional and data/information model requirements are detailed and analysed. Finally, dedicated sections for the three DataBait's (DataBait_BROWSER, DataBait_OSN and DataBait_GUI) are provided towards detailing the followed solutions, justifying the selected technologies with respect to state-of-the-art (SoTA) and finally, relating DataBait's to the overall architecture.

3.2. Client Side Browser Plugin

3.2.1. Architecture Solutions Description and SoTA Analysis

As detailed in the previous section, **DataBait_BROWSER** key goals are:

- **DataBait_B_GoalA:** Track USEMP users' volunteered and behavioural data generated via their web browser(s) (either explicitly or implicitly) and feed the latter into USEMP back-end system via USEMP API, denoted as **USEMP_API**.
- **DataBait_B_GoalB:** Track USEMP users' trackers, in terms of 3rd party online tracking and analytics services, that monitor and collect end-users digital trail on their web browser(s).
- **DataBait_B_GoalC:** To enable USEMP users to identify their web browser tracker(s) and thus, define fine-grained do-not-track (**DNT**) rules (in a flexible and intuitive manner) that will be realised and reassured by DataBait_BROWSER.
- **DataBait_B_GoalD:** To enable USEMP users to control the operation of the tool, in terms of making them aware of every action performed by DataBait_BROWSER and how the latter affects their privacy.
- **DataBait_B_GoalE:** To enable USEMP users to authenticate/authorize via DataBait_BROWSER in to their USEMP account via **DataBait_AA** in a common and transparent way across browsers.
- **DataBait_B_GoalF:** To enable USEMP users to visually and graphically interact with DataBait_BROWSER in an intuitive and QoE optimised way for each of the above goals.

The latter goals reveal the strong correlation of DataBait_BROWSER's operation with USEMP end-users' browser(s). Thus, any potential technical solution should take into consideration such a peculiarity. To that end, four SoTA technical solutions have been investigated towards determining DataBait_BROWSER technology.

3.2.1.1. Enabling DataBait_BROWSER as a Browser Add-on

A browser extension, i.e., **a browser plug-in or add-on**, is a computer program that extends the functionality of a web browser in some way[8]. Depending on the browser type, similar terms such as plug-in or add-on are used. Browser extensions can be created through the use of web technologies such as **HTML, JavaScript, and CSS**. Browser extensions can also improve the user interface of the web browser without directly affecting viewable content of a web page. This improvement can be achieved through a variety of add-ons such as toolbars and plug-ins.

Browser extensions (add-ons) popularity and variety across all major browsers has rapidly increased during the years e.g., Google Chrome [9] Mozilla Firefox[10] and Apple Safari[11], due to the browsers-centric enhanced features that they can offer to the end users. From a technical point of view, there are also multiple methodologies towards creating a browser add-on, such as:

[8] http://en.wikipedia.org/wiki/Browser_extension
[9] Chrome Web Store. https://chrome.google.com/webstore/category/extensions?_sort=1
[10] Mozilla Firefox Add-ons https://addons.mozilla.org/en-US/firefox/extensions/?sort=users
[11] Safari Extensions. http://extensions.apple.com/

- **Add-ons SDK Extensions.** The Add-ons SDK (provided for web browser developers for all the most popular browsers) is a set of simple APIs used to quickly build a browser extension (e.g., Google Chrome extensions or user-scripts). It abstracts away most of the XUL / XPCOM infrastructures, giving a more familiar HTML and JS environment to work with.
- **Bootstrapped extensions.** Bootstrapped extensions don't require a restart to be installed, like SDK extensions, but they don't have the easy access to SDK APIs or sandboxing. Every feature and action is performed manually, like tracking windows to add or remove UI. However, there are various great development tools available via existing JavaScript Modules.
- **Firefox for Android extensions.** Extensions in Firefox for Android are slightly different. Since the UI is native instead of XUL, the way extensions can modify it is different and a bit more limited. This also means overlay extensions are not supported, so the potential options are the SDK and bootstrapped extensions.
- **Overlay Extensions.** Overlay extensions are the old way of making add-ons. It might still make sense to use this approach if someone needs to create a very complex overlay or have other specific needs. However, having to restart a browser to install this kind of extension is annoying because it breaks the user's workflow.

The following figure provides a high level overview of the creation of DataBait_BROWSER as a browser add-on.

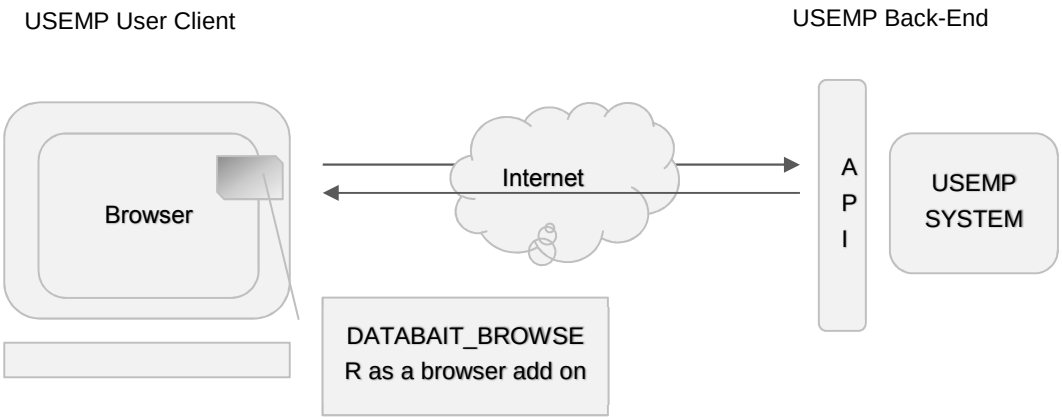


Figure 2. DataBait_BROWSER as a Browser Add-on

Enabling DataBait_BROWSER as a browser plug-in would require:

- The development and creation of a DataBait_BROWSER for each available popular browser. Let us underline that existing development frameworks enable the creation of cross-browser plug-ins.
- The publishing and distribution of the latter via corresponding browser stores (e.g., Chrome Store and Firefox Store);
- USEMP users should download and install DataBait_BROWSER into their browsers, sign-in with their USEMP account and then, DataBait_BROWSER will facilitate goals **DataBait_B_GoalA - DataBait_B_GoalF,** by directly collecting users' actions within their browsers**.**

3.2.1.2. Enabling DataBait_BROWSER as a Proxy (On-board or External)

A proxy is an **intermediate server** that sits between the client and the origin server[12]. In order to get content from the origin server, the client sends a request to the proxy naming the origin server as the target and the proxy then requests the content from the origin server and returns it to the client. The client must be specially configured to use the forward proxy to access other sites. Forward proxies are able to retrieve from a wide range of sources (in most cases anywhere on the Internet). The terms "forward proxy" and "forwarding proxy" are a general description of behaviour (forwarding traffic) and thus ambiguous.

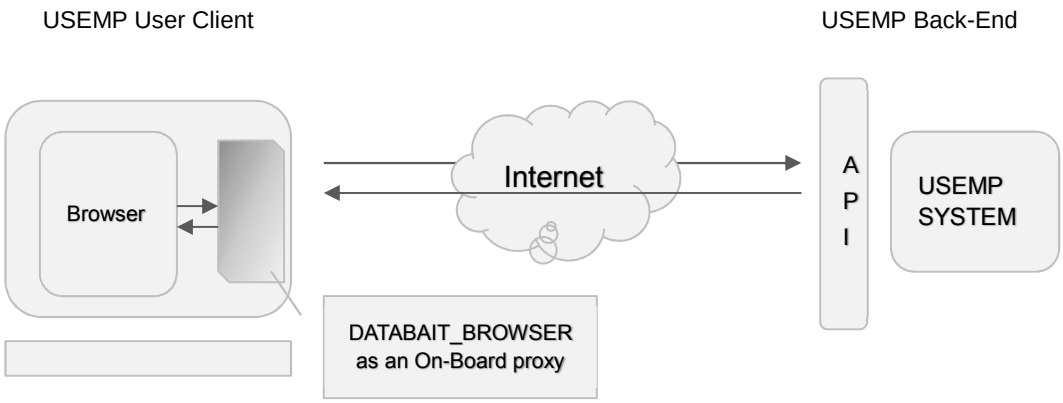


Figure 3. DataBait_BROWSER as an On Board Proxy

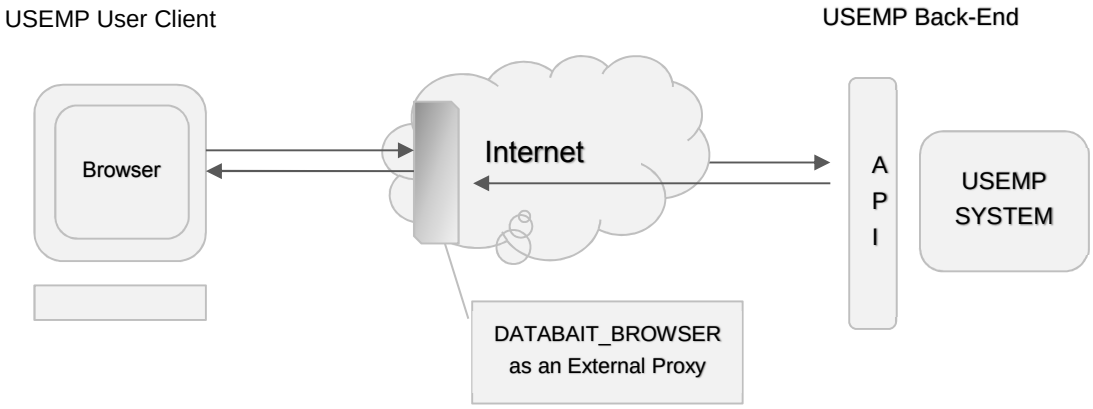


Figure 4. DataBait_BROWSER as an External Proxy

Enabling DataBait_BROWSER as proxy would require:

- The development and creation of a proxy server. A proxy server could be either **on board** i.e., operating within end-users device or **externa**l i.e., operating out-side users device.
- In both cases, the users should download and install the server and then, sign-in with their USEMP account.

[12] http://httpd.apache.org/docs/2.0/mod/mod_proxy.html#forwardreverse

- DataBait_BROWSER would then facilitate partially goals **DataBait_B_GoalA - DataBait_B_GoalF** by forwarding/proxy users' actions/network traffic at their browsers to USEMP back-end**.**

Concluding our analysis, let us interline that one of the main advantages of using an external proxy server compared to an on-board is the ability of the first to support **mobile browsers**. On the other hand, the process of installing an external proxy is much more complicated for end-user.

3.2.1.3. Enabling DataBait_BROWSER as a Hybrid Solution

A Hybrid DataBait_BROWSER solution would consist of both: a browser plug-in and an external proxy server. The reason behind such a solution is related to the complementary features supported by the above two solutions, since an external proxy is the only solution that fulfils USEMP technical requirements of supporting mobile browsers, while a plug-in supports all the USEMP features requirements apart from the latter.

3.2.2. DataBait_BROWSER Technical Solution Identification & Justification

Founded on the previous SoTA analysis and towards identifying the most appropriate technical solution for developing and creating DataBait_BROWSER, a corresponding competency matrix has been created. The completion of the latter emerged after investigating and testing the capabilities of each of the four identified solutions.

Evaluation Criteria	**Browser Add-On**	**Proxy (on board)**	**Proxy (external)**	**Hybrid Solution (Add-on & External Proxy)**
	Technical Requirements (Table 2 USEMP Tools Technical Requirements)			
Web Browser Compatibility	Chrome, Firefox, Safari	All	All	All
Mobile Browser Compatibility	No	No	Yes	Yes
Deployment / Installation	Yes (user friendly)	Yes (user friendly)	Yes (not user friendly)	Yes (not user friendly)
Tracking/Operation Traffic Management Support	Yes	Yes	Yes	Yes
	Feature Requirements (Table 1 USEMP Tools Functionality Requirements)			
Supporting USEMP Features Required (DataBait_B_GoalA–F)	Yes	No	No	Yes
	Tracking Requirements (APPENDIX D. , Table 41: User Real-time Personal and Behavioural Browsers Data and Table 42: Internet Services that Track user's Data)			
Supporting USEMP Data Tracking Requirements	Yes	No	No	Yes
	Platform Integration, Expandability and Scalability			
USEMP System Architecture Compatibility	Yes	Partially	Yes	Yes
API Compatibility/Expandability	Yes	Yes	Yes	Yes
Open Source Solutions	Yes	Yes	Yes	Yes

Table 3 USEMP DataBait_BROWSER Technical Solution Competency Matrix

Table 3 present in a concrete way the outgrowth of the conducted investigation. Let us underline that apart from the support of USEMP technical, features and tracking requirements; three additional characteristics have been examined related to a) solution's compatibility/integration with USEMP architecture, b) solution's API compatibility and expandability and finally, c) the existence of open source solutions that partially cover parts of the envisioned functionality. The latter characteristic is strongly related to overall USEMP ambition of creating an open source solution towards not only favouring a developers community-based supported and evolved solution but also, favouring the vast distribution, dissemination and use of the envisioned innovative solutions by end-users.

A close study of Table 3 reveals that the most suitable technical solution for enabling DataBait_BROWSER is a as a browser extension/add-on, since it is the only solution that support all tracking and features requirements compared to a proxy-based solution. This is mainly related to browser's add-on ability on accessing user generated data at the browser (point of creation), while the proxy is accessing them at device network traffic level (after their creation). On the other hand, browser add-on solution lacks mobile browser support. This is due to the fact that existing mobile operating systems and corresponding mobile browsers are not yet mature to support the extended use of add-ons. Thus, the only solution that could support both mobile and web browsers is the hybrid one.

Conclusion: Towards a) adopting an incremental development process, b) targeting to an efficient end-to-end prototype solution and c) given that a browser add-on solution can be easily extended either to a hybrid one or to a mobile browser add-on, ***DataBait_BROWSER will be implemented as a browser add-on for the most popular web browsers (for Microsoft and MAC OS users) i.e., Chrome, Firefox and for MAC Safari.***

3.2.3. Selected Open Source Foundations

Prior to the analysis of the overall emerging architecture and the interaction points of DataBait_BROWSER Tool with USEMP System (back-end services), in this section a short overview of two existing open source browser add-on solutions are presented that fulfil a small subset of the envisioned DataBait_BROWSER features and functionalities but could serve as the foundations of USEMP Tool development.

- **Lightbeam (for Firefox)**
  - **Developer:** Mozilla Firefox
  - **Licence:** Mozilla Public License Version 2.0
  - **Browser Version Supported:** Firefox 18 or higher. [13]
  - **Link:** Lightbeam [14]
  - **Key Operations:**
    - **OP1.** Create a record of events for every site you visit and every third party site that is stored locally on your browser.
    - **OP2.** Visually graphs these events to highlight the interactions between sites third parties.
    - **OP3.** Privacy Control
- **Chrollusion (for Chrome and Safari)**

[13] http://www.mozilla.com/en-US/firefox/fx/
[14] https://github.com/mozilla/lightbeam

- **Developer:** Mozilla Firefox
- **Licence:** GNU (GENERAL PUBLIC LICENSE)
- **Browser Version Supported:** Latest Chrome and Safari versions
- **Link:** Chrollusion[15]
- **Key Operations:**
  - **OP1.** Create a record of events for every site you visit and every third party site that is stored locally on your browser.
  - **OP2.** Visually graphs these events to highlight the interactions between sites third parties.
  - **OP3.** Block the otherwise invisible websites tracking a user's browser.

The following table summarizes the DataBait_BROWSER features supported by the above open source projects.

G. #	Operation Description	OPEN Source Solution Capabilities
G.A.	Tracking USEMP Users Web Behavioural/Personal Data	**Not Supported** Custom Tracker Creation and Integrated within the Plug-in is required. **Integration:** Sending raw data to the back-end via custom Flume-based API.
G.B.	Tracking Users' Trackers/Brands	**Partially Supported** Plug-in should be properly extended to enable DataBait USTs and features. **Integration:** Apache Flume (back end) moving streaming data into the Hadoop Distributed File System (HDFS).
G.C.	Client Side Privacy Control and Audience Management	**Partially Supported** Plug-in should be properly extended to enable DataBait USTs and features.
G.D.	Client Side Visualization	**Not Supported** Custom client GUI towards representing user's internet trackers (and their activities) via enhanced visualisations can be supported via plug-ins feeds.
G.E.	USEMP User Authorisation Authentication	**Not Supported** Custom solution should be implemented.
G.F.	Visual & Graphical Interactions	**Not Supported** Custom client GUI towards representing user's internet trackers (and their activities) via enhanced visualisations can be supported via plug-ins feeds.

Table 4. USEMP DataBait_BROWSER Open Source Solution Expected Features Coverage

[15] https://github.com/disconnectme/chrollusion

3.2.4. Overall Architecture and USEMP_API Integration

Concluding this sections analysis; the following figure provides a high level overview of the emerging USEMP Tools client-side architecture a detailed analysis of the latter is provided in the next chapter.

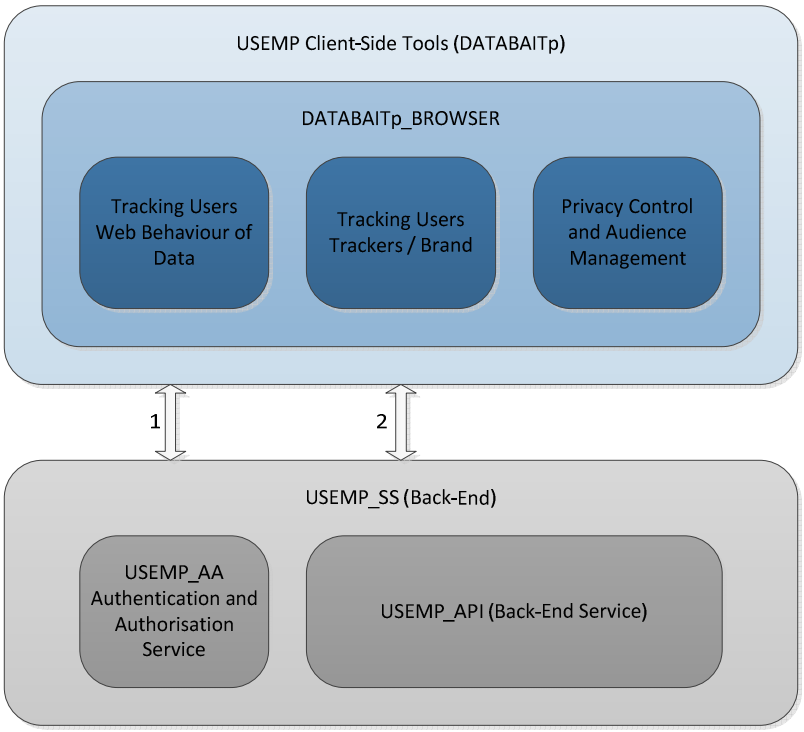


Figure 5. USEMP Client Tools Overall Architecture and API Anchor Points

DataBait_BROWSER browser add-on, upon installation in a USEMP user browser, will communicate and interact with USEMP architecture via two anchor points.

- **DataBait_AA**, is USEMP authentication and authorization service towards a) enabling USEMP users to register or log-in with their USEMP account as well as, b) retrieving appropriate authorization token(s) that will allow the add-on to communicate via USEMP_API with USEMP_SS back-end.
- **USEMP_API**, will enable the add-on to a) feed the users' browsers generated personal and behavioural data to USEMP System, as well as b) to retrieve the outcome of USEMP Privacy and Value components towards serving as the visual and graphical interface of USEMP services. The latter API will be implemented as a custom Apache Flumes Channel. Apache Flumes[16] is a distributed, reliable, and open source service for efficiently collecting, aggregating, and moving large amounts of streaming event data.

[16] http://flume.apache.org/

3.3. Social Network Integration

DataBait_OSN is not only one of the three USEMP client side Tools, but also one of the key enablers of overall envisioned USEMP System, since it serves as the interface of the latter with USEMP users' Online Social Networks (OSNs). Specifically, the key goals of DataBait_OSN are detailed in the following:

- **DataBait_O_GoalA:** Track USEMP users' personal and behavioural data generated and/or disseminated by them via their OSN (e.g., Facebook). Such data can be either historical (i.e., data generated by the user prior creating a USEMP user account, stored in the OSN system) or real-time (i.e., data generated by the user upon creating a USEMP account) and should be fed into USEMP back-end system via USEMP API (**USEMP_API)**. The latter data will be collected via OSN's exposed APIs (OSN Graph API interspace[17]), denoted as **OSN_API**.
- **DataBait_O_GoalB:** To enable USEMP users to visually and graphically interact with DataBait_OSN in an intuitive and QoE optimised way.
- **DataBait_O_GoalC:** To support mechanisms that minimize the imposed network traffic overhead (due to DataBait operation) in order not to affect the performance of the related devices/browsers.
- **DataBait_O_GoalD:** To enable USEMP users to control the operation of the tool in terms of making them aware of every action performed by DataBait_OSN and how the latter affects their privacy.
- **DataBait_O_GoalE:** To enable USEMP users authenticate/authorise via DataBait_OSN in to their USEMP account via **DataBait_AA** in a common and transparent way across browsers.
- **DataBait_O_GoalF:** To enable proper throttling mechanisms in order to reassure their efficient operation, given OSN API calls limitations.

