D7.3: BRIDGET Authoring Tools and Player–Report, Version B

2015-05-01 – 2016-07-31

Project ref. no.	FP7-ICT-2013-7 - 610691
Project acronym	BRIDGET
Start date of project (duration)	1 November, 2013 (36 months)
Document due Date:	2016-07-31
Actual date of delivery	2016-07-31
Leader of this document	Milos Markovic
Reply to	
Document status	Final

Deliverable Identification Sheet

Project ref. no.	FP7-ICT-2013- 610691
Project acronym	BRIDGET
Project full title	BRIDging the Gap for Enhanced broadcasT
Document name	BRIDGET D7.3 v4.0.docx
Security (distribution level)	PU
Contractual date of delivery	2016-07-31
Actual date of delivery	2016-07-31
Document number	
Type	R
Status & version	Final
Number of pages	43
WP / Task responsible	WP7
Other contributors	Nicola Piotto, Sergio García Lobo, Francisco Morán Burgos, Leonardo Chiariglione, Marius Preda, Alberto Messina, Adrian Gabrielli, Veronica Scurtu, Christian Tulvan
Author(s)	Milos Markovic
Project Officer	Alberto Rabbachin
Abstract	According to BRIDGET's DoW, the current D7.3 deliverable is a report to be delivered by 2016-07-27 describing version B of BRIDGET's Authoring Tools and Player, and presenting a progress report of the work conducted within WP7 in the 2015-05-01/ 2016-06-30 reporting period. D7.3 is accompanied by D7.4, that contains the corresponding SW.
Keywords	WP7, BRIDGET Authoring Tools, BRIDGET Player, Media Rendering
Sent to peer reviewer	2016-07-28
Peer review completed	2016-07-31
Circulated to partners	2016-07-31
Read by partners	2016-07-31
Mgt. Board approval	N/A

Version	Date	Reason of change
0.1	2016-06-15	Milos Markovic (HUA) – Document structure
0.2	2016-07-20	Sergio García Lobo and Francisco Morán Burgos (UPM) – Immersive media rendering
0.3	2016-07-25	IMT – Full document update
0.4	2016-07-26	Leonardo Chiariglione (CED) – Mini Bridget Authoring tool
1.0	2016-07-27	Milos Markovic (HUA) – Document integration
2.0	2016-07-28	Milos Markovic (HUA) – document updated after Leonardo's comment on section 3.4 and 5.2
3.0	2016-07-28	Milos Markovic (HUA) – document updated after the new material is added by IMT, sent for QC.
3.1	2016-07-31	QC completed, feedback to authors.
4.0	2016-07-31	QC comments implemented, final version.

Table of Contents

Table of Figures

Acronyms

AAC:	Advanced Audio Coding
AFP:	Audio Fingerprint
ARAF:	Augmented Reality Application Format
AT:	Authoring Tool
AVC:	Advanced Video Coding
BIFS:	BInary Format for Scenes
DoW:	Disposition of Work
GUI:	Graphical User Interface
HEVC:	High Efficiency Video Coding
HRTF:	Head Related Transfer Functions
JPEG:	Joint Picture Experts Group
MLAF:	Media Linking Application Format
MPEG:	Moving Picture Experts Group
MXM:	MPEG Extensible Middleware
PNG:	Portable Network Graphics
UI:	User Interface

1 Introduction

BRIDGET's Authoring Tools and Player were planned, realized and delivered in two phases. The **D7.1 deliverable**, delivered on 2015-04-30, described version A of BRIDGET's Authoring Tools and Player and presented a report of the work carried out within WP7 in the first 18 months of the project. D7.1 was released together with the deliverable D7.2 that contained the corresponding SW and related documentation. The current deliverable D7.3, due on 2016-07-31, describes version B of BRIDGET's Authoring Tools and Player and presents a report of the work carried out within WP7 in the following 18 months after the completion of version A.

Within the BRIDGET project, WP7 is assigned a threefold scope: *i)* designing seamless and effective user interfaces (Task -T7.1) for BRIDGET tools; *ii)* conducting research on innovative media rendering technologies, with particular emphasis on low complexity methods well suited to mobile devices (T7.3); and *iii)* integrating the results of WP4-6 research into a unified software/hardware architecture (T7.2 and T7.4).

The main results of this last 18 project months for each of those directions can be summarized as follows:

- **Conceptual design of BRIDGET User Interface (T7.1)**: BRIDGET is introducing a new concept for fruition of second screen content. This requires the design of a novel dedicated UI for enjoying such content effectively, without interfering with the main content displayed on the TV. After creating the basis of this UI in the first period of the project, the second period was used to refine it, to add additional media representations (e.g. 3D), to connect it to social network services and to allow end-user created content. Section 2 describes the studies conducted in order to design such a new user experience, maximize user appreciation and provide an easily customizable interface. Usability experts were consulted for these tasks; they reviewed several prototypes of the UI, providing feedback and suggestions that will be taken into consideration for next releases of the BRIDGET tools.
- **Research on low complexity media rendering (T7.3)**: several optimizations have been studied, in order to enable real-time rendering of advanced media formats, namely synthetic 3D models and binaural audio, in resource-constrained environments such as tablets and other mobile devices. In both cases, algorithmic optimizations were necessary, as well as the introduction of new data formats, in order to make the fruition of those formats possible within the BRIDGET player. Progress report on the conducted work is reported in Section 4.
- **Development of BRIDGET tools**: in line with the DoW schedule, additional releases of the BRIDGET Authoring Tool (AT) and BRIDGET Player have been completed and integrated into a full platform architecture (Sections 3 and 5) as designed by WP3 and described in D3.1, "BRIDGET System Architecture and Interfaces". The **second release of the Authoring Tool (T7.2)** integrates a vast number of functionalities, namely:
  - 3D graphics as destination content,
  - 3D objects reconstruction,
  - mesh and point cloud WebGL viewers for 3D objects
  - dashboard to orchestrate bridget activation during the live TV programs,
  - extensive layout editor
  - The **second release of the BRIDGET multi-screen player (T7.4)** is still able to play bridgets encapsulated into an ARAF format, and supports now user generated bridget and social networks integration. As for the first version of the player synchronization with the main broadcast content is guaranteed by the integration of the audio fingerprint engine and in the live scenario the synchronisation is done by the real-time communication with the AT dashboard.

In summary, the structure of the present document reflects the organization foreseen for WP7 in the DoW: for each task, main achievements are described, together with a report of the progress obtained in the first project year. In particular, section 2 describes the design of the user interfaces developed for the project (for bridgets and AT). Section 3 is dedicated to the architectural description of AT frontend and backend, whereas section 4 reports the progress achieved so far on the study of innovative media ren-

dering algorithms. Finally, section 5 describes the working principles of the Bridget Player and section 6 the contributions to standards.

2 T7.1 - User Interfaces and Bridget Presentation

Task 7.1 was the first one starting in WP7 and, indeed, at the beginning of the project major attention was devoted to the design of the BRIDGET user experience, as a new concept of fruition of second screen applications.

Within the current reporting period, additional concepts of the UIs for bridget production (AT) and consumption (Player) have been designed and integrated into BRIDGET architecture. Such concepts have been continuously refined, thanks to the feedback received by usability experts. Starting from these design rules, exemplary bridgets addressing the use cases defined in WP2 and the proof-of-concept studied in WP8 were implemented aiming at optimizing user experience for the audience targeted in each case.

Particular attention has been devoted to the usability, in particular on the player, in order to enable pleasant and effective bridget consumption without, at the same time, interfering with linear content displayed on the main screen.

Section 2.1 illustrates the studied design principles, while Section 2.2 describes the implementation details of the current UIs. As mentioned before, such UIs were already shown to usability experts, whose feedbacks are reported in Section 2.3.

2.1 Design Principles

2.1.1 Bridgets User Interface Design

We reported in D7.1 our analysis of the state of the art in second screen design together with recommendations emerged for existing experiments. At that time the following principle were exploited:

- An additional content should be presented on a single short page/screen, without any scrolling and including simple interactions and navigation;
- The timing of the content is critical and the synchronisation should be as good as possible;
- The additional content presented on the second screen should be relevant;
- Create rich media experiences that mirror the multitasking behaviour of the users.

One of the main drivers for designing BRIDGET UI concerns the user attention that has to be alternatively dedicated to the first and second screens. This entails to carefully design how and when the content on the second screen is displayed. The second screen should be used as a natural continuation and extension of the experience, and not as a means of interrupting the experience. Additionally, the interactive experiences on the second screen should be short enough in order to allow the user to resynchronize with the main story. This was used during the whole period of the project as the driver of transforming the traditional linear experience of consuming the TV content in its bridget'ed alternative.

While such concepts can be applied in authoring the stored programs, it does not fill the requirements of the live programs. In the last 18 months of the project we explored the concept of Live bridgets. Considering a live broadcast, it is impossible to assign bridgets to a specific timeframe because the exact content of the broadcast at a specific time is unknown. This determined the definition of a special type of bridget, which is permanently available for the broadcaster and may be published or cancelled depending on the content of the programme at a specific time.

To answer to this requirement, we introduced a new concept and the corresponding tool, the bridget orchestration and respectively, the TV program dashboard. In the case of live broadcasts, the broadcaster needs to be in control of the published bridgets and enable or disable bridgets depending on the current content of the broadcast. Nevertheless these bridgets also need to be created beforehand so that the broadcaster can comply with the time constraints and dynamism of the live programme. The dashboard of the live programs is illustrated in Figure 1.

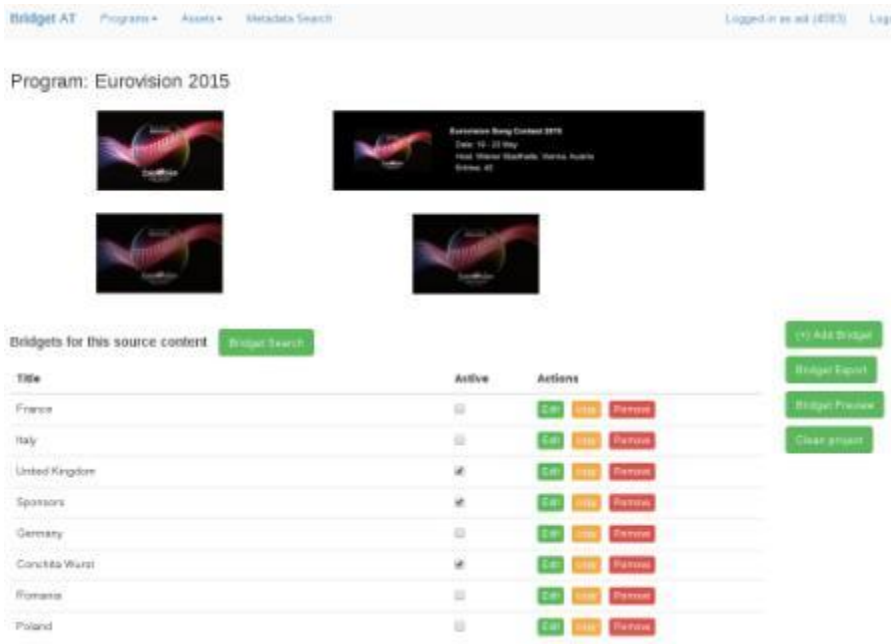


Figure 1: The dashboard of the live programs

All through the duration of the project, several UIs were tested and developed based on the results of the tests and feedback from the user trials. These UIs are presented in detail in section 2.2.

2.1.2 Authoring Tool User Interface Design

During the development of the project, an additional activity was carried on within task 2.1, namely the design of the professional AT UI. Based on the analysis of the production workflow, several conceptual designs were proposed and discussed with the consortium partners and usability experts (section **Error! Reference source not found.**). At the end of this study phase, we structured the AT UI into four design layers as presented in the following figures.