Prior to the analysis of available technical and architectural solutions for enabling DataBait_OSN towards fulfilling the above goals, an in depth SoTA analysis on modern popular OSN networks architectures is provided placing emphasis a) on OSN Graphs API attributes (i.e., the mechanism OSNs store, process and exploit end users personal data; b) OSN API (i.e., the mechanism OSNs make the latter data publicly online available); as well as c) OSN limitations (i.e., imposed limitations by OSNs in the use of their APIs towards controlling their architecture load as well as the availability of their service).

3.3.1. Online Social Networks OSN Architectures (Graph API analysis)

Todays' OSNs are cloud-based services, featuring a broad scale of functionalities (from location based services and photo sharing, to social networking and interactions) that have revolutionised the way we communicate, connect, share and eventually conduct business. An emerging trend to expose OSNs functionalities through publicly available APIs (Application Programming Interfaces) has not only redefined how software and services are delivered, but also indicates how business value is moving towards a thriving high-paced mobile application ecosystem. The main OSN service design approaches are:

[17] http://en.wikipedia.org/wiki/Social_graph

- **Web Services (WS-*)**. According to W3C (2004a, 2004b), a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards i.e.,
  - REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and
  - Arbitrary Web services, in which the service may expose an arbitrary set of operations.
- **RESTful Web Services** that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. A concrete implementation of a REST Web service follows four basic design principles (IBM, 2008):
  - Use HTTP methods explicitly (i.e. To create a resource on the server, use POST; To retrieve a resource, use GET; To change the state of a resource or to update it, use PUT; To remove or delete a resource, use DELETE).
  - Be stateless.
  - Expose directory structure-like URIs.
  - Transfer XML, JavaScript Object Notation (JSON), or both.
- **JavaScript APIs** built on JavaScript (JS), an open source client-side scripting language, and allowing access via HTTP (through a browser or a server HTTP connection) by making a call to a script on another server. They are typically client-side script APIs, for use in browsers and similar user agents.

Due to the constantly increased popularity and use of OSNs, a paradigm shift in their service design towards REST is noticed, indicated by its adoption by mainstream Web 2.0 service providers (including Google+ and Facebook, Twitter— who have deprecated or passed on SOAP and WSDL-based web services in favour of an easier-to-use, resource-oriented model to expose their services). Towards a more in depth analysis, in the following table the main design characteristics of three OSN services are provided. The platforms qualified to be analysed, i.e. Facebook, Twitter and LinkedIn, were expected to be selected because of their popularity.

Let us further underline the following observation relating an OSN's architecture and the envisioned operation of DataBait_OSN:

- **Note 1st.** An OSN is Cloud-based Service, collecting, storing and processing users' personal data via OSN Graph API.
- **Note 2nd.** OSN user's personal data are accessible only via RESTful APIs, where JSON is the most commonly used data format.
- **Note 3rd.** The use of OSNs API is limited by strict API calls/duration threshold.

Typically, a Graph API consists of objects, aggregations and connections, while objects can be connected with other secondary objects. Additionally, every object and aggregation is uniquely addressable through an id existing on the main path of an API.

Facebook paved the way towards an API design based on objects with its Graph API[18]; a unique index and a simplified dictionary for recurring, connected methods, while others, like RunKeeper, followed up with a similar approach.

Service Name	Facebook	Twitter	LinkedIn
API Name	Facebook Graph API v2.0	REST API version 1.1	LinkedIn API
Description	The Graph API is the primary way to get data in and out of Facebook's social graph. It's a low-level HTTP-based API that you can use to query data, post new stories, create check-ins or any of the other tasks that an app might need to do.	The most recent version of the Twitter REST API.	The REST API provides a simple, consistent representation of people, companies, jobs, and the interactions and relationships between them.
Documentation	Facebook Graph-API[19]	Twitter API [20]	LinkedIn API [21]
Protocol	REST/JavaScript	REST	REST/JavaScript
Data Format	JSON	JSON	JSON
Available Libraries	JavaScript/PHP/Python	JavaScript/Java/PHP/Python	JavaScript/Java/PHP/Python
Call Limitation	Not specified (approx. 50 requests/ 30 min)	350 requests/hour	100k requests/day
Authentication Method	OAuth 2.0	OAuth 2.0	OAuth 1.0

Table 5 Popular OSN Designing Principles

[18] https://developers.facebook.com/docs/graph-api
[19] https://developers.facebook.com/docs/graph-api
[20] https://dev.twitter.com/docs/api/1.1
[21] https://developer.linkedin.com/apis

3.3.2. DataBait_OSN Technical Options Identification & Solution Justification

In this section, DataBait_OSN technical solution and corresponding architecture is analysed and justified, compared to current SoTA commonly adopted solutions. Prior to our analysis let us underline that in the rest of this section **Facebook (FB) terminology has been adopted (as detailed in FB Graph API v2.0[22])**. The reasoning is threefold. First, FB Graph API is the most advanced, extended and widely used OSN API in recent years. Second, Facebook is the prime OSN that will be investigated and exploited in USEMP. Finally, due to the common use and abstraction of FB Graph API, a DataBait_OSN that is developed for realising goals DataBait_OSN_A - DataBait_OSN_G in the case of Facebook (i.e., a **DataBait_FB**), can be easily extended/adopted to support a large variety OSNs (e.g., Twitter, Runkeeper, LinkedIn, etc.) that follow the same or similar Graph API principles.

The prime way for apps and services to access (read and write) OSN users' personal volunteered and/or behavioural data is via OSN's social graph, namely OSN Graph API. Thus, in order DataBait_OSN to achieve its goals, it should be designed either as an OSN-enabled native mobile application (one per OS e.g., Android, iOS, WP8) or as an OSN-enabled web/mobile web application. Therefore, in order to support both mobile and desktop users, and to avoid the development of multiple native applications per OS:

> ***Designing Principle I.*** *DATABAIT_OSN will be developed as an OSN-enabled web application supporting both mobile and desktop browsers.*

In order for an OSN-enabled web application to access users' Graph API personal data, the following steps need to be followed towards enabling end users' to log-in via their OSN account. At a high level those steps are (as defined in the link[23]):

- Checking user's login status to see if they are already logged into the app.
- If they are not logged in, to invoke the login dialog and ask for a set of data permissions. This step is very important, since it determines the way an OSN allows its users to control the privacy of their personal data by explicitly determining which of the latter can be accessed by an application.
- Verify user's identity.
- Store the resulting access token provided by the OSN.
- Make API calls.
- Logout.

The above process reveals that in order an OSN application (i.e., DataBait_OSN) to access users' personal data, when USEMP users' connect to the latter via OSN Login, the app should be able to obtain an access token which provides temporary, secure access to OSN Graph APIs.

[22] https://developers.facebook.com/docs/graph-api/using-graph-api/v2.0
[23] https://developers.facebook.com/docs/facebook-login/login-flow-for-web/v2.0

Designing Principle II. *DATABAIT_OSN should efficiently manage the process of obtaining, storing and updating OSN API generated access tokens.*

An **access token** is an opaque string that identifies a **user, app, or page** and can be used by the DataBait_OSN web application to make Graph API calls. Access tokens are obtained via a number of methods and corresponding architecture design, as detailed in the following. The token includes information about when the token will expire and which app generated the token. There are different types of access tokens to support different use cases (as defined in Facebook documentation[24]):

- **User Access Token** – The user token is the most commonly used type of token. This kind of access token is needed any time the app calls an API to read, modify or write a specific person's data on their behalf. User access tokens are generally obtained via a login dialog and require a person to permit an app to obtain one[25].
- **App Access Token** – This kind of access token is needed to modify and read the app settings. It is generated using a pre-agreed secret between the app and OSN and is then used during calls that change app-wide settings. An app obtains an app access token via a server-to-server call[26].
- **Page Access Token** – These access tokens are similar to user access tokens, except that they provide permission to APIs that read, write or modify the data belonging to an OSN Page.
- **Client Token** - The client token is an identifier that you can embed into native mobile binaries or desktop apps to identify an app. The client token isn't meant to be a secret identifier because it's embedded in applications.

Designing Principle III. *In order DATABAIT_OSN web application to properly operate, it should be subscribed as an OSN-enabled application towards obtaining an app access token*

Designing Principle VI. *In order DATABAIT_OSN web application to properly operate, it should allow USEMP users to Login via their OSN-related account credentials towards obtaining* ***users' access token.***

In line with the previous analysis, the way DataBait_OSN obtains/manages users' access tokens plays a crucial role in the overall operation. To that end, let us underline the fact that **OSN access tokens are portable**. This means that once an OSN application

[24] https://developers.facebook.com/docs/facebook-login/access-tokens
[25] https://developers.facebook.com/docs/facebook-login/access-tokens#usertokens
[26] https://developers.facebook.com/docs/facebook-login/access-tokens#apptokens

obtains a token, the latter can be generally used from any machine - server, client or otherwise. In our case, this means that once DataBait_OSN gets an OSN user access token, then the latter can be used by USEMP System back-end in order to establish a **server-to-server** communication and retrieve a user's personal data.

Moreover, combining web interfaces and servers provides a mix of different possible configurations that can be realised. However, different configurations come with different trade-offs in terms of capabilities and security. In the rest of this section, the most popular configurations/architectures are analysed, while their pros and cons are justified towards determining the most suitable for enabling DataBait_OSN.

3.3.2.1. Architecture Option A.

"Login happens in a Web Client & API requests happen in a Web Client"

Advantages:

- Simple to implement.
- Authentication is not required often with long-term token.

Disadvantages:

- API Communication Load affect client device (since all the data are retrieved by client device/application (i.e., DataBait_OSN) and then forwarding to the served (USEMP_SS).
- No offline posting and communication is enabled.

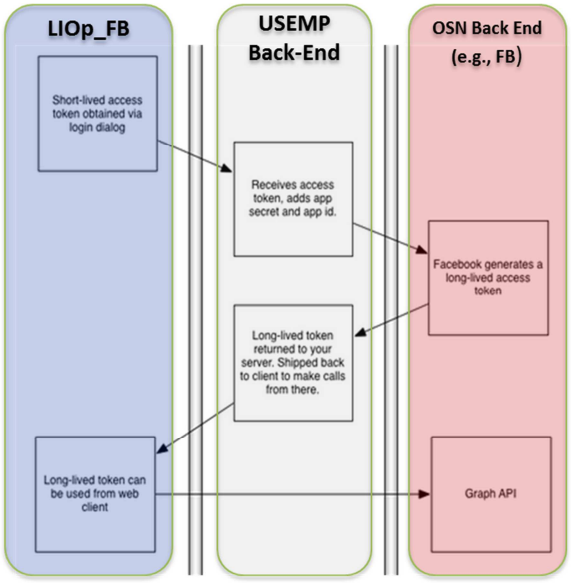


Figure 6. DataBait_OSN Architecture Option A: API Requests Enabled at the Web Client

3.3.2.2. Architecture Option B.

"Login happens in a Web Client & API requests happen on a Server"

Advantages:

- API Communication Load is not affecting client device (since all the data are directly retrieved by the served (USEMP_SS).
- Offline posting and communication is enabled.
- Advanced security features are available with server-based calls.

Disadvantages:

- Harder to be implement.
- Client should proxy access token calls.

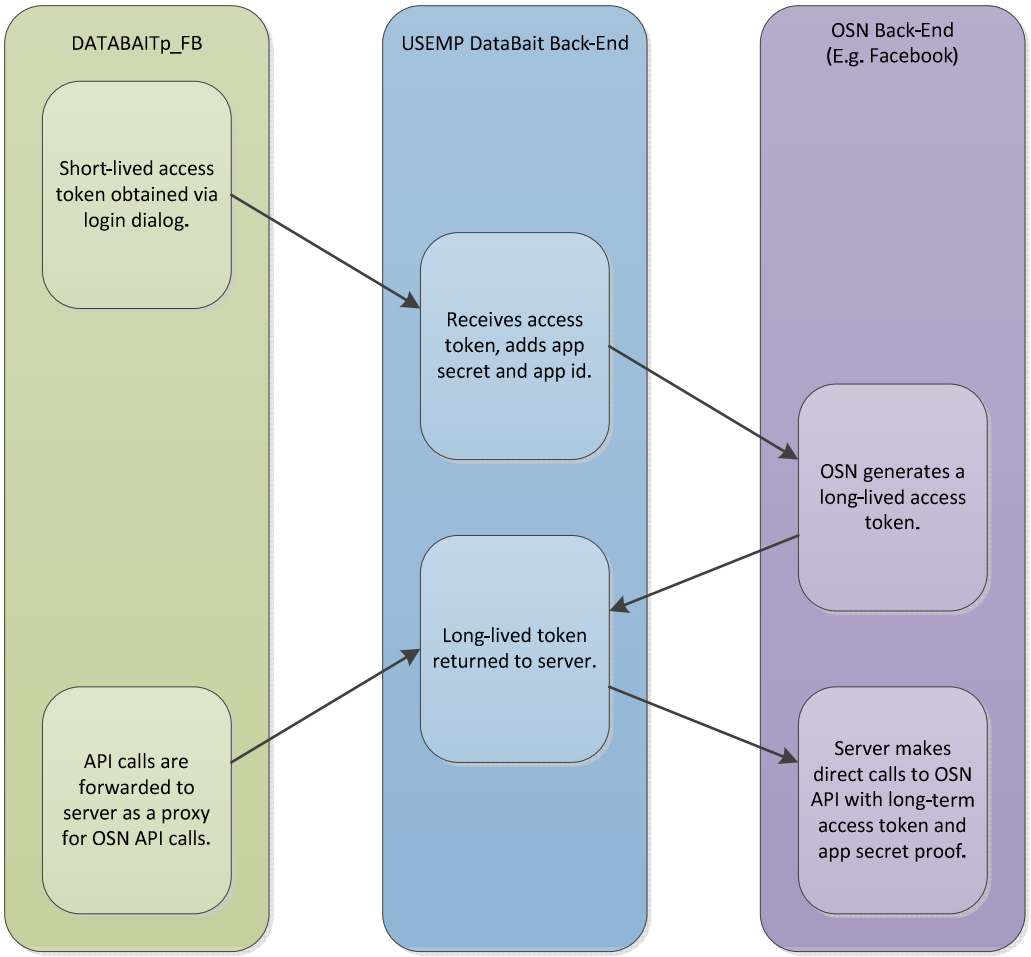


Figure 7. DataBait_OSN Architecture Option B: API Requests Enabled at the Server

Moreover, due to the following requirements:

- DataBait_OSN should be able to obtain large volumes of USEMP users' OSN personal data to support various USEMP System Algorithms;
- DataBait_OSN should be able to obtain historical data (in an offline manner in order not to affect users QoE);
- DataBait_OSN should be able to support offline data retrieval, since USEMP system algorithms will constantly process USEMP user data;

Architecture Option B is the most suitable for DataBait_OSN, thus:

***Designing Principle V**. DATABAIT_OSN will be implemented as an OSN-enabled web/mobile web application where users' login and access tokens management is ambled in a Web Client, while API requests happen on a USEMP System (via a proper Apache Flumes Channel).*

3.3.3. DataBait_OSN and OSN API Limitations

The previously defined five DataBait_OSN design principles determine the technology that will be used and the corresponding architecture that will be enabled. In this section, additional principles are determined, defining additional sub-components/algorithms/features that should be realised towards addressing various drawbacks that emerge due to OSNs API limitations.

- **OSN Graph API Rate Limiting.** As defined in FB documentation[27], there are 3 types of throttling for FB Graph API calls, but similar principles and limitations hold for all OSN cases (as shown in Table 5). Specifically, each API call goes through these different levels of throttling during its lifetime. What is more, when an API call passes the top level throttling, it is subject to be throttled at the second level and so on.
- There are three types of throttling for an OSN Graph API Calls:
    - User Level Rate Limiting (affecting all the API calls for the same USEMP user)
    - App Level Rate Limiting (affecting all the API calls for the same USEMP user via all her/his OSN apps);
    - Specific API Level Rate Limiting (affecting all the API calls for ALL USEMP users);

Towards avoiding the latter, especially when considering large volumes of USEMP users,

> ***Designing Principle VI.*** *DATABAIT_OSN should be enforced with sophisticated throttling algorithms that will respect OSNs API Limitations and thus, prevent any OSN denial of service effects.*

- **Real-Time User Object Retrieval.** As a consequence of the above limitation, special attention should be placed in the design and support of real-time OSN users' personal data retrieval. Specifically, instead of continuously searching OSN users' Graph API for new created objects OSNs Real-time User Objects Alterations Notifications[28] feature will be incorporated in DataBait_OSN. Thus, every time a users' object is created, then OSN API will communicate this change to the DataBait_OSN. The latter will be then filtered and based on sophisticated priority rules will be forwarded to the USEMP_SS for being further processed.

> ***Designing Principle VII.*** *DATABAIT_OSN should support sophisticated real-time user object alterations notifications' management mechanisms.*

[27] https://developers.facebook.com/docs/reference/ads-api/api-rate-limiting/
[28] https://developers.facebook.com/docs/graph-api/real-time-updates/v2.0

- **Multi-Lingual OSN Data.** Targeting the creation of USEMP Services that can support multiple geographies and user languages, DataBait_OSN should be able to feed USEMP Systems (and corresponding algorithm) with uniform data that could be further processed. To that end, DataBait_OSN should be integrated with OSN's Localisation Localization & Translation[29] features towards translating OSN user's data prior to their processing by the back end USEMP System.

***Designing Principle VIII**. DATABAIT_OSN should support sophisticated user data localization mechanisms.*

3.3.4. Overall Architecture and USEMP_API Integration

The following table justifies how the previously defined DataBait_OSN design principles, as well as the emerging technical and architecture solution, can reassure the fulfilment of the under consideration USEMP Tool goals (as defined at the beginning of this section).

G. #	DataBait_OSN Feature	DataBait_OSN Designing Principle
G.A.	Tracking USEMP Users OSN Behavioural/Personal Data (Historical and Real Time)	DP I, DP II, DP IV
G.B.	Client Side Visualisation	DP I, DP I
G.C.	Minimise Imposed Traffic Overhead	DP I, DP III, DP IV, DPV
G.D.	Explicit Privacy Control	DP II, DP III, DP IV
G.E.	USEMP User Authorization Authentication	DP II, DP IV
G.F.	OSN Limitations Support	DP VI, DP VII, DPVIII

Table 6 DataBait_OSN Designing Principles mapped to Tool's Goals

Concluding this sections analysis, the following figure provides a high level overview of the emerging (so far) USEMP Tools client-side architecture, a technical detailed analysis of the latter is provided in the next chapter.

[29] https://developers.facebook.com/docs/internationalization

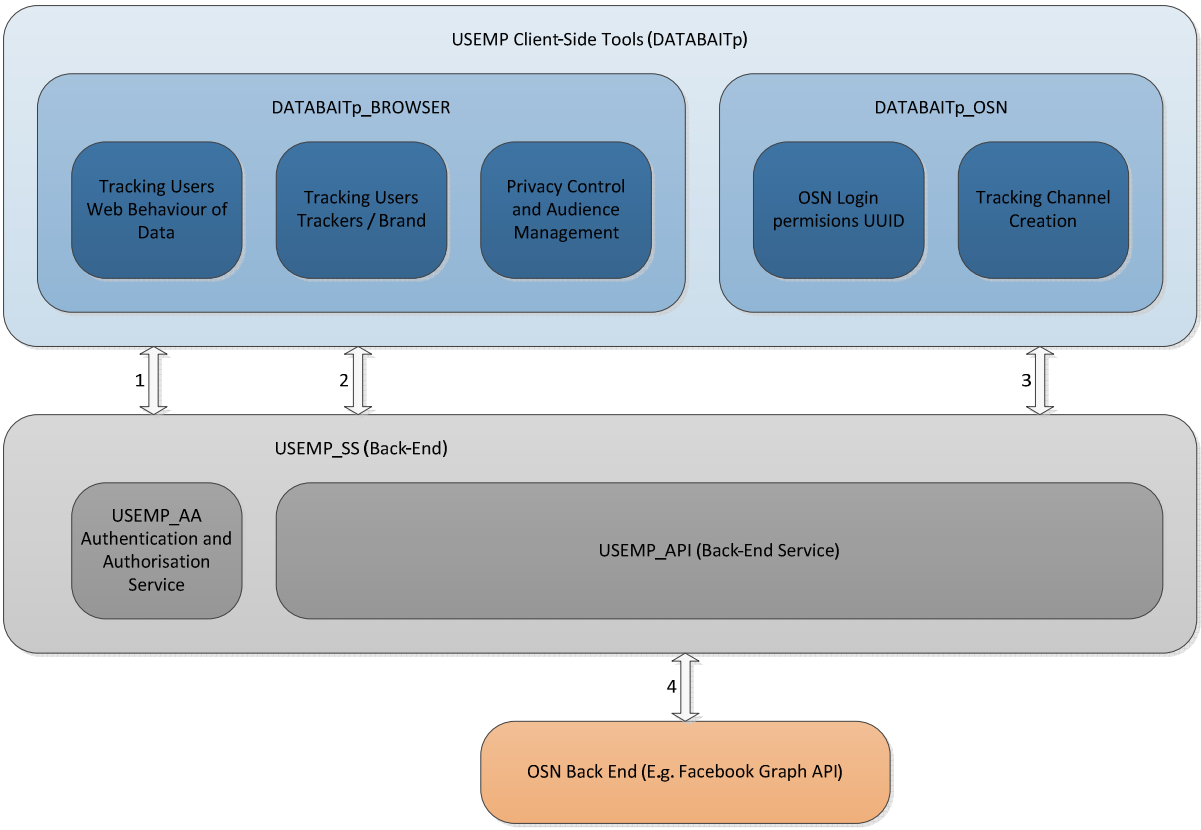


Figure 8. USEMP Client Tools Overall Architecture and API Anchor Points

DataBait_OSN web/mobile web application, upon USEMP user Login, will communicate and interact with USEMP architecture via two anchor points (interface is identified in Figure 8 arrow 1).

- **DataBait_AA,** is USEMP authentication and authorisation service towards a) enabling USEMP user to register or log-in with their USEMP account (the interface is identified in Figure 8 arrow 2).
- **USEMP_API,** will enable a) the client web app to manage OSN access tokens and establish a stable USEMP System - OSN Back-end communication and b) feed users' OSN generated personal and behavioural data to USEMP System via a Flumes enabled server-to-server communication interface. The interface identified in Figure 8 arrow 3 is USEMP_API instance (a) and arrow 4 is the USEMP_API instance (b) characterised as server-to-server.

3.4. GUI Components

DataBait_GUI, the third USEMP Tool analysed in this section, will serve as USEMP graphical users' interface, enabling USEMP users to access, visualise, exploit and interact with USEMP services and features via their computers and/or mobile devices. Specifically, **DataBait_GUI** *will be a developed as web/mobile-wed application*, supporting various device and browser types (e.g., desktop pc, tablet and mobile device browsers) towards reassuring cross-platform, cross-device and cross-browsers compatibility and rendering optimisation.

The following table summarises DataBait_GUI goals and expected features, collected via USEMP WP2 (use cases and UST), WP4 (Legislations) and WP6 (Privacy and Value Design Interface) requirements and specifications.

G. #	DataBait_GUI Requirement	Requirement Features Analysis	WP Requirement Definition
G.A.	DataBait Users Account Management	USEMP Log-in (via DataBait_AA Server Communication) USEMP OSN Log-in (OSN Auth.) USEMP User Registration	WP2 and WP7
G.B.	DataBait User GUI Visualization	USEMP User Profile USEMP User Privacy USEMP User Trackers USEMP Value Insights USEMP Audience / Influence USEMP Privacy Notifications (pop-up)	WP2, WP4 and WP6
G.C.	Explicit Privacy Control	Tracking Services/Activities Control Audience Management Settings Personal Data Protection Setting Notifications Setting Term of Use USEMP Service Deactivation	WP2, WP4 and WP7
G.D.	Technical Requirements	Cross Web Browser Compatibility Cross Mobile Browser Compatibility No Deployment / Installation Required	WP2 and WP7

Table 7 DataBait_GUI Features and Goals

3.4.1. DataBait_GUI Web Application Development

The objective of this section is to describe a number of state-of-the-art development methodologies and guidelines for the creation of **mobile rich media content web application, such as DataBait_GUI.** First, the selection of building DataBait_GUI as web/mobile web app and not as native application is justified and then, different designing approaches on delivering the mobile content of an application are analysed.

3.4.2. On Developing DataBait_GUI as a Web/Mobile Web App - *Smartphones & Tablets Support Challenges*

Smartphones and tablets are becoming increasingly popular as part of consumers digital lifestyle in European countries. These devices are used for consuming entertainment and increasingly learning content and in some cases are also integrated as part of the official curriculum. The most popular vendors of operating systems (and in some cases devices themselves) are Apple with its iOS operating system and Google with its Android operating system. Content and applications for these devices are primarily developed in three formats:

- **Web applications:** these are rich media mobile sites that can also operate without network connectivity after they have been accessed. These are primarily developed in HTML5 and JavaScript taking advantage of the increasing support from tablet and smartphone vendors.
- **Native applications:** these are native applications, which are distributed by the device vendors' purpose made application devices that can be accessed from the devices. Each vendor has usually its own application development framework. Examples are: Objective C and iOS SDK for Apple iOS devices and Java and Android SDK for Google Android devices
- **Native media formats:** these are media formats for content that can be published and controlled in vendors' content stores. For example Apple has developed the iBook format for its Book storefront to iOS devices and Amazon has developed KF8 format for its own Kindle store devices.

DATABAIT_GUI Design I. *Aiming at the creation of a DATABAIT_GUI that works on all desktop and mobile devices the option of a web application format has been selected. The main technologies that will be used are HTML5, CSS and JavaScript.*

The main web application development advantages are summarised below:

- Multi-platform – the developed web app code can run in multiple platforms and browsers;
- Instant iteration - no delays in the publication of new features and content (since there is no approval process with the application update);
- No approval process - no limitations placed on content or subject matter;

3.4.3. On DataBait_GUI Responsive Design - Cross Platform Mobile Web Design for Smartphones/Tablets

The main design and development challenges in web/mobile web applications (for tablets and smartphones) are related with the diversity of capabilities and content rendering approaches in various devices OS and mobile web browsers. Specifically, a mobile web application should:

- render without defaults in various device screen sizes should be supported;
- handle/support different media types for various mobile devices;
- support techniques for optimising user experience that vary based on device OS type and browser version;
- support integration with 3rd party applications;

In the rest of this section, two SoTA common approaches are used to minimise the complexity and increase the efficiencies in developing rich media web applications for smartphones and devices are analysed.

3.4.3.1. Fluid Web/Mobile Web App Design

In a fluid website layout, also referred to as a liquid layout, the majority of the components inside the app have percentage widths and thus, adjust to the user's screen resolution.

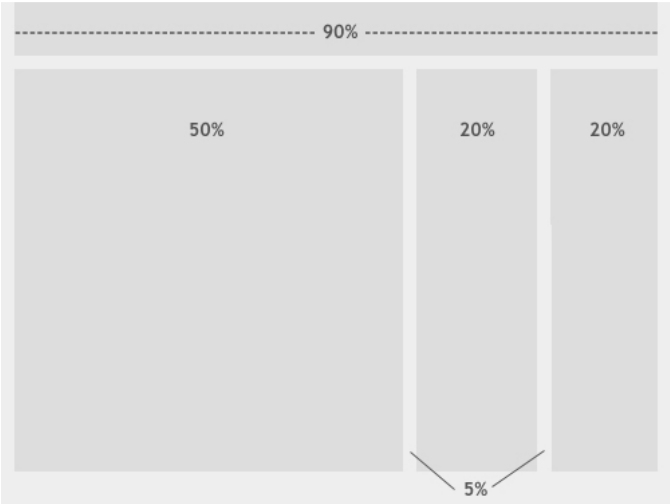


Figure 9. Example of Fluid Design

The Figure above depicts a fluid (liquid) website layout. While some designers may give set widths to certain elements in fluid layouts, such as margins, the layout in general uses percentage widths so that the view is adjusted for each user. The fluid web app design better fits the cases which a mobile-only delivery targeted approach is being utilised

- **Advantages**
  - the fluid design can be easily implemented and understood by rich media authors, since it has one design that fits all screen resolution factors;
  - the application content will render without defaults to both smartphones and tablets;
- **Disadvantages**
  - Cannot take into advantage the additional screen real estate which is available to tablets and larger devices (usually the design is based on the smallest screen to be supported), in most cases a different fluid design should be devised for larger factors devices (like tablets).

3.4.3.2. Responsive Web/Mobile Web App Design

Responsive design takes the approach of progressively defining additional elements that can be added as the resolution/width of the screen device increases. The responsive design approach is based on a number of significant resolution screen sizes that are used commonly by devices. In the Figure below a plot of the most important resolutions for screens from desktops (PC) to tablets and smartphones (resolutions below 240x320 are really reserved for mobile phones which are not smartphones) is depicted.

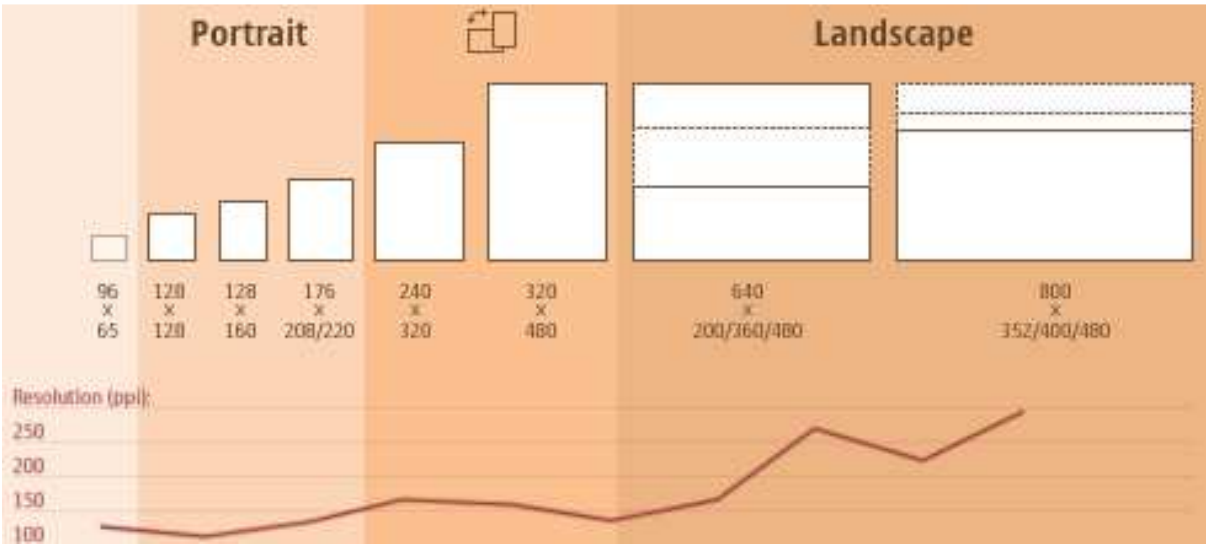


Figure 10 Popular Devices Screen Resolutions and their Popularity with New Devices

The more common approach for a responsive design is to define three (3) design variants for three large classes of devices (mobile/tablet and desktop). The designs are progressive, adding more elements as the screen factor increases (see below for an example of such a responsive design on who existing elements are re-positioned in the screen and new elements are added as more screen real-estate becomes more available).

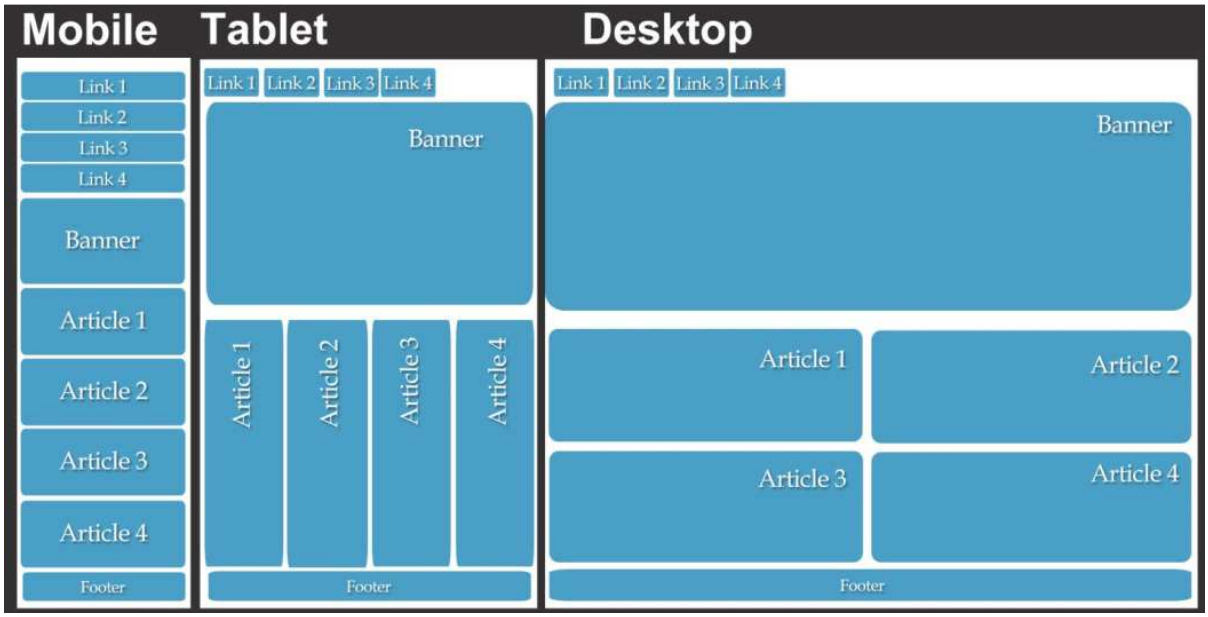


Figure 11 Example of Responsive Design

With the recent advances in web standards and responsive design techniques, it is now possible to provide a progressive, gradually enhanced experience across a wide array of browsers, using one HTML5 document and a variety of different Cascading Style Sheets (CSS). This approach does not selectively deliver content to the user through browser sniffing or server-side device detection, but is rather requesting by the mobile browser itself to render only the supported web technologies. Moreover, mobile devices that fall under the smartphone category currently have superior mobile browsers and support for HTML5 when compared to feature phones.

- **Advantages**
  - Responsive design provides additional content taking advantage of the additional screen size for larger devices like tablets and desktops (PCs).
  - Application content renders without defaults to both smartphones and tablets.
  - This approach is aligned with the development approach in which one web app handles both desktop and mobile.
- **Disadvantages**
  - responsive design requires additional effort and training to develop how the additional content should be placed by rich media applications designers for additional screen factors

DATABAIT_GUI Design II. *Aiming at creating a layout that support large audiences, with multi-browsers, multiple OS and supporting desktop, tables and high-end mobile devices, DATABAIT_GUI will follow a responsive design format.*

3.4.4. DataBait_GUI Web Application Development Designing Principles

In the previous section, emphasis is placed on justifying the two main technical design principles that will be followed for the development of DataBait_GUI (i.e., web/mobile web app with responsive design). In this section emphasis is placed on the User Interface (UI), User eXperience (UX) and QoE design principles that need to be followed towards optimising USEMP end users' experience. To that end, the following areas are analysed:

- Privacy-by-Design founded on EU Legislations
- Existing SoTA Approaches in User Personal Data Privacy Representations
- Overall UI/UX Principles (that need to be followed given the complexity of presented information).
- Due to space limitations and the nature of this deliverable, a detailed report of the latter findings is not included.

3.4.4.1. Privacy-By-Design - On EU Legislations and Directives.

An issue of high importance in the design of USEMP DataBait_GUI is the compliance with EU Directives in the area of digital data privacy, since in order USEMP System to enable the envisioned features and functionalities has to act an end-users personal data collector and controller. To that end, **Data Protection Directive** [Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data] has been mainly studied, placing emphasis on:

- **E-Privacy Directive** [Directive 2002/58, mended by Directive 2006/24/EC and Directive 2009/136/EC], especially the topics related to personal data stored and managed by smart devices.
- **Article 5 of the E-Privacy Directive,** related to application developer and the use of end-user personal data accessed and stored via their application.

In line with the above, the types of data stored on or generated by a smart device are personal data, whenever they relate to an individual, who is directly (such as by name) or indirectly identifiable to the controller or to a third party. (e.g., Location, Contacts, Unique device and customer identifiers (such as IMEI13, IMSI14, UDID1 and mobile phone number), Identity of the data subject, Identity of the phone (i.e. name of the phone), Credit card and payment data, Phone call logs, SMS or instant messaging, Browsing history, Email, etc.). As detailed in Appendix IV, the majority of the above data will be accessed via DataBait_GUI (and USEMP Tools).

Thus, "*An app developer may use third party libraries with software that provides common functionalities, such as for example a library for a social gaming platform. The app developer must ensure users are aware of any data processing undertaken by such libraries and if that is the case, that such data processing is compliant with the EU legal framework, including where relevant, by obtaining the consent of the user. In that sense, app developers must prevent use of functionalities that are hidden from the user.*"

DATABAITp_GUI Design III. *Aiming at enabling a privacy-by-design application, DATABAITp_GUI will be developed in line with corresponding EU Legislations and*

3.4.4.2. On Personal Data Monitoring/Management Applications and Initiatives

A vast amount of current initiatives that address aspects of personal data management and privacy has been investigated, documented and analysed, from publicly available solutions to research driven efforts. As characteristic examples, we refer to:

- **Mycrocosm** [30], an innovative platform that allows to track and share personal data and statistical graphs;
- **MIT Open PDM (ID3),** a platform that allows users to collect, store, and give fine-grained access to their data all while protecting their privacy;

Current research efforts have been investigated and analysed in two key areas: a) personal data management and visualisation and b) usability studies on how end-users perceive the use and management of their digital data, in term of practicality, privacy dimensions criticality, etc. As characteristic examples we refer to:

- Stanford's Centre for Internet & Society project (Aleecia McDonald, Director of Privacy);
- W3C P3P, "The Platform for Privacy Preferences";
- Nano-Notice, a privacy disclosure study at a mobile scale;

The design of DataBait_GUI will take into consideration the latter studies.

3.4.4.3. ON DataBait_GUI Visualisation and Designs Principles

Founded on the above SoTA analysis above, USEMP DataBait_GUI user experience (UX) and user interface (UI) designing principles haven derived and documented, which will serve as the core designing beacons of USEMP DataBait_GUI prototype.

DATABAITp_GUI Design IV. *Aiming at optimising USEMP end user QoE the following UX design principles will be followed.*

- **Navigation Technique.** Returning / frequent users will not have to go over the same steps again and again. Techniques to facilitate their user experience will be applied and their choices will be remembered and be on the top or pre-selected.
- **Give options without complicating**. Asking a thousand things confuses and annoys users. We will not make everything a matter of choice; we will just offer the option to amend pre-defined selections if needed.
- **One step at the time**. We won't ask users to do everything at once. We will guide them carefully through multiple steps
- **Work on ways to allow correct navigation**. Every screen cannot be identical with the previous one but the user need to be able to understand where they are, how they have got there and what are they supposed to do.
- **We won't reinvent the wheel!** We don't need to write an application from the ground up and this is not the correct way of doing it. We use shortcuts and techniques that

[30] **Mycrocosm** by MIT Media Lab, for more details go to http://www.media.mit.edu/research/groups/1452/mycrocosm

are already the market standard. The user will be pleased if he can apply what he already knows from other apps.

- **Offering ownership**. Allows the user to make the customizations that will make him feel the application is built for him.
- **Keep it brief**. Short sentences but very well written is the key.
- **Notifications.** Notifications will be managed through Settings. You can adjust whether you want notifications on your lock screen or whether you do not want them at all. You can turn on/off the Notifications sound features.

DATABAITp_GUI Design V. *Aiming at optimising USEMP end user QoE the following UI design principles will be followed.*

DataBait_GUI will provide an "Authentically Digital" experience following the principles (for more details in WP4 task 4.3):

- The design language places emphasis on good typography
- Titles and important info appear in large font size.
- Design language can be characterised as "sleek, quick, modern, intuitive, playful"
- It is flat and without flourish
- The UI recommends consistent acknowledgement of transitions, and user interactions (such as presses or swipes) by some form of natural animation or motion.

3.4.4.4. ON DataBait_GUI Web Accessibility Options

DATABAITp_GUI Design VI. *Towards supporting USEMP user with a wide range of disabilities, two levels of accessibility features will be considered in the design of DATABAITp_GUI*

- **Web Browser Accessibility Optimisation Settings**. A plethora of key features will added to the front-end of the DataBait_GUI that enable accessibility and personalization, available for use through web browser (such as, text size and style, contrast (multiple options), line spacing, template language, use of different layout, Text To Speech functionality etc.).
- **W3C Web Content Accessibility Guidelines 2.0 Compliance.** DataBait_GUI content and HTML will be designed and developed in line with W3C Web Content Accessibility Guidelines 2.0, to ensure compliance with 3rd party assistive tools (e.g. screen-readers, magnifiers etc.).

3.4.5. Overall Architecture and USEMP_API Integration

The following table justifies how the previously defined DataBait_GUI Design principles, as well as the emerging technical and architecture solution, can reassure the fulfilment of the under consideration USEMP Tool goals (as defined at the beginning of this sub-section).

G. #.	DataBait_GUI Feature	DataBait_GUI Designing Principle
G.A.	DataBait Users Account Management	DP I
G.B.	DataBait User GUI Visualization	DP II, DP IV, DP V, DP VI
G.C.	Explicit Privacy Control	DP III
G.D.	Technical Requirements	DP I, DP II

Table 8 DataBait_GUI Designing Principles mapped to Tool's Goals

Concluding this sections analysis; the following figure provides a high level overview of the emerging final USEMP Tools client-side architecture (a technical detailed analysis of the latter is provided in the next chapter).