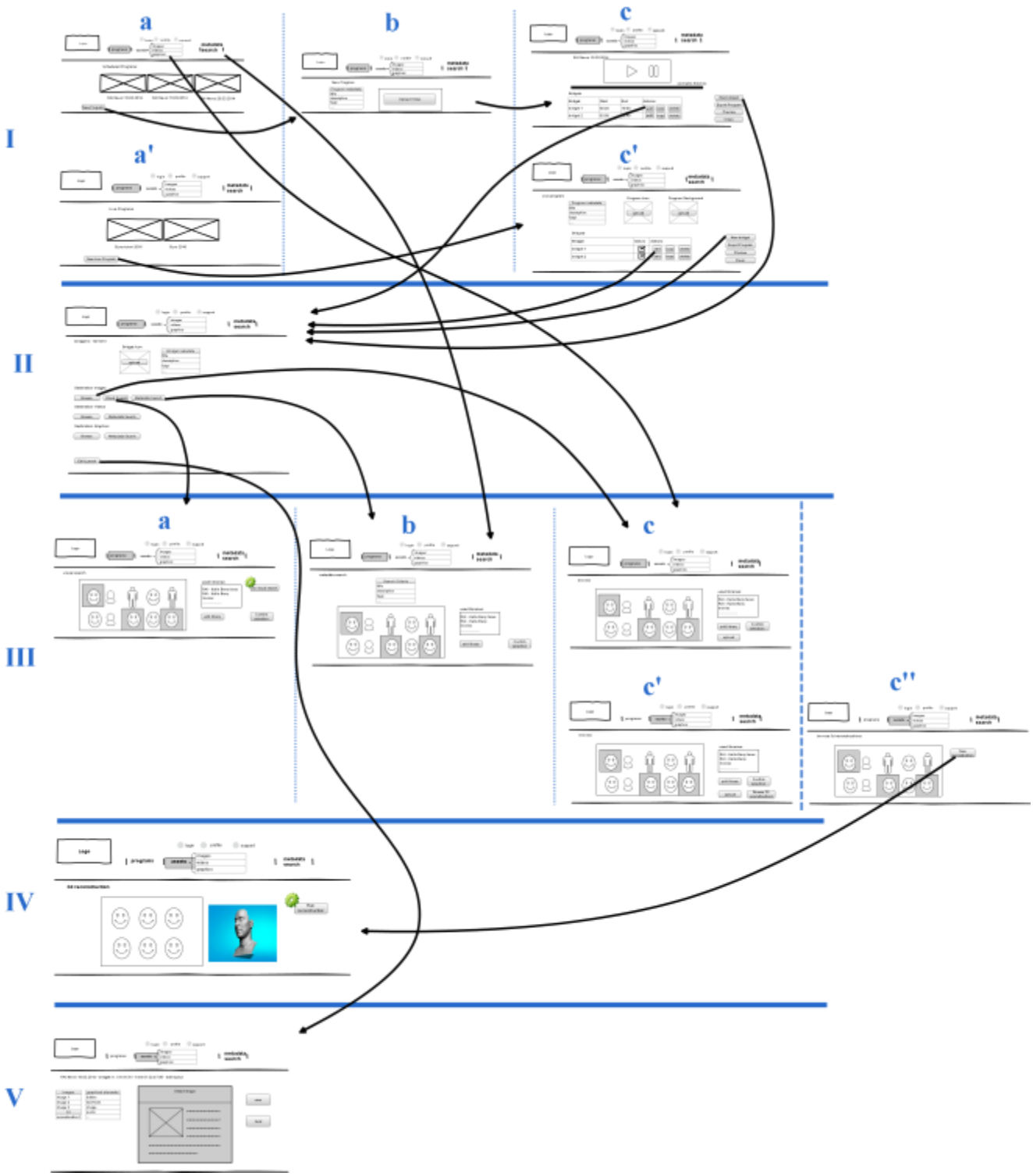


Figure 2: Conceptual design of the AT UI

I. The top level is the "*Programme level*". Here, the professional designer retrieves the entire main media (audio-video) and indications where existing bridgets are positioned (if any, in the case of scheduled programs). He can also select manually a position (frame or shot) that he wants to enrich or enable and disable live bridgets in the case of a live broadcast. Several functionalities are supported at this layer, through different interfaces:

a *Programme overview*:

- The professional designer is presented with all the available Scheduled Programmes, (that are accessible based on user permissions) (see Figure 3, interface *I.a*. is presented to professional designer);
- The professional designer is presented with all the available Live Programmes, accessible based on user permissions (see Figure 4, interface *I.a'*. is presented to professional designer);

b *Programme Creation*:

- The professional designer is allowed to load a new video associated to a Programme and specify the Programme's details (broadcast title, broadcast date, broadcast time, etc.) (see Figure 5, interface *I.b*. is presented to professional designer);
- The professional designer is allowed to define a new Live Programme and specify the Programme's details (broadcast title, broadcast date, broadcast time, etc.) (see Figure 6, interface *I.b'*. is presented to professional designer);

c The professional designer has the option to enrich one Programme. The *Programme editing* interface

- Presents the professional designer the Programme's video and the corresponding timeline containing references to pre-existing bridgets;
- Allows the professional designer to select one of the pre-existing bridgets for editing;
- Allows the professional designer to pause the video, and select a frame or a segment for enrichment – and then add a new bridget (see Figure 7, interface I.c);
- Allows the professional designer to add a new live bridget and enable or disable live bridgets to be broadcasted (see Figure 8, interface I.c');
- Allows the professional designer to edit the Programme's details (broadcast title, broadcast date, broadcast time, etc.).