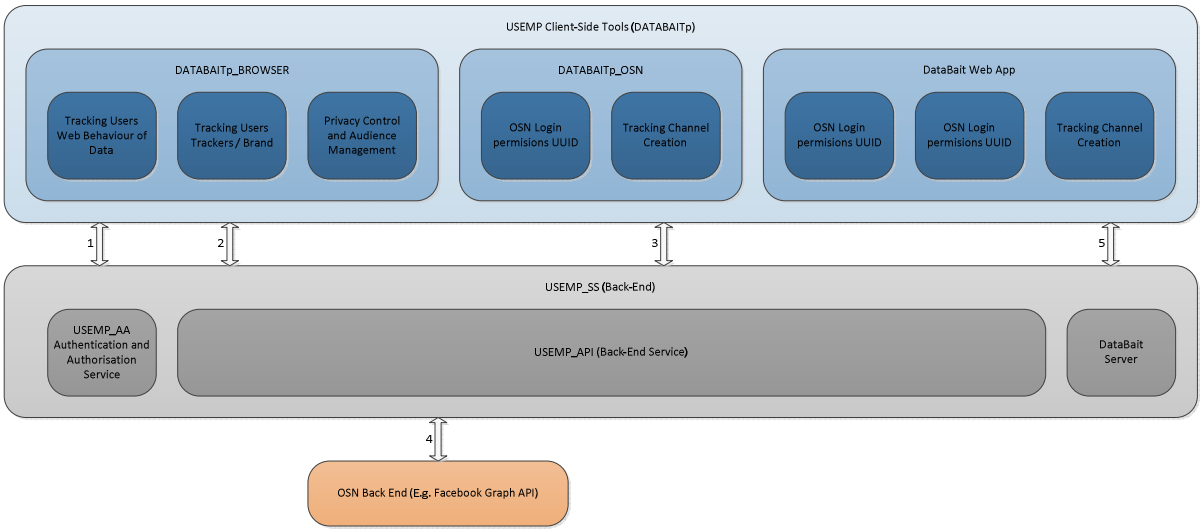


Figure 12. USEMP Client Tools Overall Architecture and API Anchor Points

DataBait_GUI web/mobile web application, upon USEMP user Logins, will communicate and interact with USEMP architecture via two anchor points.

- **DataBait_AA** is the USEMP authentication and authorisation service towards a) enabling USEMP user to register or login with their USEMP account.
- **USEMP_API** will enable the web app to retrieve USEMP Services information and graphically present them to the end users.

Concluding this section's analysis, let us underline that with the scope of this deliverable, the main technical development attributes and frameworks for creating DataBait_GUI have been analysed and justified. The actual UI design of the DataBait_GUI in terms of UI/UX flows and a number of mock-up designs will take place within the framework of WP6 Task 6.3.

3.5. Backend Services

3.5.1. Introduction to Big-Data

From the requirements of the USEMP_TOOLS and the SoTA investigation done on the DataBait_OSN (see section ***Erreur ! Source du renvoi introuvable.*** *with the* ***Erreur ! Source du renvoi introuvable.)*** and the interface to enable the publically available sources (see section 4.2.1 interface name "*connect sources (various)"*) the model of Big-Data analytics is chosen. Processing of any collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications is characterised as Big-Data processing[31].

Processing of the Big-Data is performed by the processing elements of USEMP (Technical Components, Privacy Profiling and Value Estimate). The USEMP framework needs to accommodate the processing elements of USEMP and able to scale, be flexible with multiple databases, able to orchestrate and schedule of data between processes. This framework should also be open-source in order to avoid constraints towards technologies, programming languages, scalability and licencing fees. USEMP big data framework provides a parallel processing model that can support processing of huge amount of data that relate to USEMP use cases and developed tools.

The basis for USEMP framework that can support the listed requirements above regarding the enabling technology for the services-framework is the Apache Hadoop based on Hortonworks distribution [32] [33] [34] and in particular MapReduce[35] (see Figure 13) functionality. There are alternatives to Hadoop but less popular and/or not open source (for example 'Hydra'[36], Pentaho[37]).

In addition to the big data support for USEMP developed services & tools, the selection of Hadoop as the basis for USEMP framework allows the experimentation with different options for the physical cloud infrastructure (use of private cloud, public cloud or hybrid cloud services) that can depend on the seasonality of the computational resources required for USEMP analysis tasks, the overall audience for USEMP services and the business value/cost model that can be considered for USEMP services.

[31] Information from http://en.wikipedia.org/wiki/Big_data and http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data
[32] http://hortonworks.com/hdp/
[33] http://hortonworks.com/hadoop-modern-data-architecture/#
[34] http://www.datanami.com/2013/11/06/oltp_clearly_in_hadoops_future_cutting_says/#mostRead
[35] http://wiki.apache.org/hadoop/MapReduce
[36] http://www.datanami.com/2014/03/12/hadoop_alternative_hydra_re-spawns_as_open_source/
[37] http://www.pentaho.com/

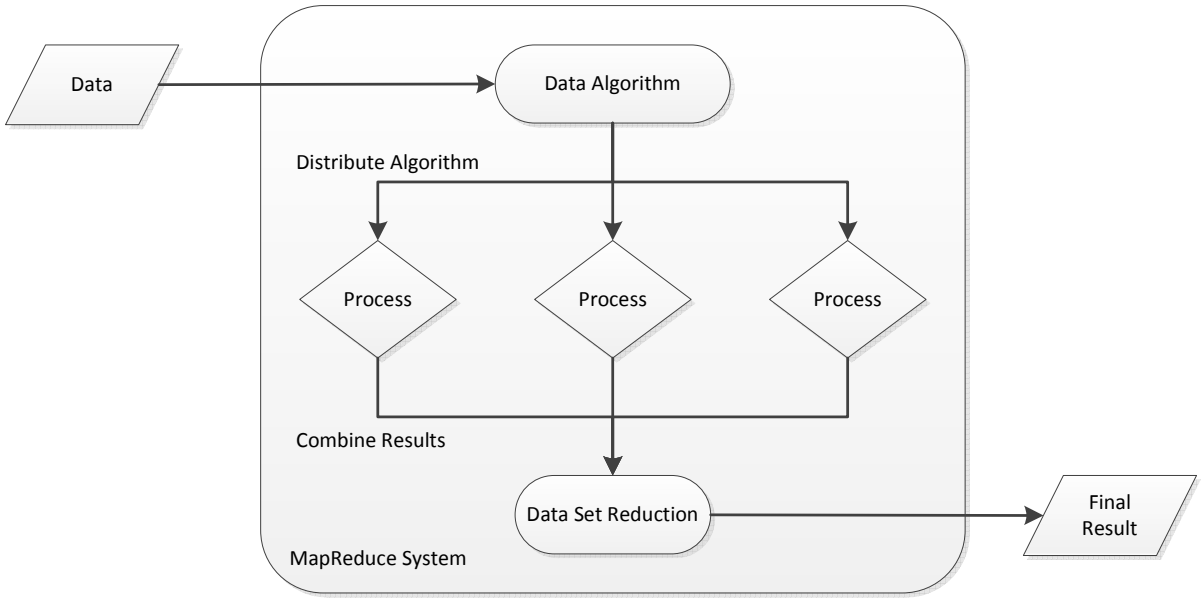


Figure 13. MapReduce Hadoop system

The OSN and the Publically available sources ("*connect sources(various)"*) instances of Big-Data gathering and processing in USEMP_SYSTEM requires data channelling from the remote system (outside USEMP_SYSTEM) to the USEMP_SS. The OSN sourcing of information follow the path OSN database → DataBait_OSN → to USEMP_SS and the "*connect sources(various)"* follow an equivalent path to the OSN from publically available sources on the web → "*connect sources(various)"* → USEMP_SS subsystem (see Figure 16. USEMP_SYSTEM architecture – TOP-LEVEL in section 4.2.2 USEMP_SS Subsystem - Component Specification).

The interfaces instances of DataBait_OSN and "connect sources (various)" will enable *"reliable, repeatable, and simple framework for managing the flow of data in and out of"* [38] big-data into the MapReduce system. The options for such a loading and management of data are Flacon[39], Oozie[40], Sqoop[41] and Flume[42]. From reading the options of loading and management of data the advantages of Flume are many. Flume[43] lets users make the most of valuable log data. Specifically, Flume allows users to:

- Stream data from multiple sources into Hadoop for analysis
- Collect high-volume Web logs in real time
- Insulate themselves from transient spikes when the rate of incoming data exceeds the rate at which data can be written to the destination
- Guarantee data delivery
- Scale horizontally to handle additional data volume

[38] "Load and manage data according to policy" of http://hortonworks.com/hdp/
[39] "What Falcon Does" of http://hortonworks.com/hadoop/falcon/
[40] "What Oozie Does" of http://hortonworks.com/hadoop/oozie/
[41] "What Sqoop Does" of http://hortonworks.com/hadoop/sqoop/
[42] "What Flume Does" of http://hortonworks.com/hadoop/flume/
[43] "What Flume Does" of http://hortonworks.com/hadoop/flume/

Once the Flume has completed the streaming of data from the publically available sources or through DataBait_OSN the data is stored in the Hadoop Distributed File System[44] (HDFS). *"HDFS is designed to be a scalable, fault-tolerant, distributed storage system"* works closely with MapReduce system (see Figure 13). Following the storage of data the MapReduce system utilises: the stored data in HDFS[45] and the USEMP processing elements (description in section 3.6.1 to 3.6.13) to produce the analytics visualised by the DataBait_GUI (see requirements in section **Erreur ! Source du renvoi introuvable.**)

3.5.2. USEMP API (client - server communication)

In the USEMP_SYSTEM, the server-side (or back-end) services requirements are many and complex with main emphasis in enabling the interfaces with the client-side, enabling Authorisation and Authentication against a database on the server side, data-acquisition from the OSN, data-acquisition from the publically available domains and data processing-management within the USEMP-SS.

USEMP TOOLS component name	Interface instance name	USEMP_SS component name	Functional requirement (Section #. #)
DataBait_BROWSER	USEMP_API with "HISTORICAL DATA DB"	Technical Components subsystem	*"DataBait_B_GoalA: Track USEMP users' personal, behavioural, tracking-information (see features coverage in Table 4) data generated via their web browser(s) (either explicitly or implicitly) and feed the latter into USEMP back-end system via USEMP API"* (section 3.2.1 and 3.2.4)
DataBait_BROWSER	USEMP_API with **(a)** "HISTORICAL DATA DB" USEMP_API with **(b)** "PRIVACY DB"	**(a)** "Technical Components" subsystem **(b) "**TC Enhancement Process" subsystem	*…"**(a)** feed the users' browsers generated personal and behavioural data to USEMP System, as well as **(b)** to retrieve the outcome of USEMP Privacy and Value estimation components towards serving as the visual and graphical interface of USEMP services." (section 3.2.4)*
DataBait_OSN or DataBait_FB	USEMP_API **(b)** with "HISTORICAL DATA DB" and AA_Account **(a)**	**(b)** Technical Components subsystem and **(a)** Identity_Manager component	*USEMP_API, will enable **(a)** the client web app to manage OSN access tokens and establish a stable USEMP System - OSN Back-end communication * **(b)** Feed users' OSN generated personal and behavioural data to USEMP System via a Flumes enabled server-to-server communication interface" (from Section 3.3.4).* ** Restrictions/limitations on the server-to-server OSN_API are included in section 3.3.3*
DataBait_GUI	USEMP_API **(a)** with PRIVACY_DB And DataBait_AA **(b)**	**(a)** "TC Enhancement Process" subsystem and **(b)** Identity Manager component	***(a)** "USEMP_API, that will enable the web app to retrieve USEMP Services information and graphically present them to the end users" (section 3.4.5)* ***(b)** …"enabling USEMP user to register or log-in with their USEMP account" (section 3.4.5)*

Table 9 USEMP_API interface features requirements summary

The USEMP_API instance that enables data-acquisition from the publically available domains (outside the USEMP_SYSTEM) requires a per database access instance interface because the volume of data and potential restrictions imposed on access suggests Big-Data

[44] "Hadoop Distributed File System (HDFS)" of http://hortonworks.com/hadoop/hdfs/
[45] "Hadoop Distributed File System (HDFS)" of http://hortonworks.com/hadoop/hdfs/

analysis model[46] (on top of the DataBait_OSN). Examples of publically available information sources are images databases[47]. Logos databases, Wikipedia[48] content etc. The placeholder-interface that enables access the 'publically available information sources' is labelled in the architecture as "*connect sources (various)"* (see section 4.2 Figure 16 and Table 13: USEMP_SS component – component specification). This interface is an instance of the USEMP_API which enable the acquisition of data for the processes in the Technical Components subsystem which have such prerequisites for the respective processing to occur.

3.5.3. USEMP DataBait_AA

The interfaces in USEMP_SS that enables the security, authorisation and authentication cross-checking in the USEMP_TOOLS (client-side) are identifiable by DataBait_AA. DataBait_AA is an interface instance is utilised in all the USEMP_TOOLS and the functional requirements are summarised Table 10.

USEMP tool name	Interface instance name	USEMP_SS service name	Functional requirement (Section #. #)
DataBait_BROWSER	DataBait_AA	Identity_Manager component	Enabling USEMP users to register or log-in with their USEMP account (section 3.2.4)
DataBait_BROWSER	DataBait_AA	Identity_Manager component	Retrieving appropriate authorisation token(s) that will allow the add-on to communicate via USEMP_API with USEMP_SS back-end (section 3.2.4)
DataBait_OSN	DataBait_AA and AA_Account	Identity_Manager component	*"The client web app to manage OSN access tokens and establish a stable USEMP System - OSN Back-end communication"* (section 3.3.4) The access tokens need to be crosschecked with OSN_USER credentials that are accessible in the USEMP_SS.
DataBait_OSN	DataBait_AA and credentials are stored in the CredDB (USEMP_SS component diagram)	Identity_Manager component	USEMP authentication and authorisation service towards enabling USEMP user to register or log-in with their USEMP account (section 3.3.4). USEMP user by making an account with USEMP_SYSTEM the access-tokens are generated.
DataBait_GUI	DataBait_AA	Identity Manager component	Part of the 'WP Requirements' WP2 and WP7 is to have features like USEMP login, Authorisation, User Registration (section 3.4.1 Table 7 and in section 3.4.5)

Table 10 DataBait_AA interface features requirements summary

3.6. Analytic Modules

The 'Analytic Modules' are a group of processes that are kindly provided by the consortium members with multiple characteristics and options where the documentation and description is available in USEMP deliverable documents D2.2 and D2.1. Following-up the general characteristics, requirements and readiness of the Technical Components implementations, in the following sections each Technical Component is described in detail (where possible) the way that could be utilised within the USEMP_SS. For a consolidated list of commands, parameters and output type's description see APPENDIX F.

[46] Information from http://en.wikipedia.org/wiki/Big_data and http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data
[47] A website that contains 1.3 billion Facebook profile pictures! http://app.thefacesoffacebook.com
[48] http://en.wikipedia.org/wiki/Wiki

3.6.1. Face Detection

The face detection tool takes image as input and outputs a prediction about the presence of one or several faces in the image. The tool will function in library mode as follows

Detection<FaceDetection> faceDet = face_detector.detect(BufferedImage img, PredictionParams params);

Prior to the execution of the method, the face detector must be initialized with a given model:

FaceDetector face_detector = new FaceDetector(DetectionResource detectionModel, Params params);

Alternatively, the model may be hard-coded and loaded by default at initialization:

FaceDetector face_detector = new FaceDetector (Params params);

3.6.2. Face Recognition

The face recognition tool takes a list of detected faces in an image as input and outputs a prediction about the presence of one or several faces from a predefined list in the image. The tool will function in library mode as follows

Recognition<FaceRecognition> face_recognized = face_recognition.recognize(Detection<FaceDetection> faceDet, , PredictionParams params);

Prior to the execution of the face recognition method, the face recognizer must be initialized with a model:

FaceRecognizer face_recognition = new FaceRecognizer (RecognitionResource recognitionModel, Params params);

Alternatively, the model may be hard-coded and loaded by default at initialization:

FaceRecognizer face_recognition = new FaceRecognizer(Params params);

3.6.3. Logo Recognition

The logo detection tool takes image as input and outputs a prediction about the presence of one or several logos from a predefined list in the image. The tool will function in library mode as follows

Logo< LogoRecognition > logos = logo_recognizer.detect(BufferedImage img, PredictionParams params);

Prior to the execution of the method, the logo recognizer must be initialized with a given model:

LogoRecognizer logo_recognizer = new FaceDetector(LogoResource model, Params params);

Alternatively, the model may be hard-coded and loaded by default at initialization:

LogoRecognizer logo_recognizer = new FaceDetector(Params params);

3.6.4. Multimedia Similarity

The multimedia similarity tool takes multimedia documents as inputs and outputs a prediction about their degree of similarity (i.e. similarity score). The tool will function in library mode and we foresee the following functioning mode.

Multimedia<MultimediaSimilarity> mmSim = mm.computeSimilarity(MultimediaFeatures mmFeats1, mmFeatures mmFeats2, TextResources textModels, ImageResources imgModels, PredictionParams params);

Prior to the execution of the multimedia similarity method, an internal call will be made to TextSimilarity, in order to extract text and/or image features and then to combine them:

TextSimilarity textFeats = txtfextractor.extract(BufferedText text, FeatureParams params);
MultimediaSimilarity imgFeats = imgfextractor.extract(BufferedImage img, FeatureParams params);
MultimediaFeatures mmFeats = merger.merge(TextFeatures textFeats, ImageFeatures imgFeats);

The above methods should be publicly exposed because they are also used by Opinion Mining and Content Location modules.

3.6.5. Text Similarity

The text similarity tool takes two texts as inputs and outputs a prediction about their degree of similarity (i.e. similarity score). The tool will function in library mode as follows.

Text<TextSimilarity> textSim = text.computSimilarity(TextFeatures textFeats1, TextFeatures textFeats2, TextResources textModels, PredictionParams params)

Prior to the execution of the text similarity method, an internal call will be made to TextSimilarity in order to extract text features:

TextSimilarity textFeats = txtfextractor.extract(BufferedText text, FeatureParams params);

The above method should be publicly exposed because it is also used by Opinion Mining and Content Location modules.

3.6.6. Opinion Mining

The opinion mining tool takes text features as input and outputs predictions (scores) about the opinion expressed in the input with three possible values: positive, neutral or negative. The tool will function in library mode as follows:

Opinion<OpinionMining> opin = opinion.extract(TextFeatures textFeats, OpinionResources opinionModels, PredictionParams params);

Prior to the execution of the opinion mining method, an internal call will be made to TextSimilarity in order to extract text features:

TextSimilarity textFeats = txtfextractor.extract(BufferedText text, FeatureParams params);

The above method should be publicly exposed.

3.6.7. Content Location

The content location tool takes a text features, image features or a combination of the two as inputs and outputs predictions (scores) about the most probable geographic location of that document. Location is expressed as a pair of latitude-longitude coordinates. As a result, we foresee three functioning modes, depending on the type of input (text alone, image alone, text and image).

Coordinates<ContentLocation> coord = content.locate(TextFeatures textFeats, ImageFeatures imgFeats, TextResources textModels, ImageResources imgModels, PredictionParams params);
Coordinates<ContentLocation> coord = content.locate(TextFeatures textFeats, NULL, TextResources textModels, NULL, PredictionParams params);
Coordinates<ContentLocation> coord = content.locate(NULL, ImageFeatures imgFeats, NULL, ImageResources imgModels, PredictionParams params);

Prior to the execution of these methods, these variants will internally make calls to text and image feature extractors from TextSimilarity and MultimediaSimilarity respectively with the following calls:

TextSimilarity textFeats = txtfextractor.extract(BufferedText text, FeatureParams params);
MultimediaSimilarity imgFeats = imgfextractor.extract(BufferedImage img, FeatureParams params);

The above feature extraction methods from TextSimilarity and MultimediaSimilarity should be publicly exposed

3.6.8. Personal Attribute Multimedia Predictor

The personal attribute multimedia predictor takes in images or image features and outputs predictions (scores) for a set of given concepts/attributes of interest (e.g. smoking, drinking, etc.). As a result, we foresee two variants of the method in library mode:

List<AttributePrediction> preds = predictor.predict(BufferedImage img, PredictionParams params);
List<AttributePrediction> preds = predictor.predict(ImageFeatures feats, PredictionParams params);

The first variant of the method will internally make a call to an appropriate feature extraction method of the form:

ImageFeatures feats = fextractor.extract(BufferedImage img, FeatureParams params);

In case such features are used by other modules (e.g. Multi-Modal Similarity), the above method should be publicly exposed and used by all dependent modules.

In both variants of the function, one should note that the AttributeMultimediaPredictor module should be initialized with an appropriate classification model **once,** since the model might be time-consuming to load (i.e. a big file):

AttributeMultimediaPredictor predictor = new AttributeMultimediaPredictor(MultimediaModel mmodel);

See D2.2, section 4.8 for an example of the intended behaviour for this model.

3.6.9. Personal Attribute Behavioral Predictor

The personal attribute behavioural predictor takes in a list of Facebook likes and a list of visited webpages and outputs predictions (scores) for a set of given concepts/attributes of interest.

List<AttributePrediction> preds = predictor.predict(List<UserActivity> transactions, PredictionParams params);

Where UserActivity should model both a Facebook like (i.e. contain an id of the liked Page or Post) and a visited webpage (i.e. contain a URL and time of visit). The same initialization consideration holds for this module as well: The AttributeBehavioralPredictor should be initialized with an appropriate classification model once, since the model will be time-consuming to load:

AttributeBehaviouralPredictor predictor = new AttributeBehaviouralPredictor(BehaviouralModel bmodel);

See D2.2, section 4.9 for an example of the intended behaviour for this model.

3.6.10. Word Count

The word count function takes in raw text and outputs statistics on the input text. This is performed differently dependent upon whether the system is run from the command line, or in library mode in which the text can be directly streamed into the system.

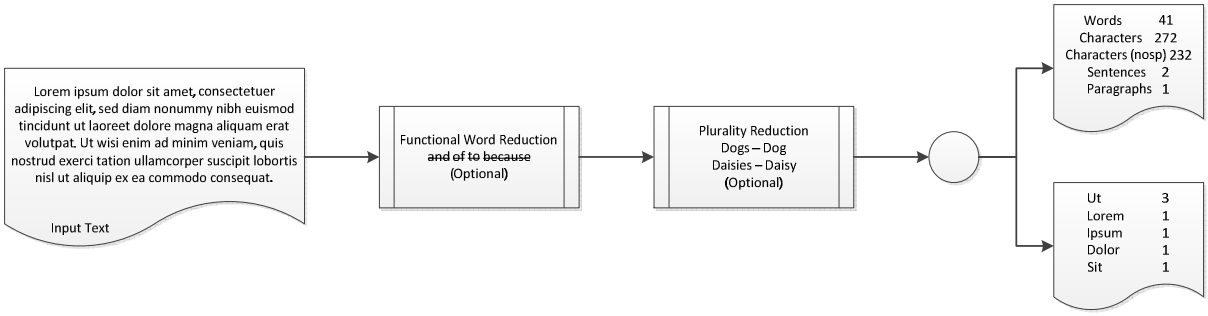


Figure 14. Word Count TC – example diagram

The primary parameters are the input text and then flags to define the addition optional settings.

- **Command-line Mode**. In command line mode, text must be input as plain text files. The URI to the file is required along with the associated options passed as flags. If custom word lists are used these must be also be passed as new line delimited text files containing word black lists.

java –jar wordcount.jar \path\to\input.txt
java –jar wordcount.jar \path\to\input.txt –wr –pr
java –jar wordcount.jar \path\to\input.txt –wr –pr –bl \path\to\blacklist.txt

In the first case the system is run with default parameters (word reduction and plurality reduction disabled, no blacklist). In the second case word reduction (-wr) and plurality reduction (-pr) and enabled. Finally in the third case a blacklist (-bl) is defined with all options turned on.

- **Library Mode.** In library mode, textual data must be streamed in via a 'StringStream' object. This allows for any type of text to be submitted for statistical calculation, or even a number of files to be processed together as one. In this version, the system is directly pulled into a Java program for execution. Input files must first be prepared

in order to be presented as a StringStream. The system can then be used as a standard java library:

WordStatistics ws = WordCount(StringStream input, bool WordReduction, bool PluralityReduction, List BlackList)

3.6.11. Tracking and Analytics

Tracking and Analytics Tool goal is to obtain advanced USEMP user-centric profile information based on the analysis, synthesis and aggregation of their personal and behavioural data collected in USEMP_SS Historical DB. Specifically the Tool will filter incoming raw data collected at USEMP_SS Historical DB via USEMP_Tools per user and per time period and thus, generate value metadata regarding a user's profile such as:

- The top most frequently visited URLs (sites) by the user during a specific time duration;
- The top most frequently visited domain by the user during a specific time duration;
- The top most frequent online activates by the user during a specific time duration;
- The volume of users OSN actions (e.g., #likes/comments/shares) over a time period;
- The distribution of users new friends over time;
- The frequency a user interacts with an OSN, the average time spend, and her/his popular activities;

The latter result will be stored periodically in USEMP Privacy DB, towards updating overall users profile and thus, to be accessible via ETL_Orchestrator component towards the rest of USEMP Tools to access them.

In order to achieve the above goal/operation Tracking and Analytics Tool will use the following method in library mode:

List<UserProfileAggregationsAttributes> upag= analytics.aggregate(user id, time_period_start, time_period_end, user profile attributes);

The function will take as input a) USEMP user unique ID, b) the timestamps of the under consideration period and c) the parameters that need to be estimated (from a given/predefined list) i.e., one or more of the above user profile metrics and thus, provide as output the values of thee required metrics in a list.

3.6.12. PRIVACY PROFILING

The purpose of privacy profiles is to provide a succinct, yet informative representation (compared to the raw transactions/data representation) that will be useful for a number of applications: ad targeting, article/product recommendation, friend recommendation, etc. There are different components in a privacy profile, e.g. demographic attributes, thematic-topic interests, personality traits, etc. A more detailed description of the user profiling representation takes place in the context of WP6 and will be made available in D6.1.

This processing component requires accessing of the raw (straight from USEMP_TOOL) dataset of HISTORICAL_DATA_DB information stored within the USEMP_SS and the results from the Technical Components in order to perform effectively the analysis of Privacy Profiling factors.

Several of the aforementioned privacy profile components may be modelled using one or more of the following representations:

- Vector of concepts/attributes and associated scores: In that case a set of concepts/ attributes of interest need to be selected (e.g. by using an established taxonomy such as *IAB* or *dmoz.org* in case of topic-oriented profiling or an established psychological profiling framework such as the Five Factor Model) and each user is represented by a vector of scores across the selected concepts/attributes. Scores indicate the extent to which a user is well characterised by the corresponding concept/attribute.
- Vector of keywords and associated scores: This is similar to the previous representation with the exception that it is not required to select a pre-specified list of concepts, but the vector could contain arbitrary keywords. This is typically a finer-grained, yet lower-level (and potentially less meaningful) profile representation.
- Latent space representation: This constitutes an abstract representation that attempts to model a user as a low-dimensional vector in the Euclidean space. Although this representation is not useful for human interpretation, it can be used by data mining components, e.g. for the classification of a user into one of a number of target classes or to compute similarities between users (useful for recommender systems).

3.6.13. VALUE ESTIMATION

Value Estimate (denoted as Value_Estimate is USEMP_SS Diagram) is one of USEMP System's post-metadata processors, part of TC_Enhancement_Process subsystem. The purpose of Value Estimate is to provide USEMP users an intuitive estimation of the value of their digital personal data shared online (either directly shared in social networks (e.g., Likes of FB) or indirectly collected by various network actors that track their web browser activities), in terms of:

- Economic value insights of their personal information disseminated online;
- A dynamically obtained value of a user's created data that will further depict the corresponding privacy compromises made;

A detailed description of the USEMP Value Estimate component and the corresponding algorithm that will be implemented takes place in the context of WP6 and will be made available in D6.2.

In this section, the envisioned technical characteristic of the Value Estimate component are highlighted, placing emphasis on a) the required input/output interfaces and data modelling as well as, b) the definition of the sub-components it will consist of.

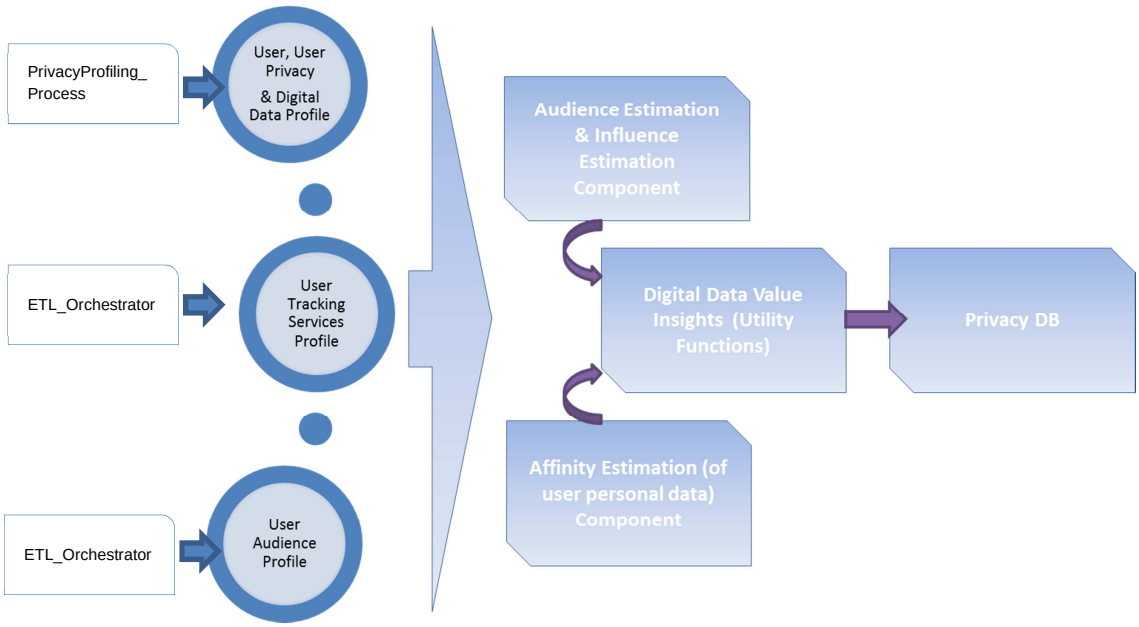


Figure 15. USEMP Value Estimate Component High Level Diagram

As depicted in the above Figure 15, Value Estimate requires as input:

- From PrivacyProfiling_Process, multiple USEMP users' privacy profile components attributes such as: digital data enhanced profile, vector of concepts/attributes and associated scores, vector of keywords and associated scores and latent space representation;
- From ETL_Orchestrator, USEMP users' trackers profile attributes (i.e., online tracking/analytics services such as ad networks, brand, analytics, that collect users personal data from the browsers and OSNs);
- From ETL_Orchestrator, USEMP users' audience estimations and profile (i.e., user OSN friend and followers that affect/interact via their OSN actions);

The latter information will be processed, aggregated and fed into the following sub-components:

- **Audience Estimation & Influence Estimation Component,** will obtain normalised indicators how relevant a USEMP user profile is to different stakeholders. Moreover, an estimation of the volume USEMP user audience will be obtained.
- **Affinity Estimation (of user personal data) Component,** will compute a ranking of USEMP user's content base [affinity, weight, time decay] (e.g., the potential ranking of a type of post], indicator of his potential rewording by a brand when she/he clicked Likes on pages of these brands, posted pictures of himself using these brands on the web, uploaded movies mentioning the brand or when he liked posts on the web related to the brand.

The above subcomponents will feed the final sub-component of Value Estimate i.e.

- **Digital Data Value Insights,** that will derive a normalised value of USEMP users' digital data and social footprint that they either directly shared in social networks (e.g., Likes of FB) or were indirectly collected by various network actors that track their activities on his web browser.

The outgrowth of the above sub-component will be fed and stored in USEMP Privacy DB. Then, via USEMP_API Value Estimate information will be provided to DataBait_GUI toward presenting the latter to USEMP users in intuitive graphical representations.

4. System Architecture, Interfaces and Integration

4.1. System Overview

The overview of the USEMP_SYSTEM is described in the concept architecture with the technical characteristics of the architecture-components. In this section the components, interfaces and interactions are going to be visualised at top-level and subsystem level. The top-level of the USEMP_SYSTEM is presenting the main the interaction (interface and ports) between the client-side subsystem (USEMP_TOOLS subsystem) and the server-side services and processes subsystem (USEMP_SS subsystem). The second-level to the architecture includes characterisation of the subsystems components and interfaces. These diagrams constitute the technical specifications of how the parts of the system are to be integrated across the system as a whole.

4.2. USEMP_SYSTEM Integration

USEMP_SYSTEM component diagram

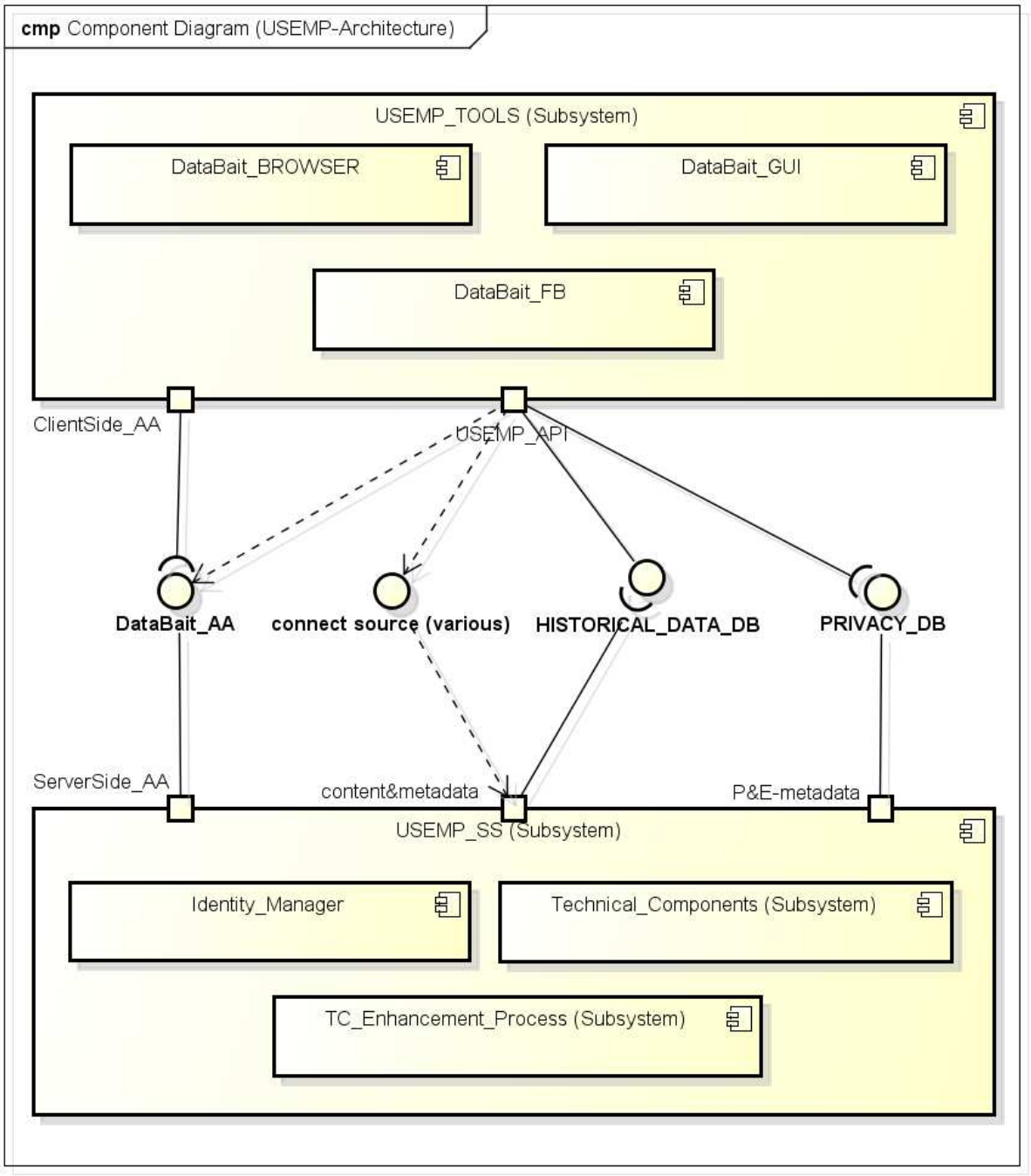


Figure 16. USEMP_SYSTEM architecture – TOP-LEVEL

The build-priority of components included in Figure 16. USEMP_SYSTEM architecture – TOP-LEVEL is documented bellow.

Component	Priority (1: highest i.e. first prototype … 4: lowest last prototype-optional)	Reason
USEMP_TOOLS	1	
USEMP_SS	1	

Table 11: USEMP_SYSTEM Architecture – component priority

4.2.1. USEMP_TOOLS Subsystem - Component Specification

Component Name	*Name = USEMP_TOOLS* *Subsystem = USEMP_SYSTEM*
Purpose (what, not how)	*USEMP_TOOLS is a subsystem that includes client side components where the interaction with the OSN_USER actor are managed.*
Inputs	*Authorisation and Authentication credentials cross-check is verified by the Identity manager and the response is provided by USEMP_SS component (DataBait_AA)* *A post-processing instance is provided from USEMP_SS in order to present the USEMP_WORLD_VIEW to OSN_USER by DataBait_GUI about the OSN interactions collected and processed by USEMP_SS (PRIVACY_DB)*
Outputs	*DataBait_BROWSER collects a digital trail dataset and is communicated to the USEMP_SS for further processing using the USEMP_API port (part of the HISTORICAL_DATA_DB)* *Day zero instance only is an instance where the OSN_USER created a USEMP account and provides the credentials (to Identity Manager component) needed by USEMP_SS to acquire the OSN datasets for processing (DataBait_AA)* *OSN_USER credential are cross-checked by Identity Manager component (DataBait_AA)*
Dependencies	*OSN_USER (actor)* *'connect Source (various)' (external)*
Interfaces	*DataBait_AA | required (Normal operation of USEMP_SYSTEM)* *DataBait_AA | provided (on USEMP_SYSTEM install-day-0 only)* *HISTORICAL_DATA_DB | provided* *PRIVACY_DB | required*
Ports	*ClientSide_AA | direction: bi-directional* *USEMP_API | direction: bi-directional*
Contracts	*DataBait_AA | contract: invariant | OSN_USER credential need to be cross-checked using the identity manager* *USEMP_API port | contract: pre-condition | Components inside the USEMP_TOOLS can interact with the USEMP_SS using the USEMP_API only when the credential have been cross-checked.*
Diagram	*USEMP_TOOLS Subsystem component is shown in Figure 16. USEMP_SYSTEM architecture – TOP-LEVEL*

Table 12: USEMP_TOOLS component – component specification

4.2.2. USEMP_SS Subsystem - Component Specification

Component Name	*Name = USEMP_SS* *Subsystem = USEMP_SYSTEM*
Purpose (what, not how)	*USEMP_SS is a subsystem that includes server side components of the USEMP_SYSTEM where the information from the OSN are processed and the OSN_USER credentials are managed.*
Inputs	*Authorisation and Authentication credentials cross-check is verified by the Identity manager (DataBait_AA)* *Digital-trail from the client-side plugins (HISTORICAL_DATA_DB)* *Open source data from wiki, images DB, news DB etc. ('various connect (various)')* *OSN data is pulled using the DataBait_FB directly to the HISTORICAL_DATA_DB ('connect source (various)')*
Outputs	*Authorisation and Authentication credentials cross-check is verified by the Identity manager (DataBait_AA)* *A post-processing instance is provided from USEMP_SS in order to present the USEMP_WORLD_VIEW to OSN_USER by DataBait_GUI about the OSN interactions collected and processed by USEMP_SS (PRIVACY_DB)*
Dependencies	*OSN accounts by OSN_USER (actor)* *'connect source (various)' (external)* *Identity Manager (component)*
Interfaces	*DataBait_AA | provided* *'connect Source (various)' | dependency* *HISTORICAL_DATA_DB | required* *PRIVACY_DB | provided*
Ports	*ServerSide_AA | direction: bi-directional* *Content&metadata | direction: input* *P&E-metadata | direction: output*
Contracts	*DataBait_AA | contract: invariant | OSN_USER credential need to be cross-checked using the identity manager* *Content&metadata port | contract: pre-condition | Collect the prerequisites HISTORICAL_DATA_DB content and metadata for the Technical_Components (Subsystem).* *Content&metadata port | contract: pre-condition | Collection of information from open-sources is essential for the processing that occurs in Technical_Components (Subsystem)* *P&E-metadata port | contract: post-condition | The content&metadata from USEMP_SS processing occurring in Technical Components (Subsystem) and TC_Enhancement_Process (Subsystem) is accumulated in the PRIVACY_DB where the graphical overlay will be applied and visible (by the OSN_USER) on the DataBait_GUI*
Diagram	*USEMP_SS subsystem component is shown in Figure 16. USEMP_SYSTEM architecture – TOP-LEVEL*

Table 13: USEMP_SS component – component specification

4.3. USEMP_TOOLS Integration

USEMP_TOOLS subsystem component diagram

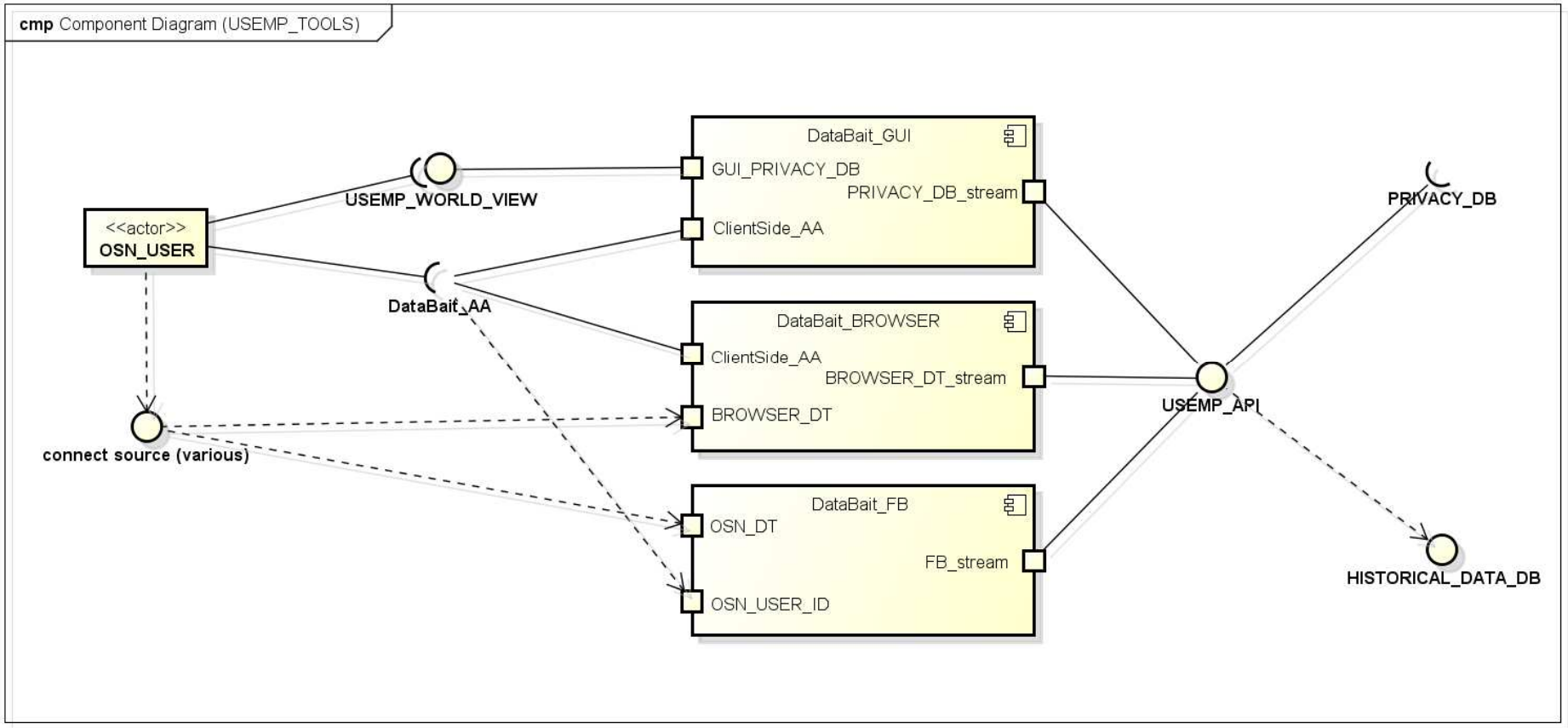


Figure 17. USEMP_TOOLS – subsystem component diagram

The build-priority of components included in Figure 17 is documented bellow.