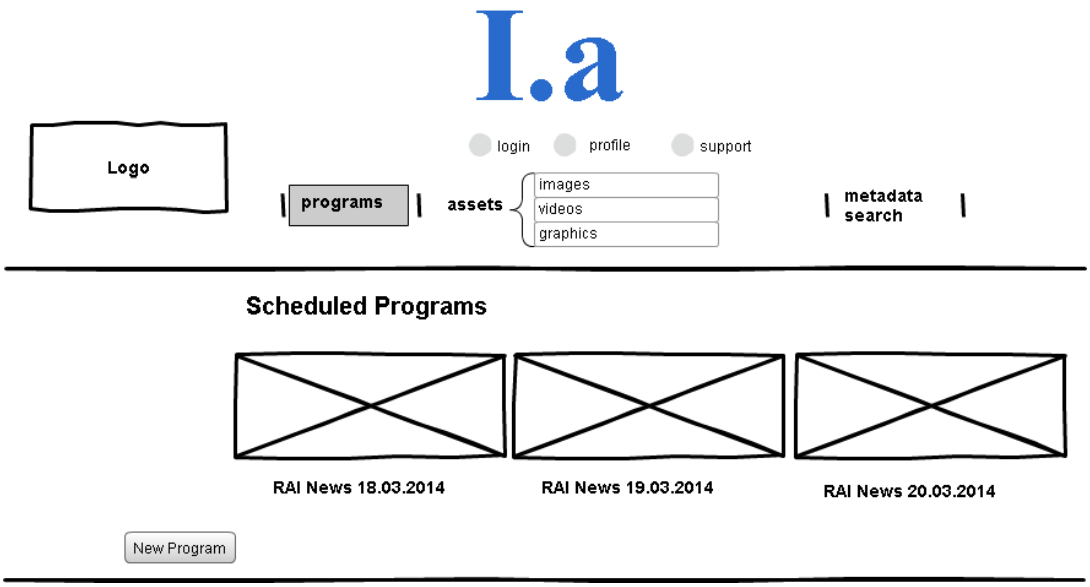


Figure 3: Conceptual design of the AT UI for Scheduled Programmes overview

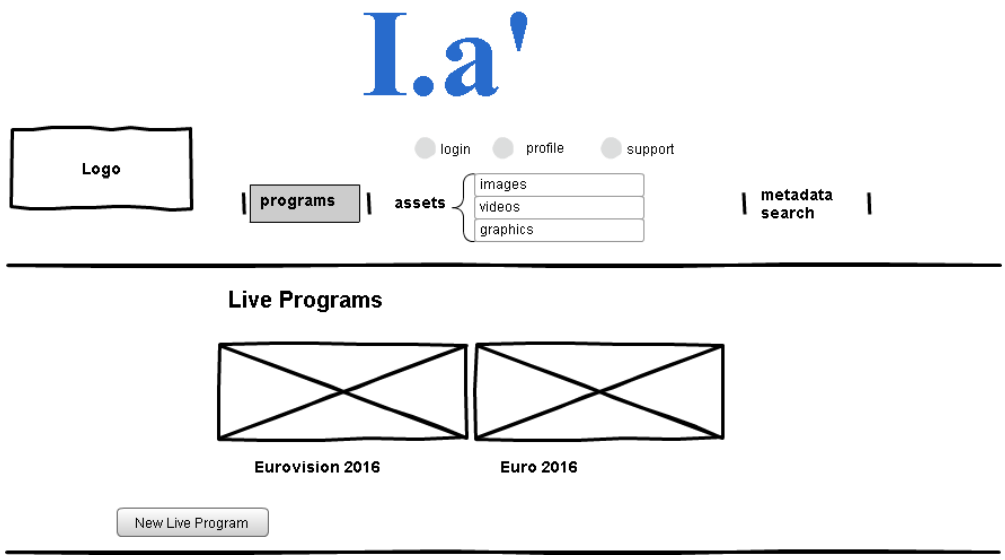


Figure 4: Conceptual design of the AT UI for Live Programmes overview

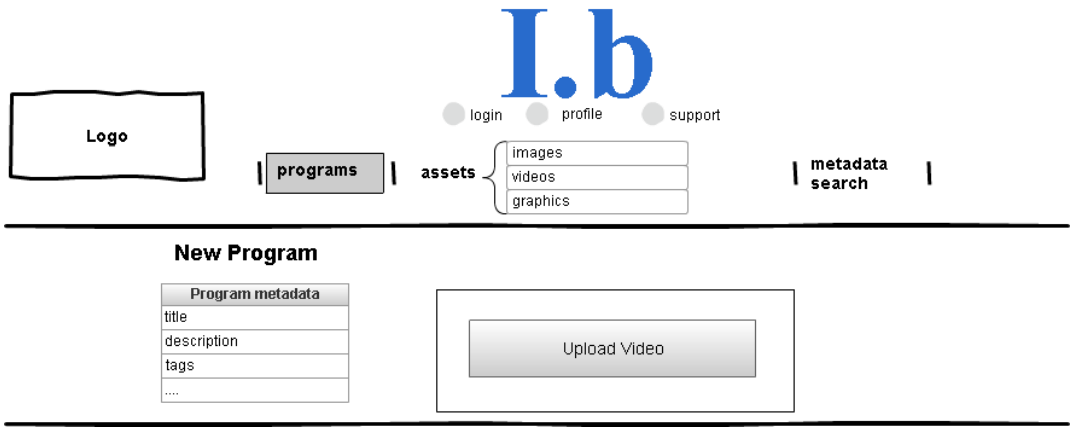


Figure 5: Conceptual design of the AT UI for Scheduled Programme creation

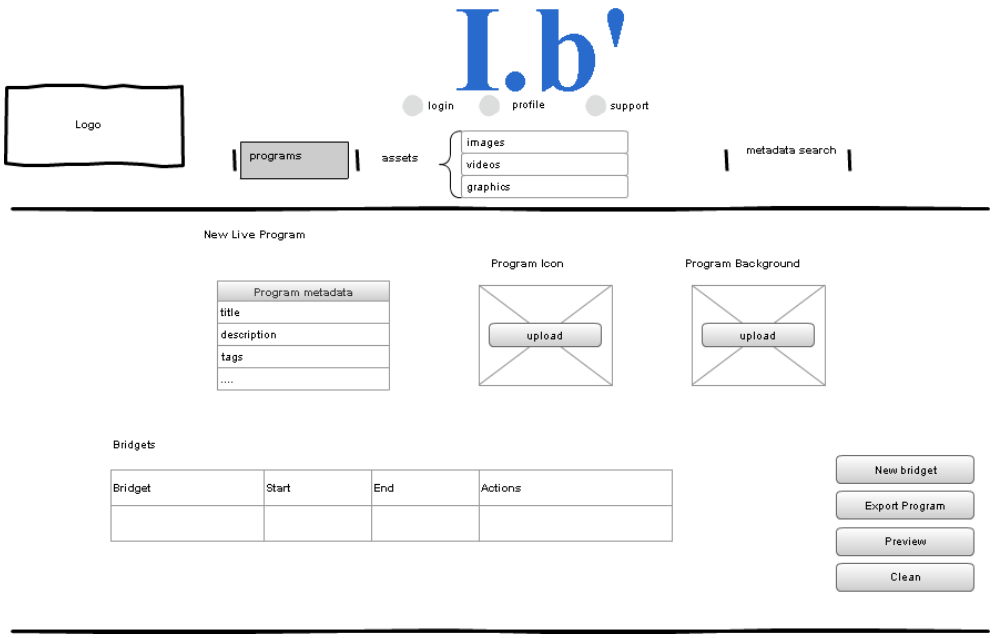


Figure 6: Conceptual design of the AT UI for Live Programme creation

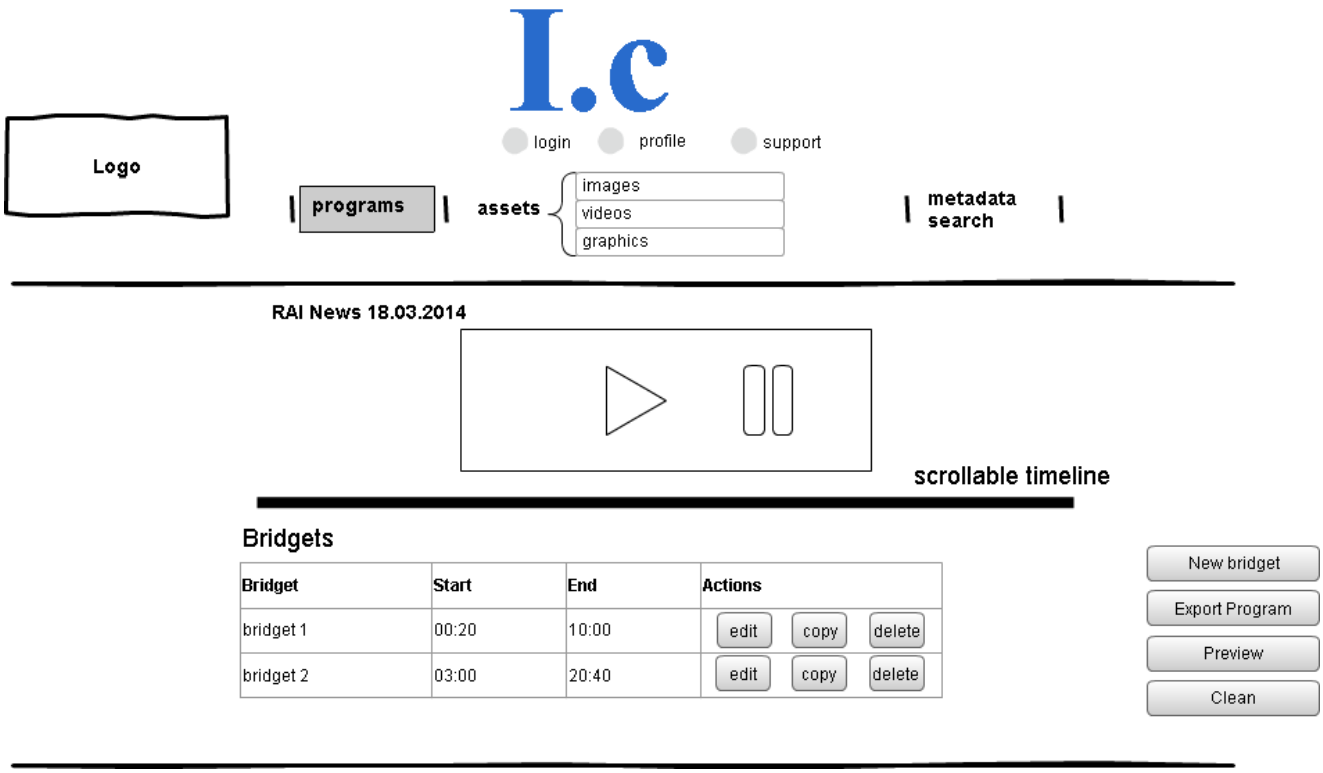


Figure 7: Conceptual design of the AT UI for Scheduled Programme editing

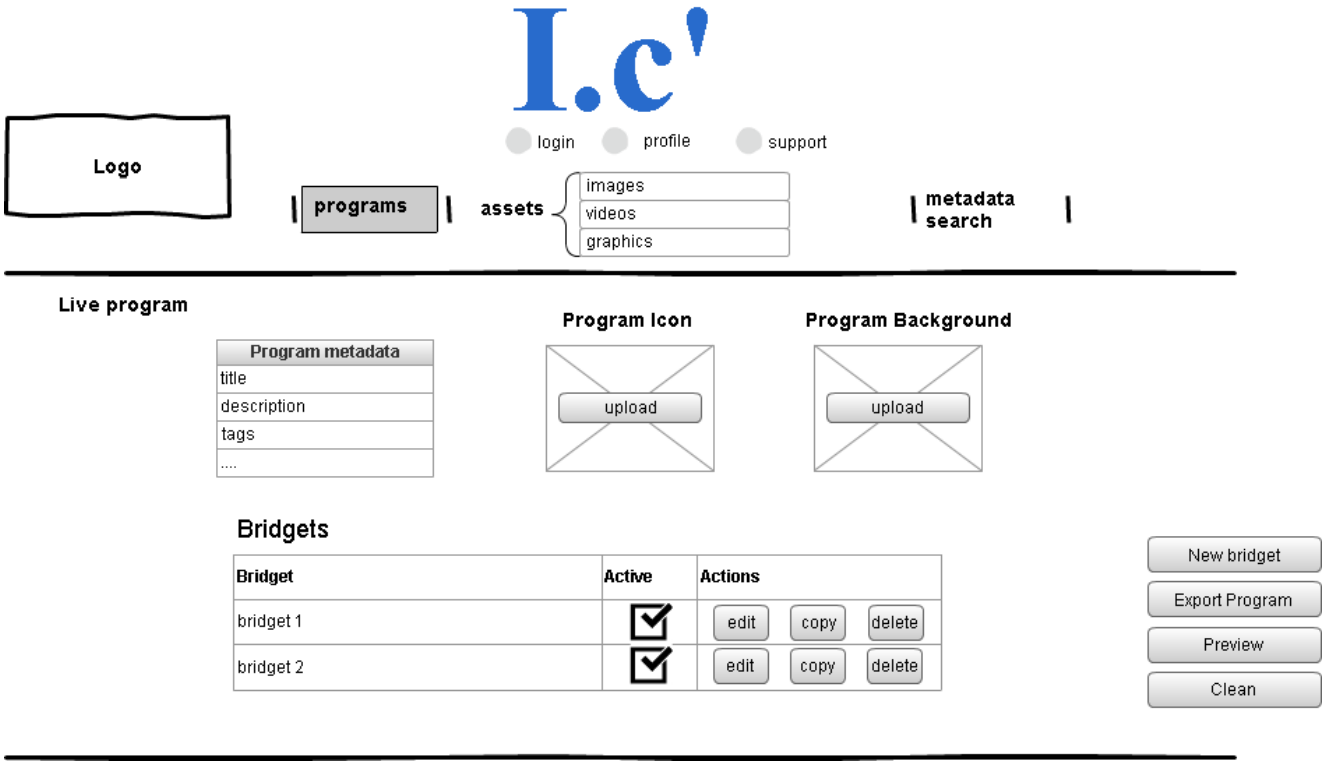


Figure 8: Conceptual design of the AT UI for Live Programme editing

II. At the second level, the "*Bridget Overview*" level, all the information related to a particular bridget is presented. It is possible here to create a new bridget or to edit an existent one.

a *Editing a bridget* (see Figure 9Figure 9, interface *II.* is presented to professional designer):

- The professional designer is presented with the corresponding segment having already selected the region used for enrichment (in the case of a scheduled programme);
- The professional designer is presented with the associated media (i.e. the bridget destination content) already used for enriching the selected region (images, 3D objects). The professional designer has also the possibility to browse through this associated media by category and search by metadata;
- The professional designer has the option of running another visual search based on the selected region from the current segment;
- The professional designer has the option of editing the bridget's presentation layout (the information displayed to the user when selecting the bridget from the Bridget Player in the end user's BRIDGET Application).

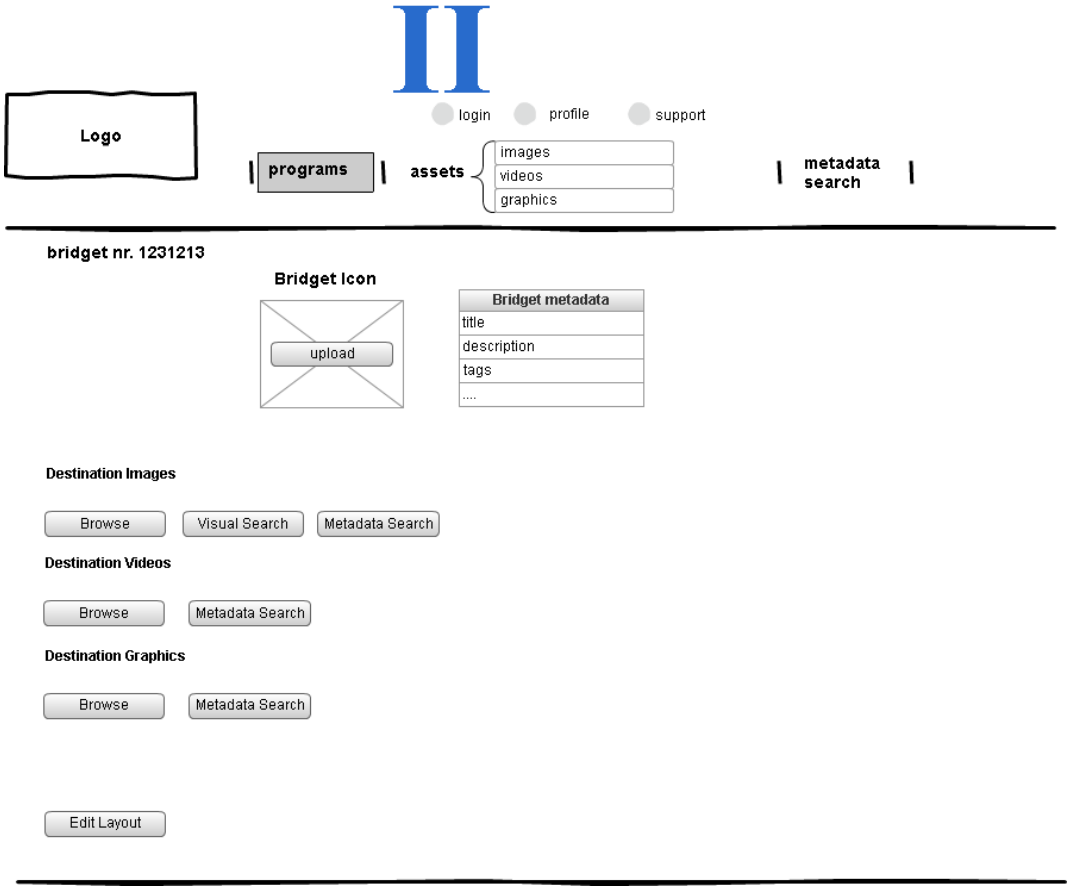


Figure 9: Conceptual design of the AT UI for editing an existing bridget

III. The third level, the "*Enrichment*" is reached when defining destination content for a bridget by browsing the asset repository using the different search options available.

a *Visual search* (see Figure 10, interface *III.a*):

- The professional designer is presented with already matched images obtained from visual search. These images are a result of previous visual searches that match at least partially the time segment.
- The professional designer is presented with the list of already selected images obtained from visual search;
- The professional designer has the option of re-running the visual search in order to obtain more results
- The professional designer has the option of adding or removing repositories for visual search and re-running the visual search on the new set of repositories; The professional designer has the option to save the images as destination content of the bridget.

b Metadata search (see Figure 11, interface III.b):

- The professional designer has the option to access this functionality also from the main AT menu
- The professional designer is presented with the list of already selected assets
- The professional designer is presented with several input fields used for metadata searching such as title, description and associated tags

- The professional designer has the option to save the assets as destination content of the bridget.

c Browse:

- The professional user has the option to access this functionality also from the main AT menu
- The professional designer is presented with the list of already selected assets (see Figure 12, interface III.c) depending on the asset type selected
- The professional designer is presented with the complete list of assets of the selected type
- The professional designer has the option to save the assets as destination content of the bridget.
- When browsing 3D assets, the professional designer has the option to view all the reconstructions created on the system (see Figure 13, interface III.c')
- When browsing 3D reconstructions, the professional designer has the option of creating a new 3D reconstruction (see Figure 14, interface III.c'')