Component	Priority (1: highest i.e. first prototype … 4: lowest last prototype-optional)	Reason
DataBait_GUI	1	Serving as the main USEMP User GUI.
DataBait_BROWSER	1	Serving as the prime USEMP user persona online data collector.
DataBait_FB	1	Serving as the prime USEMP user persona online data collector.

Table 14: USEMP_TOOLS – subsystem component priority

4.3.1. DataBait_GUI - Component Specification

Component Name	*Name = DataBait_GUI* *Subsystem = USEMP_TOOLs*
Purpose (what, not how)	*DataBait_GUI will serve as USEMP graphical users' interface, enabling USEMP users to access, visualise, exploit and interact with USEMP services and features via their computers and/or mobile devices. Specifically, DataBait_GUI will be a developed as web/mobile-wed application, supporting various device and browser types (e.g., desktop pc, tablet and mobile device browsers) towards reassuring cross-platform, cross-device and cross-browsers compatibility and rendering optimisation.*
Inputs	*DataBait_AA: Authorisation and Authentication credentials cross-check. Response is provided by USEMP_SS component (DataBait_AA)* *USEMP_API: A post-processing instance is provided from USEMP_SS in order to present the USEMP_WORLD_VIEW to OSN_USER via USEMP API.*
Outputs	*USEMP_WORLD_VIEW and ultimately USEMP_OSN (ultimately the end user)*
Dependencies	*OSN_USER (actor)* *'connect Source (various)' (external)* *Identity Manager (component)*
Interfaces	*DataBait_AA | provided* *'connect Source (various)' | dependency* *USEMP_API | provided* *HISTORICAL_DATA_DB | required* *PRIVACY_DB | provided*
Ports	*ClientSide_AA | direction: bi-directional* *USEMP_API | direction: bi-directional*
Contracts	*DataBait_AA | contract: invariant | OSN_USER credential need to be cross-checked using the identity manager* *USEMP_API port interaction | contract: pre-condition | Components inside the USEMP_TOOLS can interact with the USEMP_SS using the USEMP_API only when the credential have been cross-checked.*
Diagram	*DataBait_GUI is a component part of the Figure 17. USEMP_TOOLS – subsystem component diagram*

Table 15: DataBait_GUI – component specification

4.3.2. DataBait_BROWSER - Component Specification

Component Name	*Name = DataBait_BROWSER* *Subsystem = USEMP_TOOLs*
Purpose (what, not how)	*DataBait_BROWSER is browser plug-in (add-on), supporting for the most popular web browsers i.e., Chrome, Firefox and Safari, and will serve as users' digital trail tracker and aggregator, specific to the browser utilized by the data-subject (USEMP User).* *1) Track USEMP users' personal and behavioural data generated via their web browser(s) (either explicitly or implicitly) and feed the latter into USEMP back-end system via USEMP API, denoted as USEMP_API.* *2) Tack USEMP users' trackers, in terms of 3rd party online tracking and analytics services, that monitor and collect end-users digital trail on their web browser(s).* *3) Enable USEMP users to identify their web browser tracker(s) and thus, define fine-grained do not track (DNT) rules (in a flexible and intuitive manner).*
Inputs	*DataBait_AA: Authorisation and Authentication credentials cross-check. Response is provided by USEMP_SS component (DataBait_AA)* *USEMP_API: A post-processing instance is provided from USEMP_SS in order to present the USEMP_WORLD_VIEW to OSN_USER via USEMP API.*
Outputs	*USEMP_WORLD_VIEW and ultimately USEMP_OSN (ultimately the end user)*
Dependencies	*'connect Source (various)' (external)* *Identity Manager (component)*
Interfaces	*DataBait_AA | provided* *'connect Source (various)' | dependency* *USEMP_API | provided* *HISTORICAL_DATA_DB | required*
Ports	*ClientSide_AA | direction: bi-directional* *USEMP_API | direction: bi-directional*
Contracts	*DataBait_AA | contract: invariant | OSN_USER credential need to be cross-checked using the identity manager* *USEMP_API port interaction | contract: pre-condition | Components inside the USEMP_TOOLS can interact with the USEMP_SS using the USEMP_API only when the credential have been cross-checked.*
Diagram	*DataBait_BROWSER is a component part of the Figure 17. USEMP_TOOLS – subsystem component diagram*

Table 16: DataBait_BROWSER – component specification

4.3.3. DataBait_OSN - Component Specification

name Component	*Name = DataBait_OSN* *Subsystem = USEMP_TOOLs*
Purpose (what, not how)	*DataBait_OSN is an OSN-enabled web/mobile web application (e.g., a Facebook application), which will function cross-browser and across user's device. DataBait_OSN and will enable USEMP users to log-in to the OSN account and thus, provide USEMP System access to their OSN stored personal data.*
Inputs	*Authorisation and Authentication credentials cross-check is verified by the Identity manager and the response is provided by USEMP_SS component (DataBait_AA)* *A post-processing instance is provided from USEMP_SS in order to present the USEMP_WORLD_VIEW to OSN_USER by DataBait_GUI about the OSN interactions collected and processed by USEMP_SS (PRIVACY_DB)*
Outputs	*DataBait_AA: Authorisation and Authentication credentials cross-check. Response is provided by USEMP_SS component (DataBait_AA)* *USEMP_API: A post-processing instance is provided from USEMP_SS in order to present the USEMP_WORLD_VIEW to OSN_USER via USEMP API.*
Dependencies	*USEMP_WORLD_VIEW and ultimately USEMP_OSN (ultimately the end user)*
Interfaces	*'connect Source (various)' (external)* *Identity Manager (component)* *DataBait_AA | provided* *'connect Source (various)' | dependency* *USEMP_API | provided* *HISTORICAL_DATA_DB | required*
Ports	*ClientSide_AA | direction: bi-directional* *USEMP_API | direction: bi-directional*
Contracts	*DataBait_AA | contract: invariant | OSN_USER credential need to be cross-checked using the identity manager* *USEMP_API port interaction | contract: pre-condition | Components inside the USEMP_TOOLS can interact with the USEMP_SS using the USEMP_API only when the credential have been cross-checked.*
Diagram	*DataBait-FB is a component part of the Figure 17. USEMP_TOOLS – subsystem component diagram*

Table 17: DataBait_FB – component specification

4.4. USEMP_SS Integration

USEMP_SS subsystem component diagram

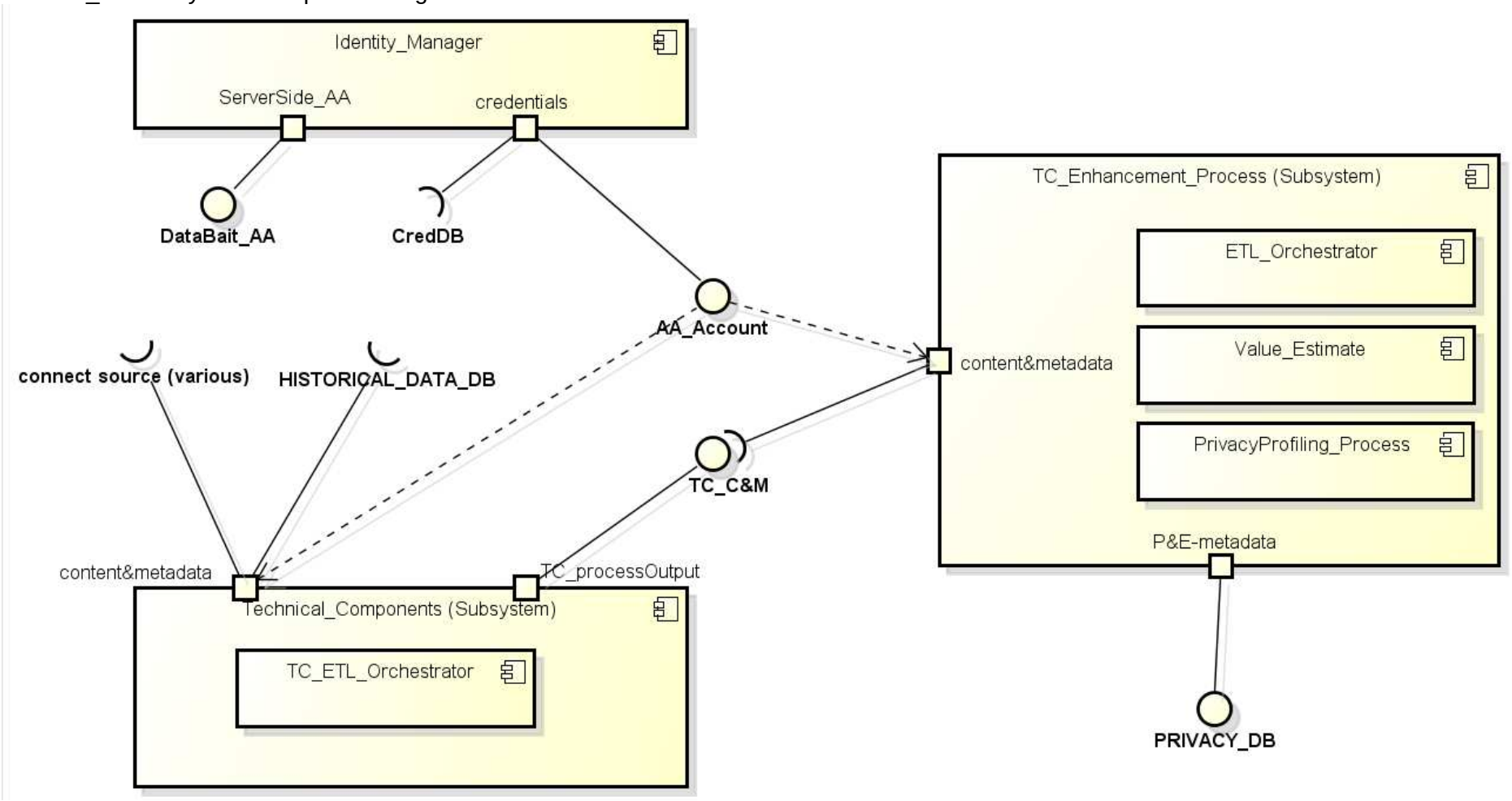


Figure 18. USEMP_SS Subsystem –component diagram

The build-priority of components included in Figure 16. USEMP_SYSTEM architecture – TOP-LEVEL is documented bellow.

Component	Priority (1: highest i.e. first prototype … 4: lowest last prototype-optional)	Reason
Identity Manager	1	
Technical Components (Subsystem)	1	
TC_Enhancement_Process (Subsystem)	1	

Table 18: USEMP_SS Subsystem – component priority

4.4.1. Identity Manager - Component Specification

Component Name	*Name = Identity Manager* *Subsystem = USEMP_SS*
Purpose (what, not how)	*Identity Manager is a component that provides credentials and cross-checking of credentials throughout the USEMP_SYSTEM. DataBait_AA is the dedicated interface for the Identity Manager to communicate to the client-side when cross-checking is required.*
Inputs	*A new OSN_USER is created and the credentials are stored in the CredDB (DataBait_AA and CredDB)* *An OSN_USER logs-in to the DataBait_GUI requires credentials cross-checking (DataBait_AA, AA_Account and CredDB)* *USEMP_SS processes are dependent on the credentials to acquire the right content&metadata (AA_Account, CredDB)* *USEMP_TOOLS subsystem provides on day zero the credentials when the OSN_USER is creating account to the USEMP_SYSTEM (DataBait_AA)*
Outputs	*Day zero instance only is an instance where the OSN_USER created a USEMP account and provides the credentials (to Identity Manager component) needed by USEMP_SS to acquire the OSN datasets for processing (DataBait_AA)* *OSN_USER credential are cross-checked by Identity Manager component and the DB of users (DataBait_AA, CredDB)*
Dependencies	*OSN_USER (actor)* *CredDB (DB interface)* *DataBait_FB port OSN_USER_ID (component port)*
Interfaces	*DataBait_AA | required* *DataBait_AA | provided (on USEMP_SYSTEM install-day-0 only)* *CredDB | provided* *AA_Account | provided*
Ports	*ServerSide_AA | direction: bi-directional* *credentials | direction: bi-directional*
Contracts	*DataBait_AA | contract: invariant | OSN_USER credential need to be cross-checked using the identity manager* *DataBait_AA | contract: invariant | OSN_USER credential need to be provided to the CredDB* *AA_Account interface | contract: pre-condition | Components inside the USEMP_SS can interact when the credential have been cross-checked with the CredDB.*
Diagram	*Identity_Manager is a component part of the Figure 18. USEMP_SS Subsystem – component diagram*

Table 19: Identity Manager – component specification

4.4.2. Technical Components Subsystem - Component Specification

Component Name	*Name = Technical_Components (Subsystem)* *Subsystem = USEMP_SS*
Purpose (what, not how)	*This subsystem encapsulates three distinct processes* • *orchestrate of content&metadata coming in* • *orchestrate commands and parameters to the Technical Component processes* • *orchestrate the content&metadata results (TC_processOutput) from the Technical Component processes*
Inputs	*Source open-data from various-sources like image DB, wiki-DB, various media news reports etc. (connect source (various))* *USEMP_TOOLS collection of data from DataBait_BROWSER and DataBait_FB is made available through the USEMP_API and processed in the Technical_Components (Subsystem) (HISTORICAL_DATA_DB)*
Outputs	*The processed content and metadata from this component is made available for further processing in TC_Enhancement_Process (Subsystem) (TC_C&M)*
Dependencies	*Identity_Manager (component)*
Interfaces	*Connect source (various) | required* *HISTORICAL_DATA_DB | required* *TC_C&M | provided*
Ports	*Content&metadata | direction: output* *TC_processOutput | direction: input*
Contracts	*Connect source (various) | contract: pre-condition | open sources of information that the Technical component require to train or process the information against need always to be there* *HISTORICAL_DATA_DB and AA_Account | contract: pre-condition | a complete set of the information from the DataBait_FB and the Digital Trail set from the DataBait_BROWSER needs to be available with the OSN_USER account cross-checked to process using the Technical Components processes.* *TC_C&M | contract: post-condition | the outcome of the Technical Components is made available after the processing is complete.*
Diagram	*Technical_Components Subsystem is a component part of the Figure 18. USEMP_SS Subsystem –component diagram*

Table 20: Technical Components Subsystem – component specification

4.4.3. TC_Enhancement_Process Subsystem

Component Name	*Name = TC_Enhancement_Process* *Subsystem = USEMP_SS*
Purpose (what, not how)	*This subsystem encapsulates three distinct processes* • *orchestrate of content&metadata coming in (TC_C&M) and the credentials of the OSN_USER (AA_Account)* • *orchestrate commands and parameters to processes Value_Estimate and PrivacyProfiling_Process* • *orchestrate the content&metadata results (P&E-metadata) from the processes*
Inputs	*Content and metadata coming in from the Technical_Components subsystem (TC_C&M)*
Outputs	*content&metadata results (P&E-metadata) from the TC_Enhancement_Process (Subsystem) processing (PRIVACY_DB)*
Dependencies	*AA_Account (Interface)*
Interfaces	*TC_C&M | required* *PRIVACY_DB | provided*
Ports	*content&metadata | direction: output* *P&E-metadata | direction: input*
Contracts	*TC_C&M | contract: pre-condition | The Technical Components Content and Metadata are required to be true when component is invoked* *PRIVACY_DB | contract: post-condition | The Orchestrated outcome from this subsystem processes is the PRIVACY_DB information*
Diagram	*TC_Enhancement_Process Subsystem is a component part of the Figure 18. USEMP_SS Subsystem –component diagram*

Table 21: TC_Enhancement_Process Subsystem – component specification

4.5. Analytic Modules Integration

Analytic Components subsystem component diagram

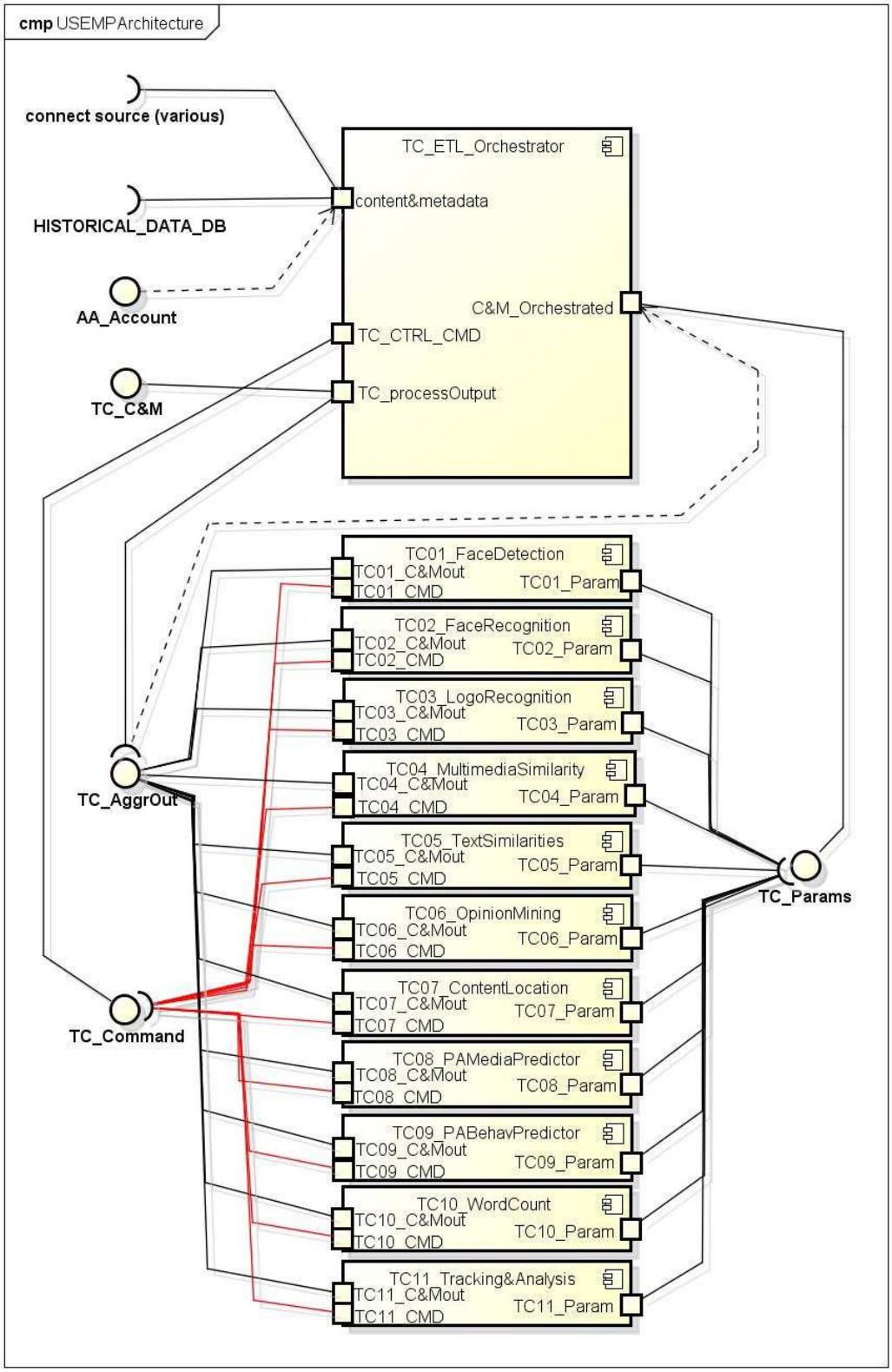


Figure 19. Technical Components – subsystem component diagram

For a consolidated list of commands, parameters and output types description see APPENDIX F. The build-priority of components included in Figure 16. USEMP_SYSTEM architecture – TOP-LEVEL is documented bellow.

Component	Priority (1: highest i.e. first prototype … 4: lowest last prototype-optional)	Reason
TC_ETL_Orchestrator	1	
TC01_FaceDetection	2	Dependent on the operation of the Orchestrator.
TC02_FaceRecognition	2	Dependent on the operation of the Orchestrator.
TC03_LogoRecognition	2	Dependent on the operation of the Orchestrator.
TC04_MultimediaSimilarity	2	Dependent on the operation of the Orchestrator.
TC05_TextSimilarities	2	Dependent on the operation of the Orchestrator.
TC06_OpinionMining	2	Dependent on the operation of the Orchestrator.
TC07_ContentLocation	2	Dependent on the operation of the Orchestrator.
TC08_PAMediaPredictor	2	Dependent on the operation of the Orchestrator.
TC09_PABehavPredictor	2	Dependent on the operation of the Orchestrator.
TC10_WordCount	2	Dependent on the operation of the Orchestrator.
TC11_Tracking&Analysis	3	Dependent on the operation of the Tools and Orchestrator.

Table 22: Technical Components – subsystem component priority

4.5.1. TC_ETL_Orchestrator - Component Specification

Component Name	*Name = TC_ETL_Orchestrator* *Subsystem = Technical_Components*
Purpose (what, not how)	*The Orchestrator is the general manager of this subsystem responsible for scheduling execution and distribution-aggregation of content&metadata datasets from the technical components processes.*
Inputs	*The open-source data from websites and databases required for the Technical components training and possible iterations of the processes (content source (various))* *USEMP-account-required (or closed) sources of data collected from the USEMP_TOOLS (HISTORICAL _DATA_DB)* *Post-processing data from the Technical Components processes are collected and are available on the C&M_Orchestrated port (TC_AggrOut).*
Outputs	*Some Technical Component require an iteration-process with the subset of TC_AggrOut dataset the dependencies arrows (dotted arrow) make the TC_AggrOut available as an input of each component (TC_Params)* *The relevant subset of the HISTORICAL _DATA_DB (and/or TC_AggrOut) objects are made available to each Technical_Component (TC_Params)* *Each Technical Component requires a command scheduled by the orchestrator with the appropriate parameters and arguments for processing and store the content&metadata (TC_CTRL_CMD)* *The aggregated post-processed data from Technical Components are made available by this component (TC_C&M)*
Dependencies	*AA_Account (Interface)* *Technical Components (TC01-TC11) (Components)* *TC_AggrOut (Interface)*
Interfaces	*Connect source (various) | required* *HISTORICAL_DATA_DB | required* *TC_AggrOut | required* *TC_Command | provided* *TC_C&M | provided* *TC_Params | provided*
Ports	*content&metadata | direction: output* *TC_CTRL_CMD | direction: input* *TC_processOutput | direction: bi-directional* *C&M_Orchestrated | direction: bi-directional*
Contracts	*Connect source (various) interface | contract: pre-condition | Open source databases links need to establish and verified before certain Technical Components that require these information can execute.* *HISTORICAL_DATA_DB interface | contract: pre-condition | Databases collected from the DataBait_BROWSER and DataBait_FB need to be up-to-date and datasets need to be cross-checked with the identity manager component (OSN_USER)* *TC_Params interface | contract post-condition | The content will exist for each Technical Component that requires them as parameters* *TC_Command interface | condition: invariant | The appropriate command needs to be provided in every Technical Component to optimise the orchestration of data process* *TC_processOutput interface | condition: post-condition | This interface implements the aggregation of the Technical Components results and providing this aggregation as an output of this Subsystem.*
Diagram	*TC_ETL_Orchestrator is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 23: TC_ETL_Orchestrator – component specification

4.5.2. TC01_FaceDetection - Component Specification

Component Name	*Name = TC01_FaceDetection* *Subsystem = Technical_Components*
Purpose (what, not how)	*Takes an image as input and outputs a prediction about the presence of one or several faces in the image*
Inputs	*An image during the detection phase (TC_Params). Need the model at initialisation (TC_Command).*
Outputs	*Prediction about the presence of faces within the image. Each face is localized by a box (X, Y, W, H), where (X, Y) is the coordinates of the upper-left corner, W the width and H the height of the box (TC_AggrOut).*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC01_C&Mout | direction: input* *TC01_CMD | direction: output* *TC01_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of model parameters* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC01_FaceDetection is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 24: TC01_FaceDetection – component specification

4.5.3. TC02_FaceRecognition - Component Specification

Component Name	*Name = TC02_FaceRecognition* *Subsystem = Technical_Components*
Purpose (what, not how)	*Takes a list of detected faces in an image as input and outputs a prediction about the presence of one or several faces from a predefined list in the image*
Inputs	*An image during the detection phase (TC_Params). Needs the model at initialisation (TC_Command).*
Outputs	*For each input (detected face) it returns an identifier to the face recognized (TC_AggrOut). A special identifier "0" is dedicated to indicate that none of the face of the base has been recognized.*
Dependencies	*TC_ETL_Orchestrator (Component)* *TC01_FaceDeteciton*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC02_C&Mout | direction: input* *TC02_CMD | direction: output* *TC02_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of model parameters* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC02_FaceRecognition is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 25: TC02_FaceRecognition – component specification

4.5.4. TC03_LogoRecognition- Component Specification

Component Name	*Name = TC03_LogoRecognition* *Subsystem = Technical_Components*
Purpose (what, not how)	*Tool takes image as input and outputs a prediction about the presence of one or several logos from a predefined list in the image*
Inputs	*An image during the detection phase (TC_Params). Needs the model(s) at initialisation (TC_Command).*
Outputs	*Prediction about the presence of logos within the image (TC_AggrOut). Each logo is localized by a box (X, Y, W, H), where (X, Y) is the coordinates of the upper-left corner, W the width and H the height of the box.*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC03_C&Mout | direction: input* *TC03_CMD | direction: output* *TC03_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of model parameters* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC03_LogoRecognition is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 26: TC03_LogoRecognition – component specification

4.5.5. TC04_MultimediaSimilarity - Component Specification

Component Name	*Name = TC04_MultimediaSimilarity* *Subsystem = Technical_Components*
Purpose (what, not how)	*Takes multimedia documents as inputs and outputs a prediction about their degree of similarity (i.e. similarity score).*
Inputs	*Two multimedia documents (TC_Params)*
Outputs	*A degree of similarity expressed by a float (TC_AggrOut). The higher, the more similar the documents.*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC04_C&Mout | direction: input* *TC04_CMD | direction: output* *TC04_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of model parameters* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC04_MultimediaSimilarity is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 27: TC04_MultimediaSimilarity – component specification

4.5.6. TC05_TextSimilarities - Component Specification

Component Name	*Name = TC05_TextSimilarities* *Subsystem = Technical_Components*
Purpose (what, not how)	*Takes two texts as inputs and outputs a prediction about their degree of similarity (i.e. similarity score).*
Inputs	*Two text documents (TC_Params).*
Outputs	*A degree of similarity given by a float (TC_AggrOut). The higher, the more similar the documents.*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC05_C&Mout | direction: input* *TC05_CMD | direction: output* *TC05_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of model parameters* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC05_TextSimilarities is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 28: TC05_TextSimilarities – component specification

4.5.7. TC06_OpinionMining - Component Specification

Component Name	*Name = TC06_OpinionMining* *Subsystem = Technical_Components*
Purpose (what, not how)	*Takes text features as input and outputs predictions (scores) about the opinion expressed in the input with three possible values: positive, neutral or negative.*
Inputs	*Textual features, obtained through the module TextSimilarities (TC_Params).*
Outputs	*Three float values (TC_AggrOut), each corresponding to the following opinion regarding the input:* *- positive opinion* *- neutral opinion* *- negative opinion*
Dependencies	*TC_ETL_Orchestrator (Component)* *TC_05_TextSimilarities*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC06_C&Mout | direction: input* *TC06_CMD | direction: output* *TC06_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of model parameters* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC06_OpinionMining is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 29: TC06_OpinionMining – component specification

4.5.8. TC07_ContentLocation - Component Specification

Component Name	*Name = TC07_ContentLocation* *Subsystem = Technical_Components*
Purpose (what, not how)	*takes a text features, image features or a combination of the two as inputs and outputs predictions (scores) about the most probable geographic location of that document*
Inputs	*Takes textual and visual features and models (TC_Params). A couple (feature, model) may be unused but at least one of them must be provided.*
Outputs	*Location, expressed as a pair of latitude-longitude coordinates (TC_AggrOut).*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC07_C&Mout | direction: input* *TC07_CMD | direction: output* *TC07_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of model parameters* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned*
Diagram	*TC07_ContentLocation is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 30: TC07_ContentLocation – component specification

4.5.9. TC08_PAMediaPredictor - Component Specification

Component Name	*Name = TC08_PAMediaPredictor* *Subsystem = Technical Components*
Purpose (what, not how)	*The personal attribute multimedia prediction function takes raw images or image features input via passing a buffered-image object or an image-features object respectively. Additional parameters pertain to the classification configuration. The function output is an array-list of concepts/attributes with the associated prediction scores.*
Inputs	*Prediction model parameters (TC_Command)* *Raw images input (TC_Params) OR extracted features (TC_AggrOut)*
Outputs	*Technical Component content and metadata in this case the arraylist of concepts/attributes with the associated prediction scores (TC_AggrOut)*
Dependencies	*TC_ETL_Orchestrator (Component)* *(to be defined) TCXX_FeatureExtractor*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC08_C&Mout | direction: input* *TC08_CMD | direction: output* *TC08_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of classification model parameters (see 2.2.4.8 for command details)* *TC_Params | contract: pre-condition | explanation: Input image object.* *TC_Params | contract: pre-condition | explanation: Input image features object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC08_PAMediaPredictor is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 31: TC08_PAMediaPredictor – component specification

4.5.10. TC09_PABehavPredictor - Component Specification

Component Name	*Name = TC09_PABehavPredictor* *Subsystem = Technical Components*
Purpose (what, not how)	*The personal attribute behaviour prediction function takes raw digital trails input via passing an arraylist object. Digital trail may represent Facebook likes and visited webpages. Additional parameters pertain to the classification configuration. The function output is an arraylist of concepts/attributes with the associated prediction scores.*
Inputs	*Prediction model parameters (TC_Command)* *Raw digital trail input, i.e. Facebook likes and visited webpages (TC_Params)*
Outputs	*Technical Component content and metadata in this case the arraylist of concepts/attributes with the associated prediction scores (TC_AggrOut)*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC09_C&Mout | direction: input* *TC09_CMD | direction: output* *TC09_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of classification model parameters (see 3.6.9 for command details)* *TC_Params | contract: pre-condition | explanation: Input digital trails object.* *TC_AggrOut | contract: post-condition | explanation: Output object returned.*
Diagram	*TC09_PABehavPredictor is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 32: TC09_PABehavPredictor – component specification

4.5.11. TC10_WordCount - Component Specification

Component Name	*Name = TC10_WordCount* *Subsystem = Technical_Components*
Purpose (what, not how)	*The word count function takes raw text input either from a command line file uri, or via passing text directly as a formatted string. Additional parameters are an optional word blacklist, or enabling default functional word reduction, as well as a Boolean parameter to reduce word plurality to singular form and count all such words as identical. The function output is an arraylist of words with the associated word frequency. Statistics about the input text makeup can be queried from the object but are not returned by default*
Inputs	*TC_ETL_Orchestrator provides the command-line with all the parameters (TC_Command)* *Raw text input (TC_Params)*
Outputs	*Technical Component content and metadata in this case the arraylist of words with the associated word frequency (TC_AggrOut)*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC10_C&Mout | direction: input* *TC10_CMD | direction: output* *TC10_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: a command needs to be provided to the process with parameters. Parameters consists of: input parameters location pointer, mode of operation of the instance and output parameters location pointer (see 0 for command details)* *TC10_Params | contract: pre-condition | explanation: Input parameters location (where the TC_Command "input parameters location pointer" points to).* *TC_AggrOut | contract: post-condition | explanation: Location where the output content and metadata of the process are accumulated.*
Diagram	*TC10_WordCount is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 33: TC10_WordCount – component specification

4.5.12. TC11_Tracking&Analysis - Component Specification

Component Name	*Name = TC011_Tracking&Analytics* *Subsystem = USEMP Architecture.*
Purpose (what, not how)	*Tracking and Analytics Tool purpose is to obtain advanced USEMP users-centric profile attribute based on the analysis, synthesis and aggregation of their personal and behavioural data collected in USEMP_SS Historical DB.*
Inputs	*User ID (TC_Command)* *Time_period_start, Time_period_end, (TC_Command)* *User profile attribute to be calculated (TC_Command)* *User profile raw event data (TC_Params)*
Outputs	*Technical Component content and metadata in this case the arraylist of concepts/attributes with user profile attributes (TC_AggrOut)*
Dependencies	*TC_ETL_Orchestrator (Component)*
Interfaces	*TC_Params | required* *TC_Command | required* *TC_AggrOut | provided*
Ports	*TC11_C&Mout | direction: input* *TC11_CMD | direction: output* *TC11_Params | direction: output*
Contracts	*TC_Command | contract: pre-condition | explanation: User profile parameter details* *TC_Params | contract: pre-condition | explanation: Input raw data.* *TC_AggrOut | contract: post-condition | explanation: Output object (list) returned.*
Diagram	*TC11_Tracking&Analysis is a component part of the Figure 19. Technical Components – subsystem component diagram*

Table 34: TC11_Tracking&Analysis – component specification

4.6. TC_Enhancement_Process Integration

TC_Enhancement_Process subsystem component diagram

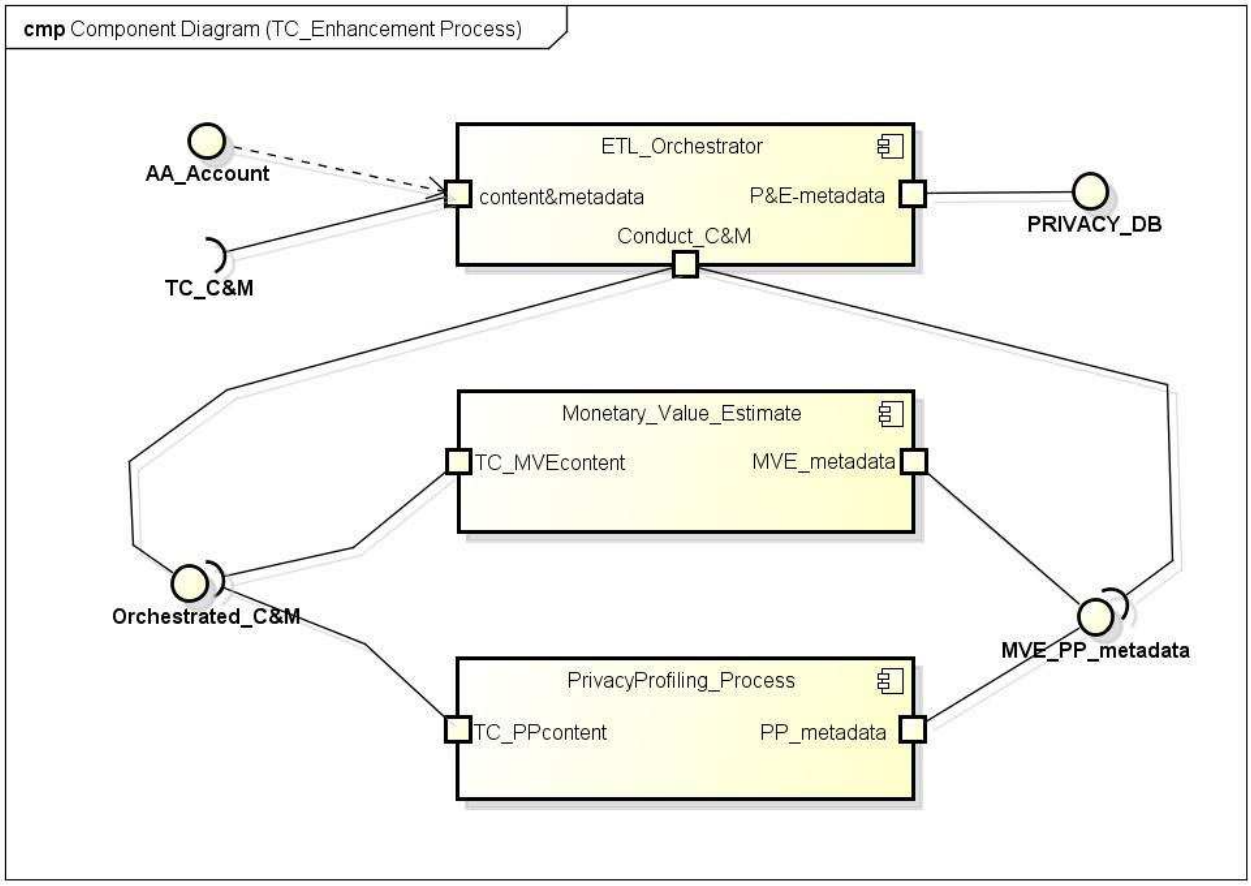


Figure 20. TC_Enhancement_Process – subsystem component diagram

The build-priority of components included in Figure 20. TC_Enhancement_Process – subsystem component diagram is documented bellow.

Component	Priority (1: highest i.e. first prototype … 4: lowest last prototype-optional)	Reason
ETL_Orchestrator	2	Requires research and input from WP5-6, preliminary implementations can include straight through of information from technical components subsystem.
PrivacyProfiling_Process	2	
Value Estimate	2	

Table 35: TC_Enhancement_Process – subsystem component priority

4.6.1. ETL_Orchestrator - Component Specification

Component Name	*Name = ETL_Orchestrator* *Subsystem = TC_Enhancement_Process*
Purpose (what, not how)	*The role of the ETL_Orchestrator in this subsystem is to provide throughput of data from the technical components output (TC_C&M) with the account crosschecked using (AA_Account) distribute through the Conduct_C&M interface the dataset to the Value_Estimate process and a copy to the PrivacyProfiling_Process and Aggregate the results. The Aggregated result from the processes are combined and made available to the PRIVACY_DB*
Inputs	*Technical Components Content and Metadata results from the Technical_Components subsystem (TC_C&M)* *The Aggregated results from Value_Estimate process and PrivacyProfiling_Process are made available to the PRIVACY_DB interface (MVE_PP_metadata)*
Outputs	*TC_C&M is are copied into the Value_Estimate process and PrivacyProfiling_Process (Orchestrated_C&M)* *Aggregate the results from Value_Estimate process and PrivacyProfiling_Process (PRIVACY_DB)*
Dependencies	*AA_Account interface from Identity Manager component (component)*
Interfaces	*Orchestrated_C&M | provided* *MVE_PP_metadata | required* *TC_C&M | required* *PRIVACY_DB | provided*
Ports	*content&metadata |output* *Conduct_C&M | bi-directional* *PRIVACY_DB | input*
Contracts	*AA_Account | contract: precondition | crosschecking the credentials of the profile* *Conductor_C&M | contract: invariant | Provide the content and metadata and the wait for the response from the processes*
Diagram	*ETL_Orchestrator is a component part of the Figure 20. TC_Enhancement_Process – subsystem component diagram*

Table 36: ETL_Orchestrator – component specification

4.6.2. Value_Estimate - Component Specification

Component Name	*Name = Value_Estimate* *Subsystem = TC_Enhancement_Process*
Purpose (what, not how)	*Aims at estimating the:* • *Economic value insights of their personal information disseminated online;* • *A dynamically obtained value of a user's created data that will further depict the user to a particular brand or product (i.e. customer value) b) social media influence and 'reach' with respect to brands/product (brand ambassadors) corresponding privacy compromises made;* *For more detailed description of this component functionality please read D6.1 of WP6*
Inputs	*See output from the ETL_Orchestrator (Orchestrated_C&M)*
Outputs	*See input from the ETL_Orchestrator (MVE_PP_metadata)*
Dependencies	
Interfaces	*Orchestrated_C&M | required* *MVE_PP_metadata | provided*
Ports	*TC_MVEcontent | direction : output* *MVE_metadata | direction : input*
Contracts	
Diagram	*Value_Estimate is a component part of the Figure 20. TC_Enhancement_Process – subsystem component diagram*

Table 37: Value_Estimate – component specification

4.6.3. PrivacyProfiling_Process - Component Specification

Component Name	*Name = PrivacyProfiling_Process* *Subsystem = TC_Enhancement_Process*
Purpose (what, not how)	*Aims at deriving estimates with respect to user OSN profile attributes that are considered "private". For more detailed description of this component functionality please read D6.1 of WP6*
Inputs	*See output from the ETL_Orchestrator (Orchestrated_C&M)*
Outputs	*See input from the ETL_Orchestrator (MVE_PP_metadata)*
Dependencies	
Interfaces	*Orchestrated_C&M | required* *MVE_PP_metadata | provided*
Ports	*TC_PPcontent | direction : output* *PP_metadata | direction : input*
Contracts	
Diagram	*PrivacyProfiling_Process is a component part of the Figure 20. TC_Enhancement_Process – subsystem component diagram*