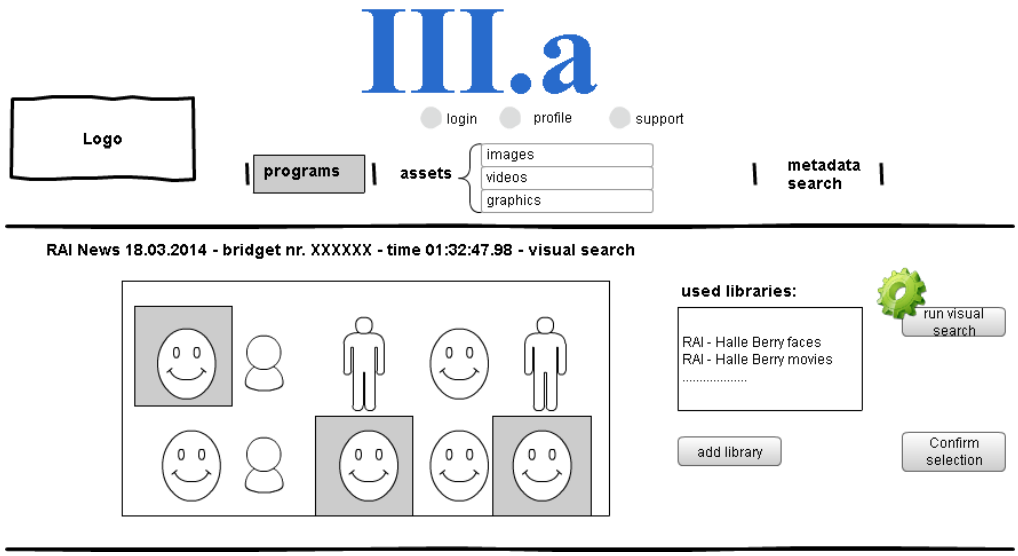


Figure 10: Conceptual design of the AT UI for visual search enrichment

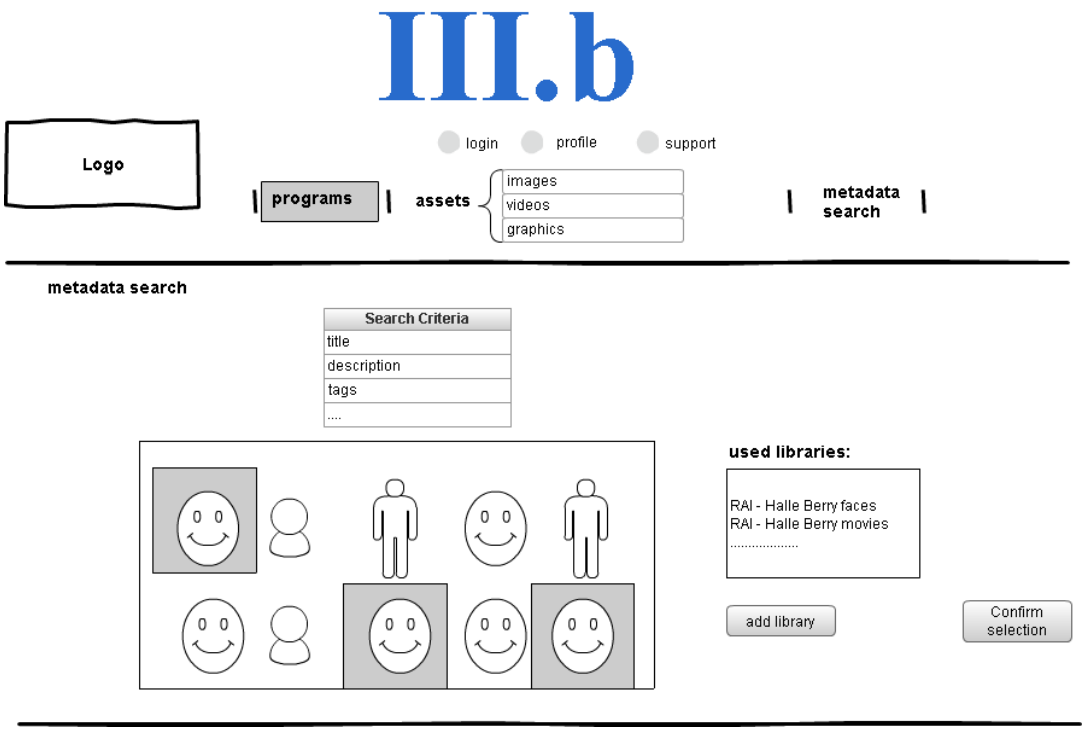


Figure 11: Conceptual design of the AT UI for Metadata search

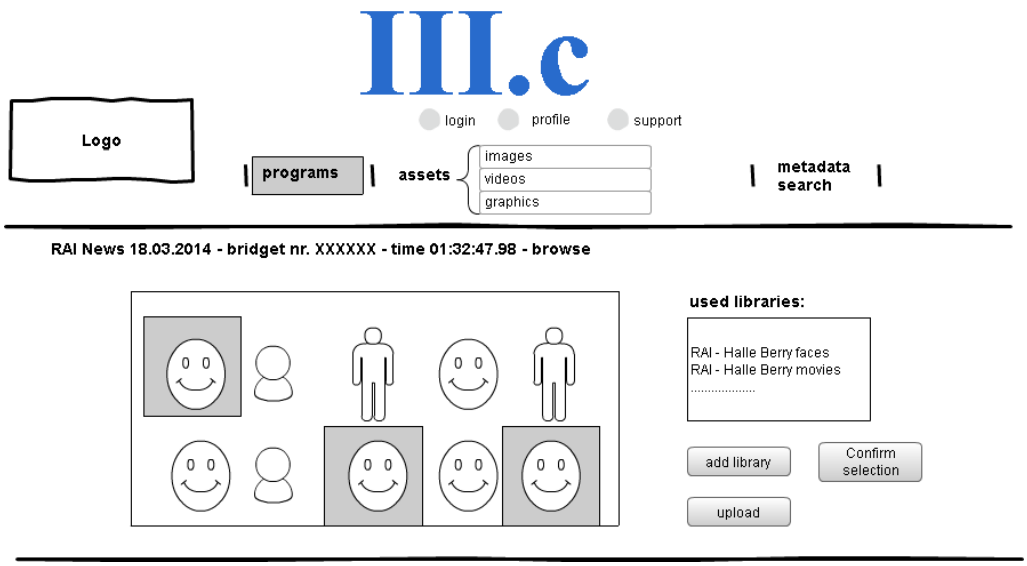


Figure 12: Conceptual design of the AT UI for browsing assets

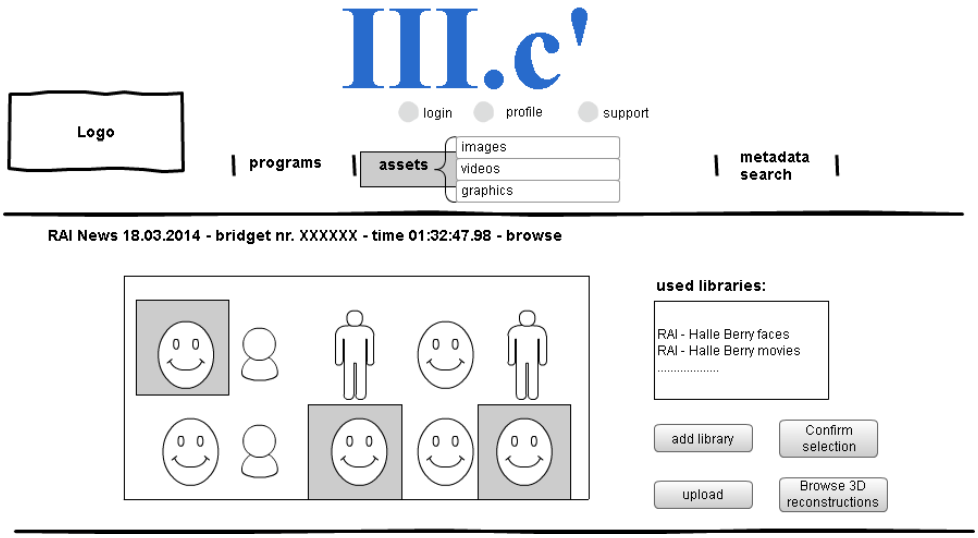


Figure 13: Conceptual design of the AT UI for browsing 3D assets

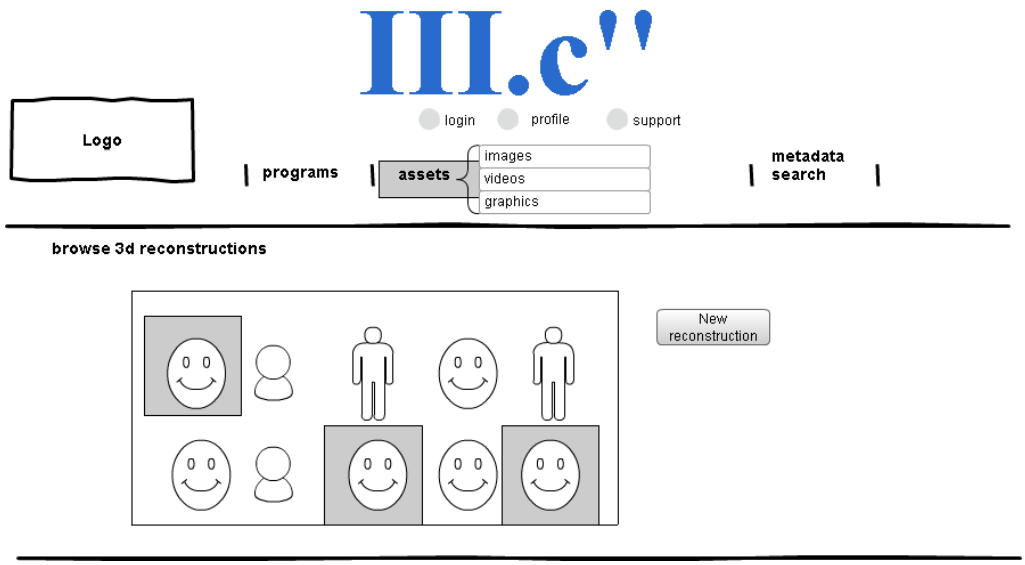


Figure 14: Conceptual design of the AT UI for browsing 3D reconstructions

IV. The fourth level "3D reconstruction" (illustrated in Figure 15, interface IV)

- The professional designer is able to access this functionality from either the AT's main menu, or by browsing Graphic Assets when assigning destination content for a bridget
- The professional designer is able to upload or modify a dataset used for the re-construction
- The professional designer is able to select options for the reconstruction (eg. which 3D reconstruction engine to use)
- The professional designer is able to visualize the result of the 3D reconstruction in several formats (depending on what type of reconstruction has been selected)

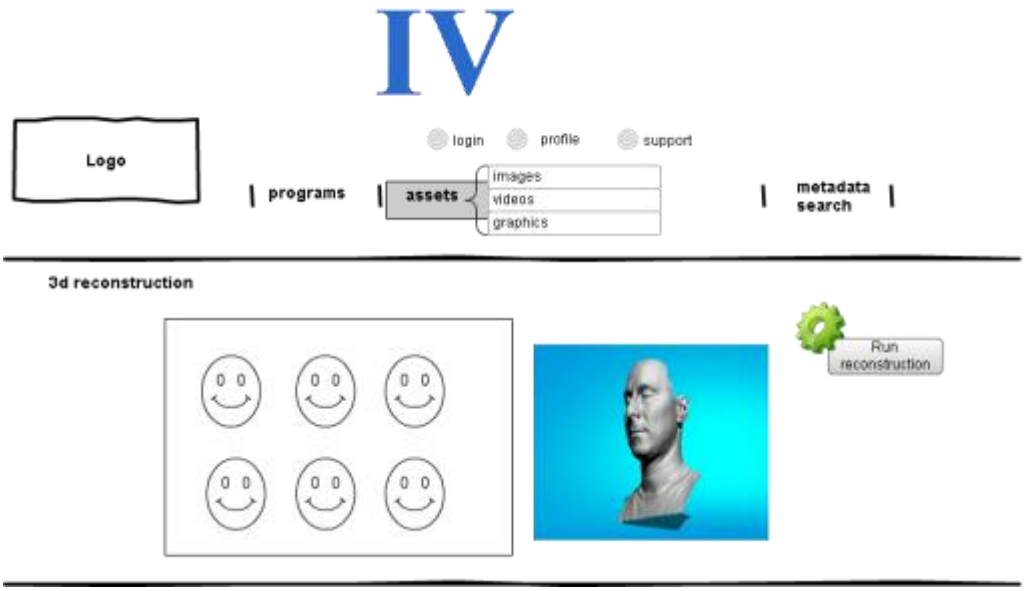


Figure 15: Conceptual design of the AT UI for 3D Reconstruction

V. The fifth level "*Bridget presentation layout editor*" (illustrated in Figure 16, interface *IV*) is the one allowing to edit the layout of a bridget (selecting the interactive behaviour of the bridget as well as the spatial of the media elements composing the bridget).

- The professional designer is presented with a special area which allows manipulation (drag and drop, resize, etc...) of images, 3D objects, buttons and text fields;
- The professional designer is presented with a list of associated images and 3D objects obtained from visual search and 3D reconstruction;
- The professional designer is presented with a list of widgets that can be added to the layout: button, text field, image, audio, etc.

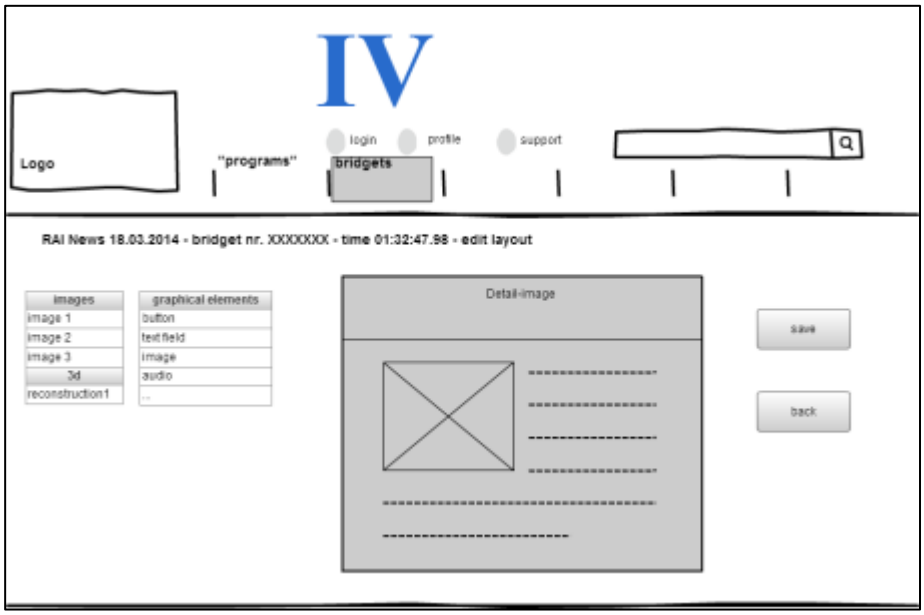


Figure 16: Conceptual design of the AT UI for editing the bridget presentation layout

The activity on this task is continuing and in collaboration with T7.2, whose objective is to implement, test and validate these concepts with professional content producers.

Snapshots of the implemented AT UIs are displayed in section 2.2.

2.2 User Interface Implementation

The design guidelines summarized in the previous section were used to enrich the consumption experience with second screen content for four Bridget use cases, representing various kinds of programs: news, a documentary on Torino architecture, a TV talk show about the Concordia accident, and a TV entertainment show. These programs in combination with the related UI design will be introduced in more detail in the following sub-sections. While these programs were made available in the first period of the project, their live counterpart (the AT dashboard and the corresponding communication between the player and the dashboard) was carried out in the second period.

Figure 17: Current RAI programs selected to BRIDGET experiments

For each program, a detailed analysis of the content was performed, interesting (temporal) hot points were detected and related content was collected from various sources. Figure 18 shows this enrichment process for the News program.

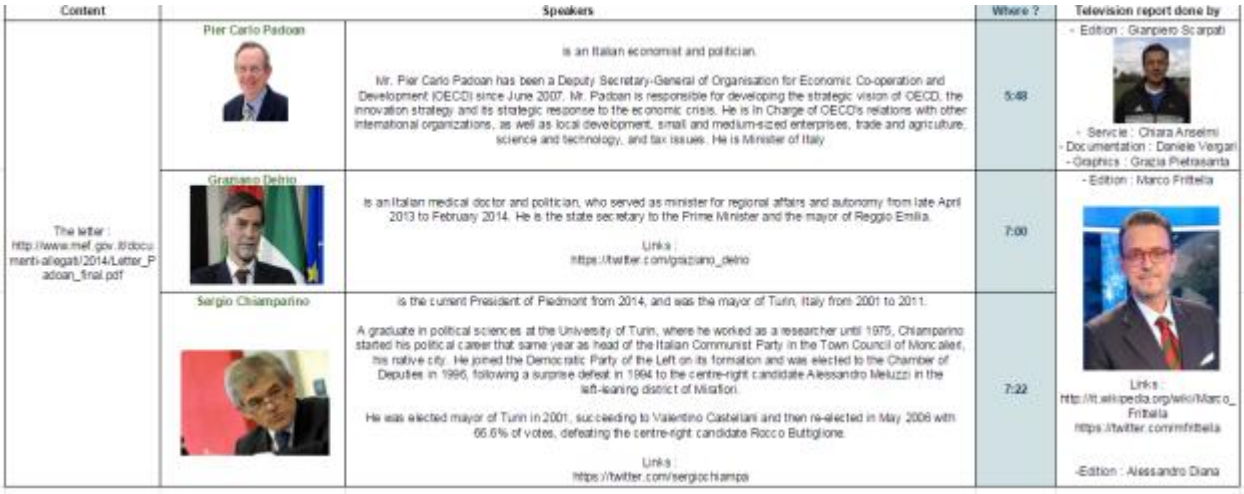


Figure 18: Identification of the hot spots and collection of relevant content

The implementation of this design was carried out in 2 stages: first, an initial UI design was created as shown in Figure 19Figure 19. This phase focused more on enabling the features and functionalities of the UI on the BRIDGET Player.

Figure 19: Illustrations of the first UI design as presented on the tablet for one of the RAI programs

The second phase of the UI design concentrated on the presentation of such functionalities on the second screen, in order to optimize the user experience. Therefore, three different scenarios were elaborated for three of the RAI programs: news, a documentary on Torino architecture and a TV entertainment show. The UI design was centred not only on the content itself, but also on the targeted audience.

Let us note that the UI for the TV talk show "Porta a Porta" was not redesigned, since the first version of the UI was found to be suitable for the presented content.

2.2.1 News UI

Figure 20Figure 20 shows the current version of the UI design for the news report. The main difference from the first version of the UI consists in the fact that, in the later version, each bridget is providing enrichment to only one of the subjects discussed in the news report. The user can browse at any time the summary of the show, in order to see the list of the discussed subjects. Users don't have access to all the bridgets from the beginning, but only when such bridgets become "available", i.e. when the related subject is presented in the show (following the principles described in Section 2.1).

Figure 20: Illustrations of the second UI design as presented on the tablet for the news report

2.2.2 Documentary UI

Figure 21Figure 21 presents the current version of the UI for the documentary on Torino architecture. In this case, as the documentary is presenting the landmarks of a city, the entire UI design was conceived around the map of the city. Each time a bridget becomes available, i.e. a new landmark is presented in the documentary, it is positioned on the map. A new feature implemented within this design is the possibility to view the restaurants and hotels situated in the landmark's neighbourhood. At the end, the user is presented with a small quiz related to the presented content.

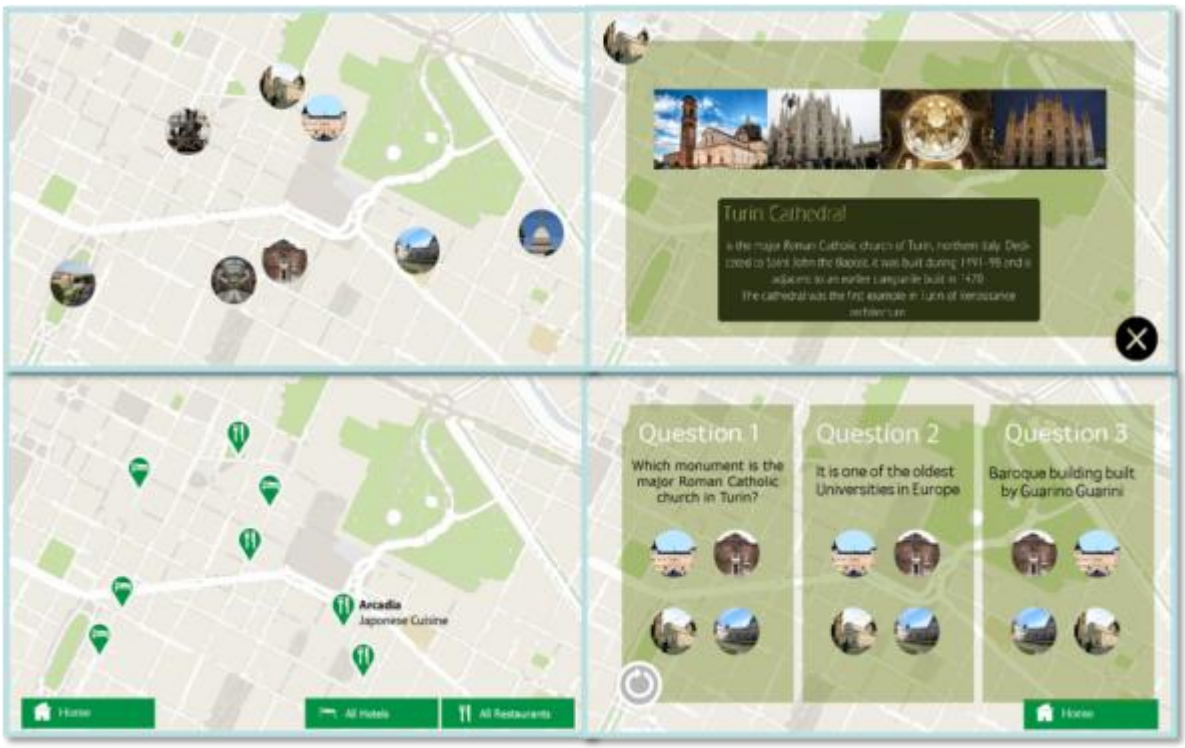


Figure 21: Illustrations of the second UI design as presented on the tablet for a documentary on Torino architecture

To validate the 3D algorithms, we integrated the results of the 3D reconstruction for several buildings and monuments from the city of Turin: Arco monumentale all'arma di artiglieria, Parco Valentino (obtained from 286 pictures), Villa Sartirana, Parco della Tesoriera (obtained from 181 pictures) and Palazzo Carignano facade, Piazza Carignano (obtained from 165 pictures). The reconstructed 3D objects are displayed in full screen and the end-user can chose the view point as illustrated in Figure 22.

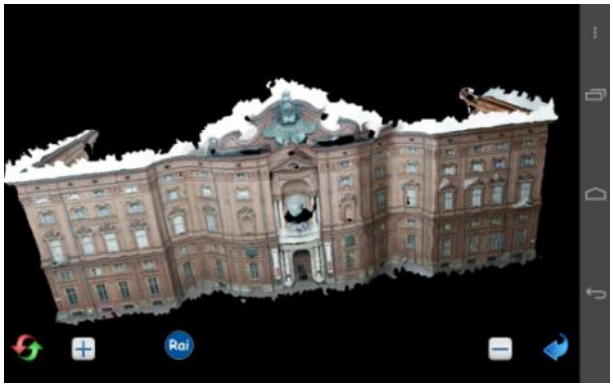


Figure 22: Visualisation of a 3D reconstructed object

In the second part of the project, we also addressed two user-related aspects: creating new content by the end-user (what we initially called mini AT) and sharing on social networks. The first aspect was addressed by allowing the end-user to upload his/her own content as a bridget destination. That is, for each bridget (and if the professional content designer activate it), it is possible to transform it into editable bridget by exposing an "edit" icon as shown in Figure 23.

Figure 23: Edit icon and upload user generated content

In such case, the image proposed by the end-user is locally used to change the bridget. It is also possible to upload this image on the server, and after being accepted by the professional content creator, the new bridget can be broadcasted to other users.

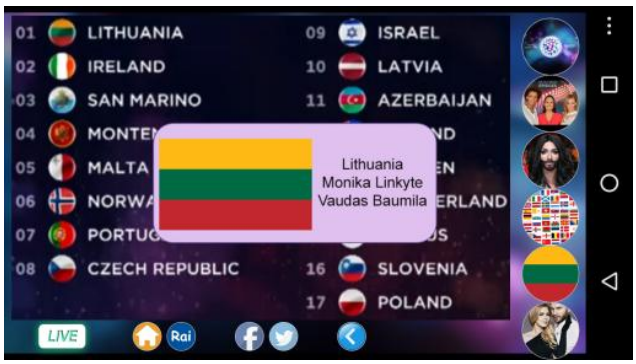
2.2.3 Entertainment Show UI

Figure 24 presents the current version of the UI design for the TV entertainment show. Since the target audience of this show is teenage girls, the features and design were focused on the presented subjects, i.e. street style; make up, look and hair styles. At the end of the program, the user also is presented with a dressing game. The game consists of creating different outfits for a character by combining garments from the proposed gallery. The game provides a feedback for each created outfit regarding the user's skills in matching different clothes (e.g. "The shoes don't match", "Try again", "This is amazing!" It's so fancy! "Love the colours" etc).

Figure 24: Illustrations of the second UI design as presented on the tablet for a TV entertainment show

2.2.4 Eurovision content

Figure 25 presents the current version of the UI for the Eurovision contest, considered in the BRIDGET project as a live program. As the main concept of the contest is the country representativeness, the entire UI design was conceived around the illustrations of countries and the artists for each one. The moment when a song will be transmitted is not known in advance, therefore the content creator will author only the spatial properties of the bridget, its temporal behaviour will be controlled by the dashboard. In particular, the following bridgets with enrichments about different portions of the main clip were designed (info and pictures about the ESC in general, info and pictures about the three female presenters, info and pictures about Conchita Wurst, names of singers belonging to different participant nations, info and pictures about Lithuania, info and pictures about Lithuanian singers).