Table 38: PrivacyProfiling_Process – component specification

APPENDIX A. Subsystem Specification template

Subsystem <NAME>
SUBSYSTEM component diagram

Component	Priority (1: highest i.e. first prototype … 4: lowest last prototype-optional)	Reason
Component <name>	3	
Component <name>	2	
Component <name>	1	
Component <name>	1	
	1	
	1	
	1	

Table 39: <name> Subsystem – component priority

APPENDIX B. Subsystem component specification template

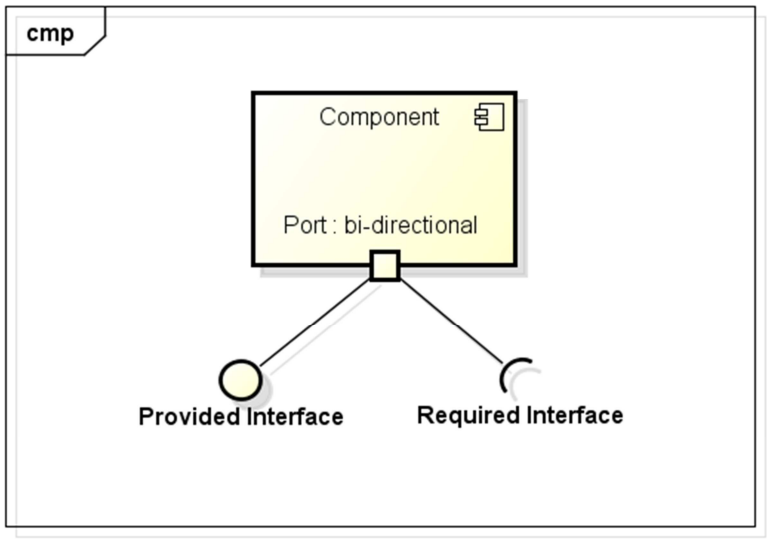
name Component	*Name = name* *Subsystem = name*
Purpose (what, not how)	*freeform description*
Inputs	** freeform description (interface name)*
Outputs	** freeform description (interface name)*
Dependencies	** name (actor or component or subsystem or external)*
Interfaces	** name | provided or required* *{tip} provided interface: one supplied by this component* *{tip} required interface: one supplied by another component (i.e. dependency*
Ports	*(tip) interaction points* ** name | direction: input or output or bi-directional* *{tip} bi-directional = a port has both required & provided interfaces;* *{tip} output = a port has only required interfaces;* *{tip} input = a port has only provided interfaces;*
Contracts	*(tip) explanation of key constraints & behaviours, per interface or applied to whole component* ** interface or whole | contract: pre-condition or post-condition or invariant | explanation* *{tip} pre-condition: that which must be true when interface or component is invoked* *{tip} post-condition: outcome when pre-conditions are true* *{tip} invariant: that which must always be true*
Diagram	*(tip) cross-reference uml2 component diagram of link to diagram* 

Table 40: <name> component – component specification

APPENDIX C. Concept Glossary

Definitions:

Digital trail	Is the trail of digital-information provided by the OSN_USER interactions and captured by the USEMP TOOLS
HADOOP	Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware
HaP#.#	Hadoop Application Processing step/s
USEMP_API	This is a transparent (to the OSN_USER) 'glue' processes otherwise recognised as Application programming interface (API) between provided and acquired access to the back-end USEMP processes and processed information.
DataBait-AA	Part of the USEMP-TOOLS is where the OSN_USER is asked to Authorise-Authenticate login to the USEMP-TOOLS in order to interact with the WEB-APP or alter the privacy preferences or to activate the installed USEMP-TOOLS
DataBait-BROWSER	Is part of the USEMP-TOOLS plugin that gathers the digital trail specific to the browser utilised by the data-subject
DataBait-FB	Is the USEMP-TOOLS plugin that gathers the digital trail specific to FACEBOOK OSN
DataBait-GUI	Is part of the client side plugin visualising USEMP_SS findings located in PRIVACY_DB
VALUE ESTIMATE	Is a process where emerged 'value/importance' (when possible) is assigned to content and/or to metadata processed by the TECHNICAL COMPONENTS. The weight-values are stored in the PRIVACY_DB in order to influence alert/warning level
OSN/s	Online Social Network/s
OSN_USER	The online social network data-subject is the 'actor' logging-in to the OSN, being inquisitive of the digital-trail, employs the USEMP-TOOLS
PRIVACY_DB	Is a database that USEMP-TOOLS builds up from processing the OSN_USER digital-trail
TECHNICAL COMPONENTS	Are the collection of functional blocks utilised to analyse aspects of the OSN_USER digital-trail (depending on the workflow) in conjunction with open source data provided by Wikipedia data, logo DB, location DB etc.
USEMP	Project full title: "User Empowerment for Enhanced Online Presence Management"
USEMP-TOOLS	(USEMP as above) 'Tools' word is used to signify the umbrella of functional blocks installed to enable the USEMP concepts. In the Architecture signifies the set of tools/GUI tools on the client side of the USEMP_SYSTEM
USEMP-SS	This architecture subsystem component characterises the group of processes that are located in the backend or Server Side (SS) of USEMP_SYSTEM

APPENDIX D. Data Structure for USEMP-TOOL and USEMP_SS communication

A. User Real-time Personal and Behavioural Browsers Data

#	Name	Description	Type	Metadata	Frequency
A1	Site Unique Visits	web sites (URL) visited by the user	URL (name)	Time Stamp	On Page-view
A2	Site Visits	# of times a user visited a web site (URL)	# of visits	Time Stamp	On Page-view
A3	Time Spent Per Site	Time a user spent during one visit. (time opened the URL at his browser)	Time	Time Stamp End, Time Stamp Start	On Browser Tab Close
A4	Images	Images Uploaded/Accessed/Downloaded by the user.	Image Object (e.g., tiff, png, gif, etc.)	Time Stamp, Type, URL of the page containing the image, the URL of the image.	On Action
A5	Videos	Videos Uploaded/Accessed/Downloaded by the user.	URL (of the video)	Time Stamp, Type, URL of the page containing the image	On Action
A6	Actions	Click on specific element at the web site.	URL	Time Stamp, Any Metadata available (to determine the type)	On Action
A7	Text	Text Uploaded/Accessed at the web site.	Text	Time Stamp, Type (e.g., email, etc.)	On Action
A8	News	Page views of specific news elements.	URL	Time Stamp, Any Metadata available (determining that the text element is news)	On Page-view

Table 41: User Real-time Personal and Behavioural Browsers Data

B. Internet Services that Track user's Data

#	Name	Description	Type	Metadata	Frequency
B1	# of Trackers for Site URL	The number of tracking services when a DataBait user visits URL	# of Trackers	Timestamp, Site URL,	On Page-view
B2	Tracker	The ID of the tracking services when a DataBait user visits a URL	URL	Timestamp, Site URL, Tracker ID (tracker URL), Tracker Type (e.g., Analytics, Advertising, Beacons, Social)	On Page-view
	Example: (021233132132131, https://www.linkedin.com/home, DoubleClick, https://ad-emea.doubleclick.net/adi/linkedin.dart, Advertising).				
B3	Tracker email	A Tracker of users email (e.g., google-mail)	URL	Timestamp, Site URL, Tracker ID (tracker URL), Tracker Type (e.g., Analytics, Advertising, Beacons)	On Page-view

Table 42: Internet Services that Track user's Data

C. User Historical and Real-time OSN Data

(FB Graph API Terminology is used)

#	Name	Description	Type	Metadata	Frequency
C1	Posts Feed [49]	An individual entry in a profile's feed. The profile could be a user, page, app, or group.	URL/JSON	Multiple see. [49]	Past and New. (Periodic search e.g., weekly)
C2	Likes and Unlikes [50]	The Facebook Pages that this person has 'liked'.	URL/JSON	Multiple see [50].	Past and New. (Periodic search e.g., weekly)
C3	Photos Or Photos Uploaded [51]	Represents an individual photo on Facebook.	URL/JSON	Multiple see [51] Tagged Images	Past and New. (Periodic search e.g., weekly)
C4	Friends-list or Friends [52]	A person's 'friend lists' - these are groupings of friends such as "Acquaintances" or "Close Friends", or any others that may have been created.	URL/JSON	Multiple see [52].	Past and New. (Periodic search e.g. weekly)
C5	Friends' activities upon user's OSN objects	Represents an action of a friend in one of a user's objects on Facebook.	URL/JSON	Multiple	New (Periodic search e.g., weekly)
C6	Ads shown to the user	Not Feasible	-	-	-
C7	News [53] (/home)	The person's news feed.	URL/JSON	Multiple see [53].	Past and New. (Periodic search e.g., weekly)
C8	User Profile_[50]-* and Interests	A user represents a person on Facebook. The /{user-id} node returns a single user.	URL/JSON	Multiple see [54].	Past and New. (Periodic search e.g., weekly)

Table 43: User Historical and Real-time OSN Data (with FB Graph API Terminology)

[49] https://developers.facebook.com/docs/graph-api/reference/v2.0/user/feed/
[50] https://developers.facebook.com/docs/graph-api/reference/v2.0/user/likes
[51] https://developers.facebook.com/docs/graph-api/reference/v2.0/photo/
[52] https://developers.facebook.com/docs/graph-api/reference/v2.0/user/friendlists
[53] https://developers.facebook.com/docs/graph-api/reference/v2.0/user/home/
[54] https://developers.facebook.com/docs/graph-api/reference/v2.0/user

D. User Personal Data for Training USEMP Tool Algorithms

(Note if a data type already exist then just repeat it)

#	Name	Description	Type	Metadata	Frequency
D1	Likes (metric: C2)	Facebook pages that the user has liked.	List of URLs (or Facebook page ids)	Timestamp of like (may be useful)	Once in the beginning to get a first bulk of personal data and then periodically to give updates to the user.
D2	Shared Pages (metric: C1)	Pages/Links that the user has shared.	List of URLs.	Timestamp of share	-Weekly
D3	Site Unique Visits (metric: A1)	URLs that the user visits in their browser.	List of URLs	Timestamp and duration of visit	Real Time
D4	Trackers (metric: B2)	Tracker URLs/ids associated with the visited websites.	List of URLs/ids	Same as #A3	Real Time
D5	FB Images (metric: C3)	Images that the user has uploaded and where the user is tagged.	List of URLs (or byte arrays)	Timestamp, accompanying metadata	Similar to #A1.
D6	User network (metric: C4)	List of friends + connections between them (probably useful for value estimation)	List of Facebook profile ids, list of connections between them	Any available profile info for friends.	Once in the beginning and then on a regular (but not very frequent basis)
D7	User friends reactions (metric: C5)	The reactions of friends in a user's posts.	List of likes, shares, comments on user's posts, comments.	Timestamp, comment text	Similar to #A1.

Table 44: User Personal Data for Training USEMP Tool Algorithms

APPENDIX E. Data-access level structure

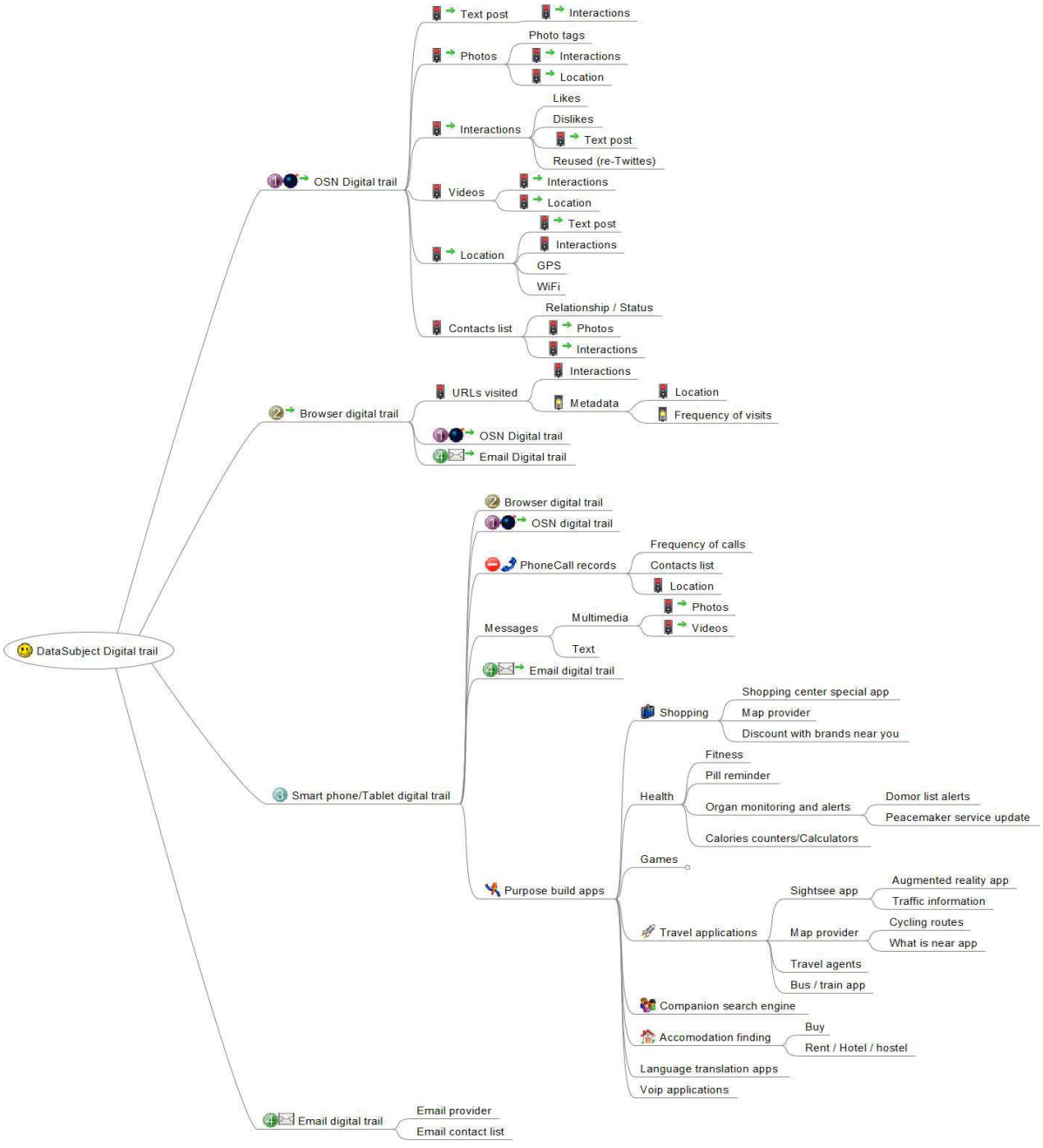


Figure 21. Data-subject digital trail free-mind

APPENDIX F. TC Requirements table

TC_ID (page)	Technical Component (TC) name + mode	TC_Command 1…n x [Class of options]	TC_PARAMS 1…n x [TYPE], …	TC_AGGROUT 1…n x [TYPE], …	ADDITIONAL
TC01 (p54)	Face detection **LIB1 mode**	1..n x [PredictionParams]	1x [Image]	1..n x [Image, position, size]	Output includes part of the input image Class name of output is <*FaceDetection*>
	Face detection **INIT mode**	1..n x [PredictionParams] 1 x [PredictionModel]			Prior to the execution of the method, the face detector must be initialized with a given model
	Face detection **LIB2 mode**		1x [Image]	1..n x [Image, position, size]	In this mode the INIT mode is already performed. Class name of output is <*FaceDetection*>
TC02 (p54)	Face Recognition **LIB1 mode**	1..n x [RecognitionParams]	1x [Image] 1x [link2DB(faces)]	1..n x String	Access to DB of faces to compare Class name of output is < *FaceRecognizer* >
	Face Recognition **INIT mode**	1..n x [RecognitionParams] 1 x [Recognition Model]	1x [Image] 1x [link2DB(faces)]		Prior to the execution of the method, the face Recognition must be initialized with a given model
	Face Recognition **LIB2 mode**		1x [Image] 1x [link2DB(faces)]	1..n x String	Class name of output is < *FaceRecognizer*
TC03 (p54)	Logo Recognition **LIB1 mode**	1..n x [PredictionParams]	1x [Image] [1x [link2DB(logos)]	1..n x String	Access to DB of logo(s) to compare in order to return the name of the logo associated Class name of output is < *LogoRecognizer* >
	Logo Recognition **INIT mode**	1..n x [PredictionParams] 1 x [LogoResource model]			Initialisation mode
	Logo Recognition **LIB2 mode**		1x [Image] [1x [link2DB(logos)]	1..n x String	Class name of output is < *LogoRecognizer* >
TC04 (p55)	Multi-Modal Similarity **INIT mode**				TC05 needs to be performed first TC07 needs to be performed first MultimediaFeatures mmFeats = Merge (TC05, TC07)
	Multi-Modal Similarity **LIB1 mode**	1..n x [MultimediaFeatures], 1 x [PredictionParams] 1 x [MultimediaFeatures]	[1x String] [1x CompareString]	1 x String	LIB1 mode: 2 strings (raw text) are evaluated regarding the concept Class name of output is < MultimediaSimilarity >
	Multi-Modal Similarity **LIB2 mode**	1..n x [MultimediaFeatures], 1 x [PredictionParams] 1 x [MultimediaFeatures]	[1x String] [1x Image] [1x PublicURL]	1 x String	Mode2: String + Image are evaluated regarding the concept in a URL Class name of output is < MultimediaSimilarity >

TC05 (p55)	Text Similarity **LIB1 mode**	1..n x [TextFeatures], 1..n x [TextModel]	[1x String] [1..n x Link2DB(PublicTEXT)	1..n x String	Domain(s) the text refers to or Domains of interest of the user. Class name of output is < TextSimilarity>
	Text Similarity **INIT mode**	1..n [FeatureParams]	[1x String] [1..n x Link2DB(PublicTEXT)	1..n x String	Initialisation mode
TC06 (p55)	Opinion Mining **LIB1 mode**	1..n x [TextFeatures] , 1..n [FeatureParams]	1x [String] 1..n x [CompareString]	1..n x [String, Enum]	LIB1 (Enum: Positive or neutral or negative characterization). Class name of output is <OpinionMining>
	Opinion Mining **LIB2 mode**	1..n x [TextFeatures], 1..n [FeatureParams]	1x [String] 1x [Image] 1..n x [PublicURL]	1..n x [String, Enum]	LIB2 (Enum: Positive or neutral or negative characterization). Class name of output is <OpinionMining>
	Opinion Mining **INIT mode**	1..n [FeatureParams]	1x [String]	1..n x [String, Enum]	Initialisation mode
TC07 (p56)	Content Location (Mode 1)		[1x Image] [1 x link2DB(Public locations)	1..n x LocStruct	Mode 1: Input only [image] is provided with the DB of locations. LocStruct is [1x coordinate pairs], [1x String corresponding place names] [1x int%] probability for the location prediction to be accurate
	Content Location (Mode 2)		[1x Text] [1 x link2DB(Public locations)	1..n x LocStruct	Mode 2: Input only [Text] is provided with the DB of locations. LocStruct is [1x coordinate pairs], [1x String corresponding place names] [1x int%] probability for the location prediction to be accurate
	Content Location (Mode 3)		[1x Image] [1x Text] [1 x link2DB(Public locations)	1..n x LocStruct	Mode 1: Input only [Image+Text] is provided with the DB of locations. LocStruct is [1x coordinate pairs], [1x String corresponding place names] [1x int%] probability for the location prediction to be accurate
TC08 (p56)	Personal Attribute Multimedia Predictor		[1x Image] [1x String]	[1x Struct(P.A.M.Pred)]	Struct is a pre-specified number of attributes for example: [(S, 0.77, TRUE), (D, 0.32, FALSE), (XS, 0.01, FALSE)] with S: Smoking, D: Drinking, XS: Extreme Sports
TC09 (p56)	Personal Attribute Behavioral Predictor	1..n x [PredictionParams], 1 x [BehaviouralModel]	1 x [<UserActivity>]	1 x [<AttributePrediction>]	Parameter “UserActivity” is an array of elements of User Activity could be Likes, Visits to websites, etc. and in the simplest case could be represented as URLs. Example of user activity is a pre-specified number of attributes for example: [(H, 0.04, FALSE), (D, 0.82, TRUE), (L, 0.72, TRUE)] with H: Homosexual, D: Diabetic and L: Liberal

TC10 (p57)	Word Count (Mode 1)	bool WordReduction	1x [String]	1..n x [String]	Output class is WordStatistics
	Word Count (Mode 2)	bool WordReduction, bool PluralityReduction	1x [String]	1..n x [Struct(TextStats)]	Output class is WordStatistics
	Word Count (Mode 3)	bool WordReduction, bool PluralityReduction, List BlackList	1x [String]	1..n x [Struct(TextStats)]	Output class is WordStatistics
TC11 (p58)	Tracking and Analytics Function	user profile attributes	user id, time_period_start, time_period_end	1..n x [Struct(Profiling_Digital Trail)]	Output class is UserProfileAggregationsAttributes