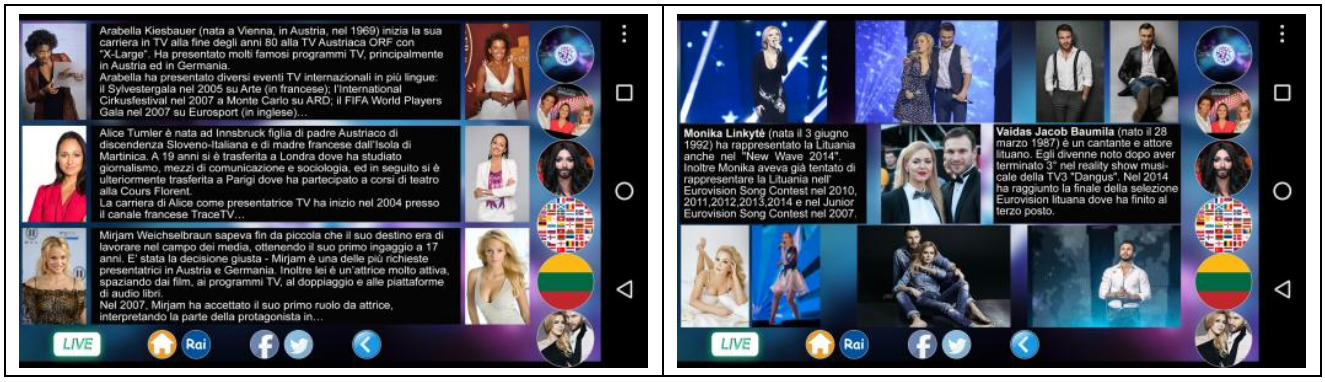


Figure 25 - Screenshots indicating navigation in a live Bridget program.

2.3 Content used to validate 3D algorithms

The content used to validate the 3D tools integration on the Bridget platform consist of several datasets representing buildings or monuments captured in Turin, Lisbon or Munich. The dataset acquisition protocol was followed for each type of 3D reconstruction tool. The image datasets were acquired using cameras that are able to save the camera information (exif information). In the figures below, we present a few datasets that were used during the testing of the tools and their results: Palazzo Carignano (Figure 26), Arco Valentino (Figure 27), Torre de Belém (Figure 28), "The piglet"(italian, "il Porcellino") from the Italian artist Pietro Tacca (Figure 29) and "The lion" sculpture (Figure 30).

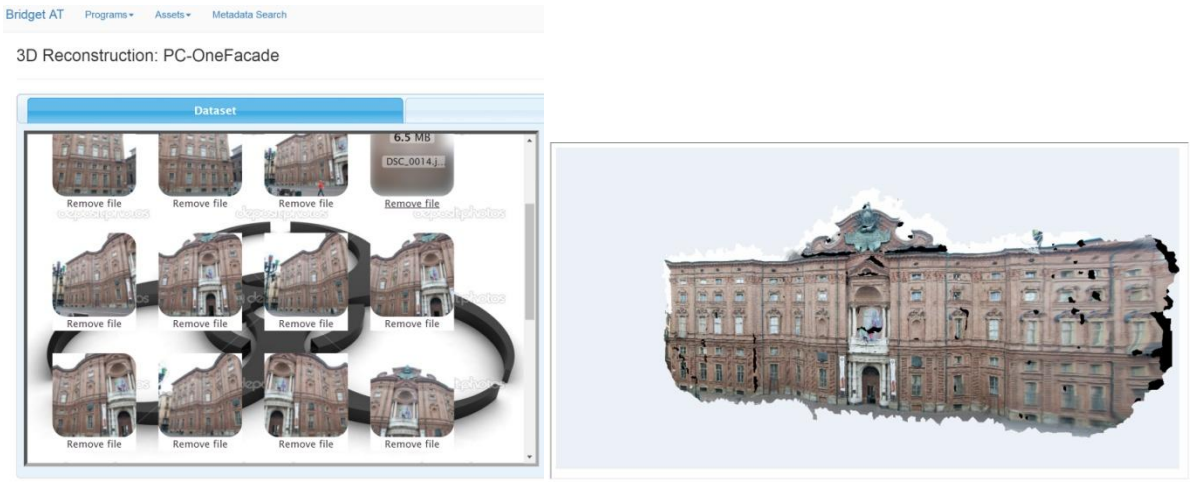


Figure 26: Palazzo Carignano – 3D reconstruction dataset and results

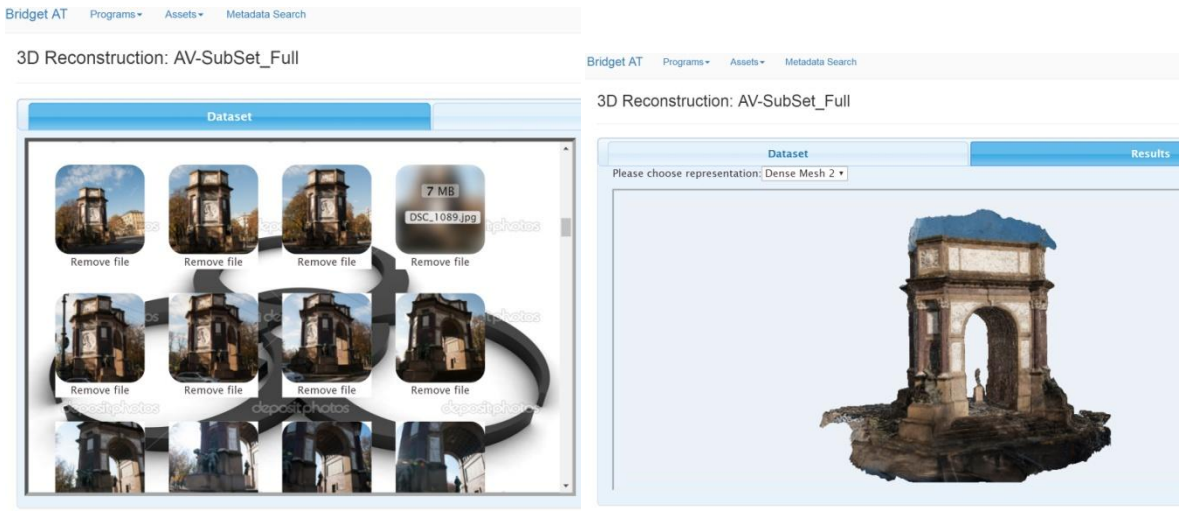


Figure 27: Arco Valentino – 3D reconstruction dataset and results

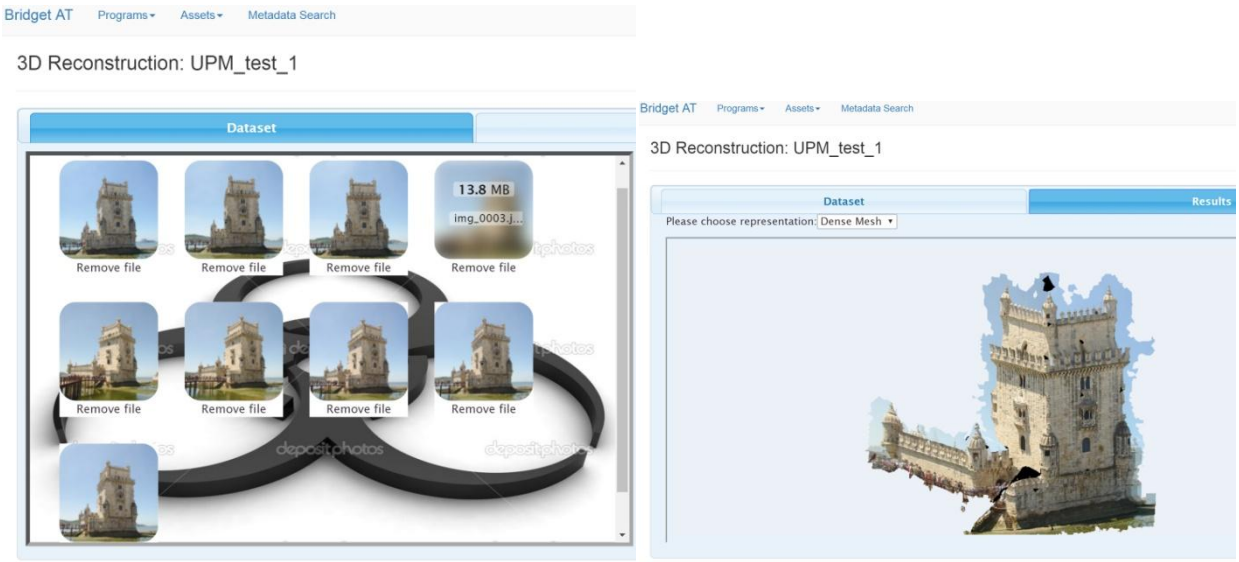


Figure 28: Torre de Belem – 3D reconstruction dataset and results

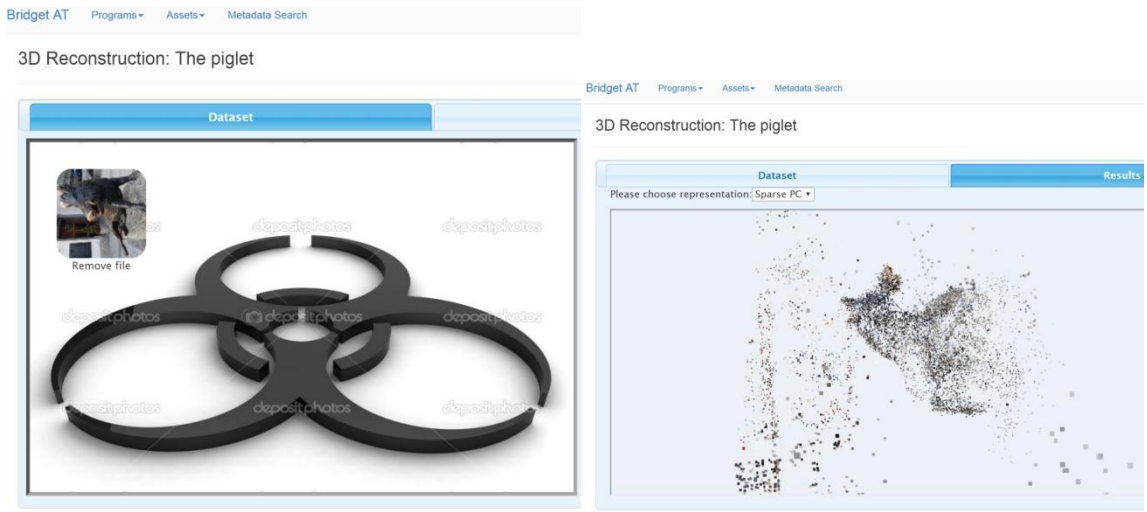


Figure 29: The Piglet – 3D reconstruction dataset and results

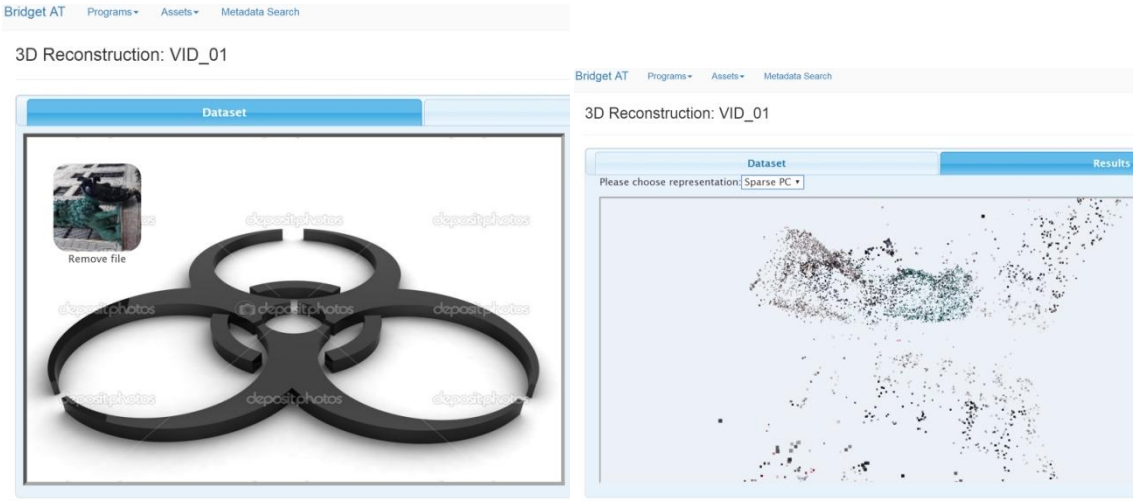


Figure 30: The Lion – 3D reconstruction dataset and results

3 T7.2 - Authoring Tools

This Task, which started in M5 of the project, has two objectives: *i)* to design and implement full-featured studio version AT meant for professional broadcasters (*professional*) and content providers, *ii)* to design and implement a simplified (*mini*) AT usable by consumers on end-users terminals. During the first 18 months of the project, the activity was focused on the *professional AT*, and this activity continued in the last part of the project Implementation of the *mini AT* was done by enabling end-users to add their own bridget destination content.

Section 3.1 describes the overall architecture of the developed AT; more details about the frontend and the backend systems are provided respectively in 3.2 and 3.3.

3.1 Architecture Design and Implementation

Professional authors goal is to present on the second screen content that has semantic connections with the first screen content: additional information about people, places, events … In order to do so, they can use traditional, text based, search when the semantic concept can be formalized as text. However, the answer to this kind of search is in most of the cases presented also as textual information. While including this classic mechanism in the authoring tool, BRIGET goes beyond this paradigm allowing also visual links performed by using visual search technologies developed in WP5: by using an image as query, the professional author obtains a set of similar images from the repository, images that may be already connected to other metadata (text or other media). He can then chose the ones he want to include in the bridget and select what kind of information will be presented (the media itself, metadata that comes with the media…) and how this will be presented (the layout).

All those functionalities are made available through the *professional AT,* implemented as a web tool based on the MyMultimediaWorld [1] platform. This framework, property of IMT, manages multimedia content and the associated descriptions in a cloud-friendly manner. This task was dedicated to the integration of the novel functionalities developed in WP[4-6] in the form of corresponding plug-ins. Each plug-in corresponds to an engine in the MXM functional architecture designed by WP3 and described in D3.1. Some components were integrated already in the version A of the authoring tool, such as temporal segmenter, visual search, audio fingerprint extraction and matching. The version B of the authoring tool additionally integrated 3D reconstruction engines and the live programs orchestrator (the dashboard), in addition to the improved tools from version A.

The AT has access to additional multi-media repositories, according to the architecture depicted in Figure 31: the content of such repositories (*Program storage*) is previously analysed, in order to automatically extract descriptors uniquely representing such content that are stored on the *Bridget storage*. Therefore, when the media to enrich is uploaded, an automatic processing chain may be started, search-

ing for associations in the repository, collecting the corresponding metadata (*Metadata storage*) and proposing to the content producer several types of enrichments (images, text, videos). This process may be also started for only a specific part of the program.

Finally, the AT packages the assets, UIs, links to external services and resources in a standardised and consistent manner, and those packages are ready for delivery and consumption at the player side. The standard used as a basis for the data format created by the authoring tool is MPEG-A Part 18 (Media linking Application Format), a standard initiated by the BRIDGET partners during the BRIDGET project The activity related to the bridget layout editor was planned in two phases: in the first phase bridget templates are used, and in the second phase a full editor was integrated in the AT.

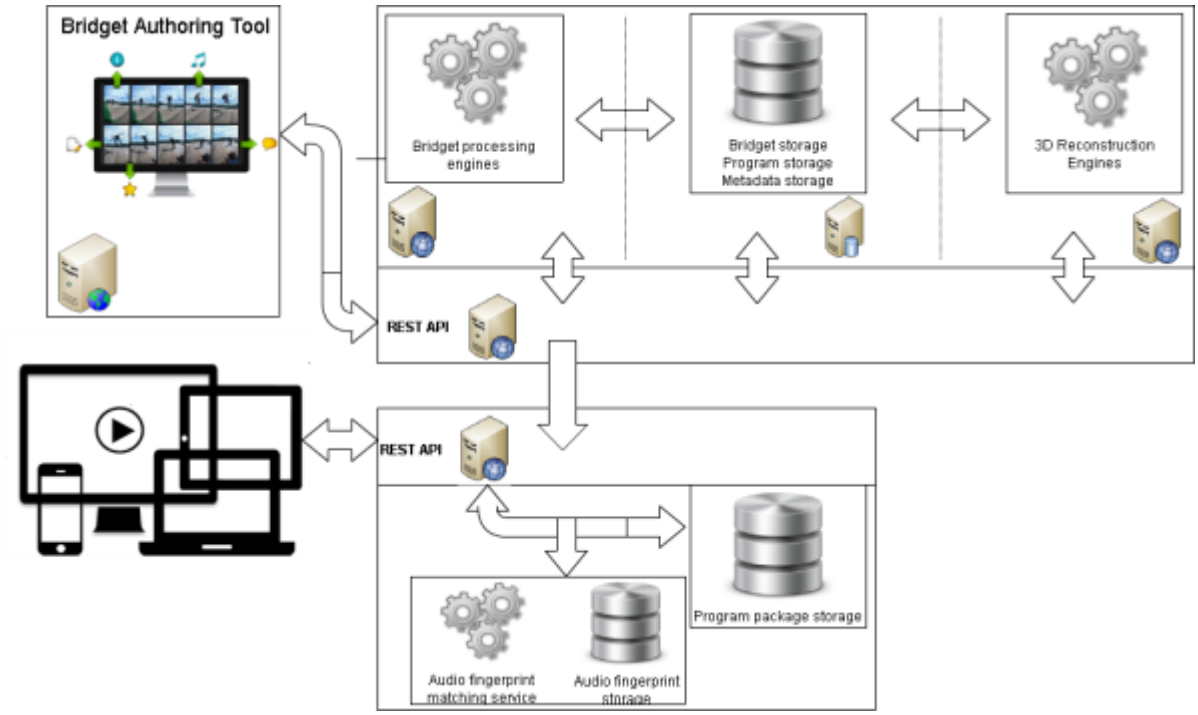


Figure 31: BRIDGET AT frontend and backend architecture.

3.2 Authoring Tool Frontend

The AT frontend is a browser-based web application that allows a professional designer (author) to create and store bridgets [1]. It has been developed as an HTML5 application running entirely in the browser. This allows an easy implementation and efficient logical separation between frontend and backend. It uses REACT.js [2] to exploit efficient client-side templating and redraw on each application state transition. It interacts with the backend using pure HTTP REST APIs.

The development process carried out so far can be summarized in the following breakdown of activities:

- Interaction with RAI experts and professionals to select user requirements and critical aspects of a user interface for the authoring of bridgets, as described in section 2.1 and **Error! Reference source not found.**;
- Translation of user requirements into functional requirement and drafting the user workflow then shared with the partners for agreement. Such requirements are aligned with the general BRIDGET requirements described in deliverable 2.2;
- Analysis of software solution and libraries to support browser client-side application development and creation of rich user interface;
- Definition of a first set of APIs to support the first minimum bridget workflow;

- Creation of interfaces and logic for source and destination content upload and description (metadata);
- Creation of user interfaces and application logic to view enriched multimedia contents with associated bridgets on the timeline;
- Addition of advanced UI functionalities for:
  - Enabling user selection of time segments in the multimedia player;
  - Enabling the user definition of the design to be used inside the scene.
- Creation of the user workflow to:
  - Register/Login/Logout/Unregister;
  - Create and remove bridgets;
  - Enable or disable live bridgets
  - Edit metadata;
  - Run visual searches and select results;
  - Run 3D Reconstructions.

As an example, the Figures below demonstrate how some of the BRIDGET AT functionalities can be accessed.

Figure 32: Video is played to find source content for a bridget

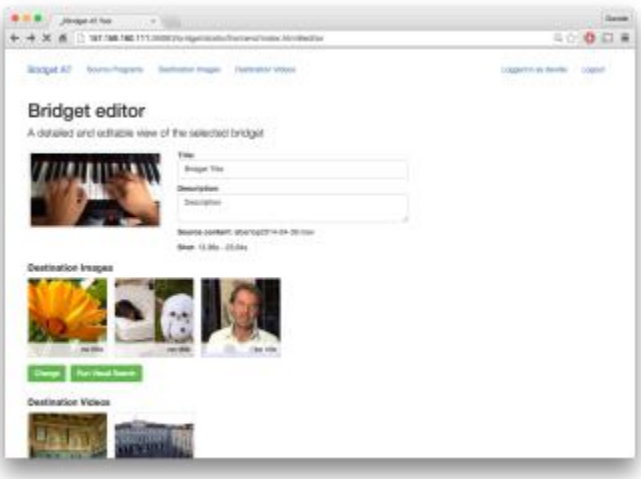


Figure 33: A bridget is created

Figure 34: Images suggested by CDVS search

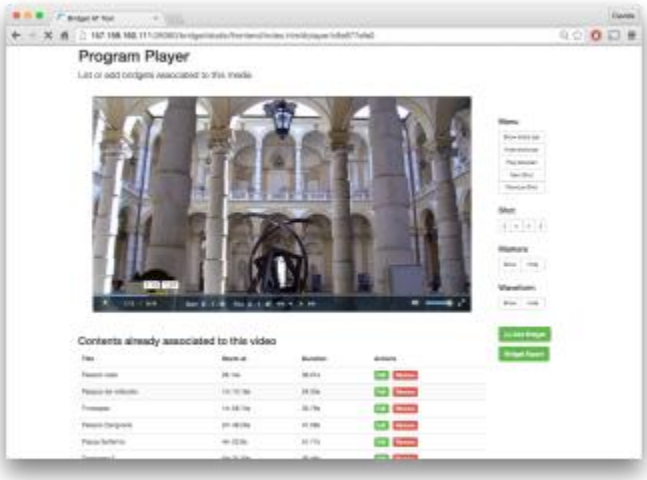


Figure 35: Viewing all bridgets in a program

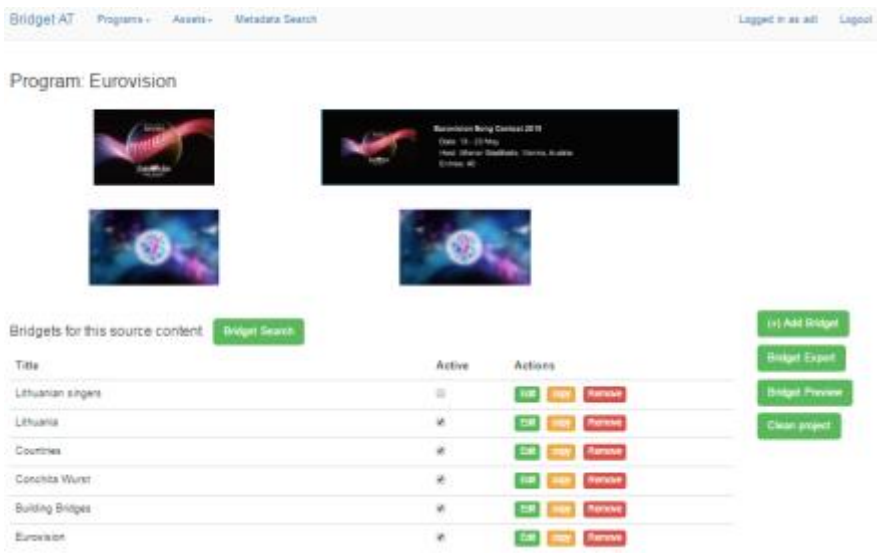


Figure 36: Live Programme Dashboard

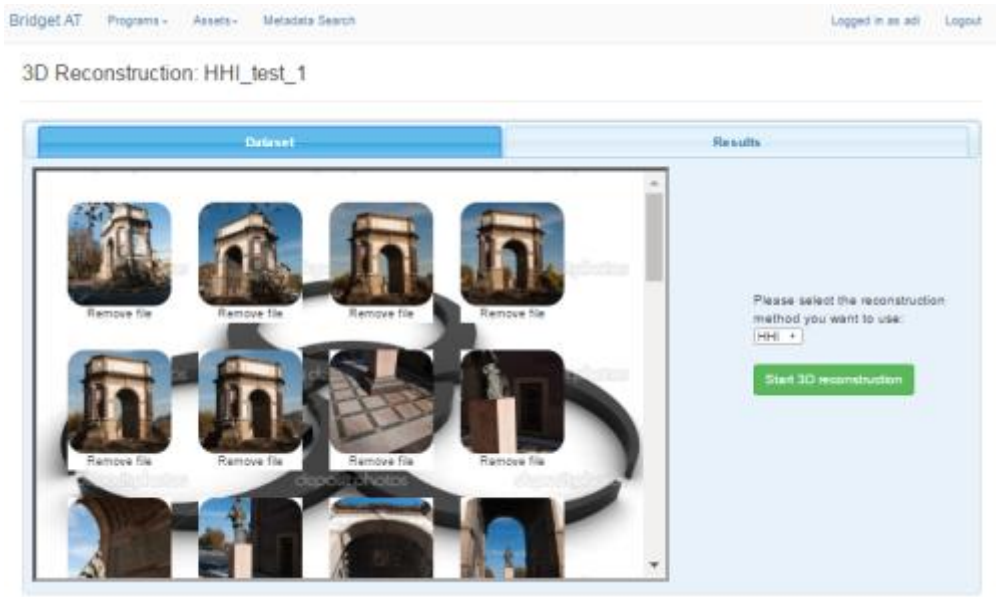


Figure 37: 3D Reconstruction – dataset

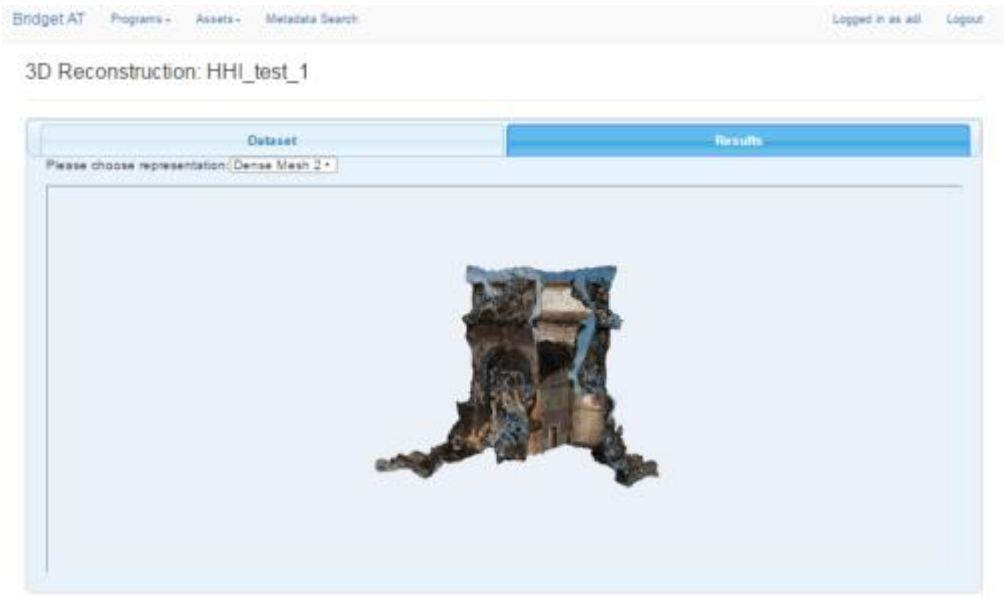


Figure 38: 3D Reconstruction - result visualization

Figure 39: Layout Editor

3.3 Authoring Tool Backend

During the first half of the project, the integrated Bridget Engines were focused on preparing and processing of the program, namely the source video content. These engines produce results that enable the Authoring Tool to better handle destination content and to create bridgets. Because the back-end was designed and developed in a modular manner, it allows the addition of new engines to the already existing structure without altering the processing flow.

The actions taken in the second half of the project were focused on the integration of the 3D reconstruction tools. The particularity of the tools is that in the processing chain they make use of the GPU's parallelization capabilities, thus introducing the necessity of GPU ready nodes in the already existing cluster structure. We built and updated the cluster definition, resulting in a hybrid structure to support native calls to the GPU and added NVidia CUDA support (Figure 40). Also, a new definition of processing routes was developed to define the processing nodes selection based on the tool's particularities.

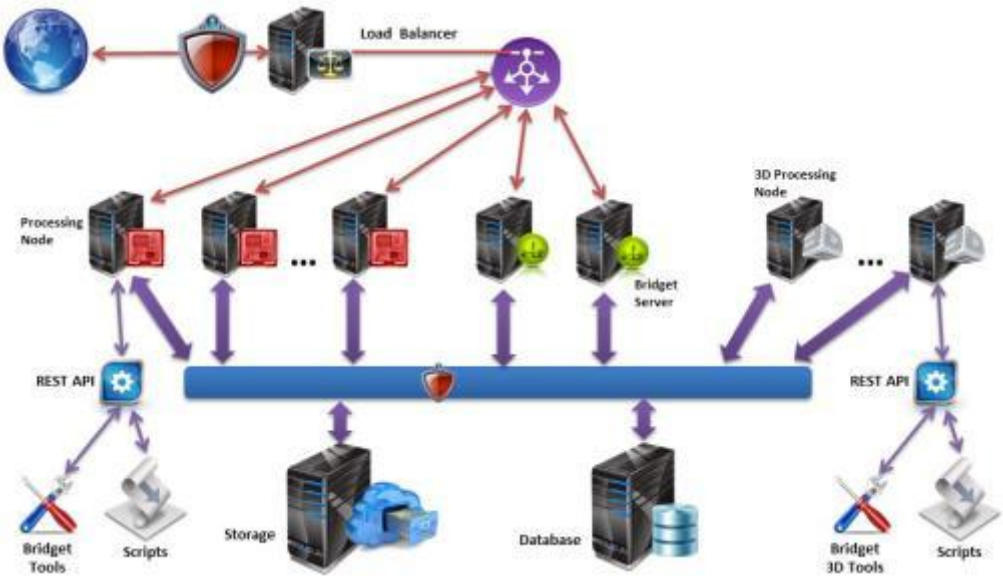


Figure 40: Updated cluster structure – 3D processing nodes

As for the previously integrated engines, we defined hierarchical processing calls by splitting the flow into sequential or parallel tasks, decreasing the processing time and making possible the control of each step of the processing flow. As shown in Figure 41, the processing is routed based on the author's tools selection and the output results differ depending on the selected tools.

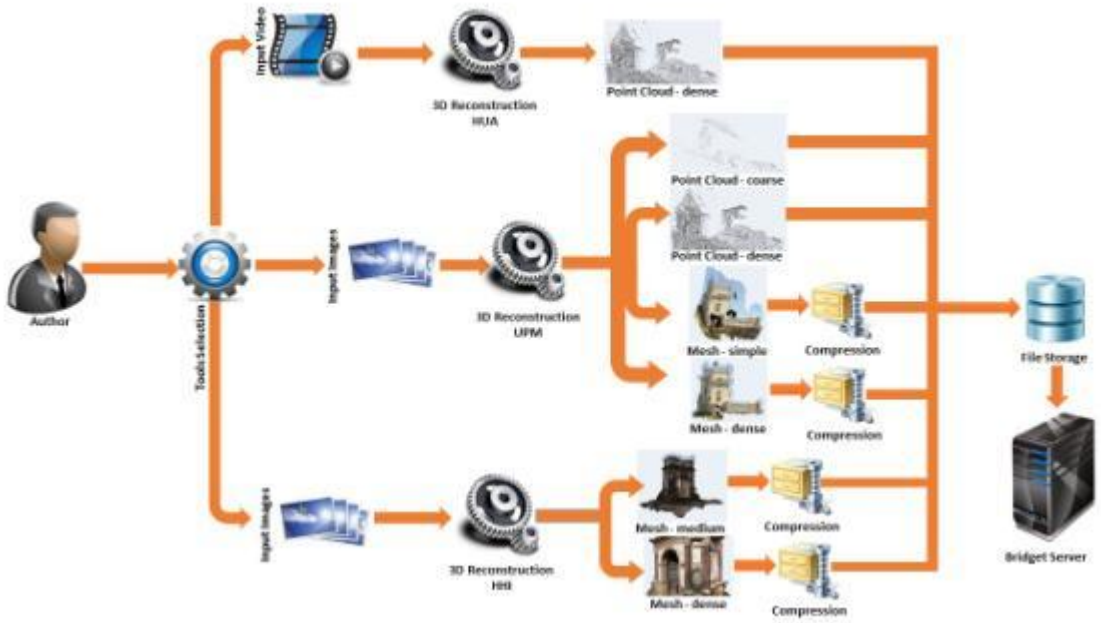


Figure 41: Block architecture of the 3D Reconstruction Authoring Tool Backend

Taking into account that the entire Authoring tool interface is web-based, additional post-processing modules were developed to provide web visualization of the 3D reconstruction results, such as: web player for the point cloud and textured mesh results. These features, allow the author to analyse and visualize the results without installing any additional software packages.

The development of the AT backend has also taken advantage of the original design of the system and by accessing the APIs already available or by creating new APIs that integrate with the original system, there has been minimal change in what was delivered in version A, described in D7.1: BRIDGET Authoring Tools and Player –Report – Version A.

The following components have been integrated:

Proto Engine	Overall Description	Current AT integration status	Source
3D Reconstruction	Creates a 3D model from input images and/or videos	*Fully integrated*	WP6
3D Compression	Produces a compressed 3D graphics data structure from a 3D model that can be efficiently transmitted, and later decompressed and rendered to screen	*Fully integrated*	WP6
Bridget Description	Provides access to bridget data structures and the associated metadata	*Fully integrated*	WP7

The AT uses an internal data model which is mapped to a distributed database. During the authoring process, data is extracted from the database and exported in a format interpreted by the BRIDGET Player. The data model used by the Player is based on MPEG MLAF [3]. The database behind the AT is structured at a conceptual level as depicted in Figure 42.

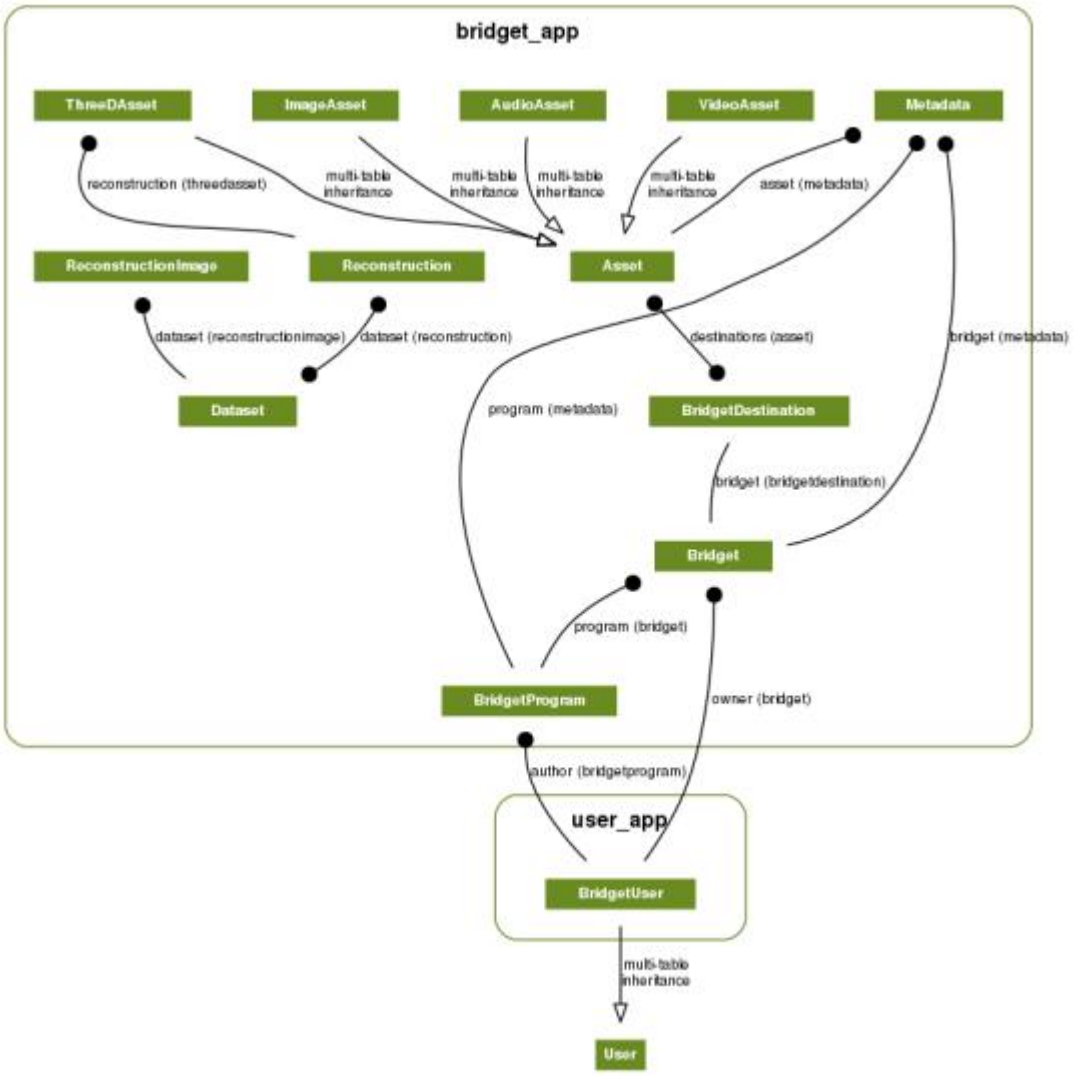


Figure 42: Authoring tools database structure

4 T7.3 - Immersive Media Rendering

This task was active in the reporting period between May 2015 and June 2016. In this period, research has been conducted towards two main directions:

1. Update of the 3D audio rendering module so that it can support microphone array recordings as an input to the audio processing unit (4.1);
2. Optimized rendering of synthetic 3D models (4.2).

4.1 3D audio module for "recorded" bridget

In order to allow broadcast producers to create interactive 3D audio bridget of a dynamic scene, a method for rendering the audio scene based on free field microphone array is developed. A pre-processing unit is added in the audio engine, (see **Error! Reference source not found.**) and its goal is to process microphone signals recorded by the microphone array so they can be used as input to the binaural synthesis. More technical details on the pre-processing module are given in D6.3.

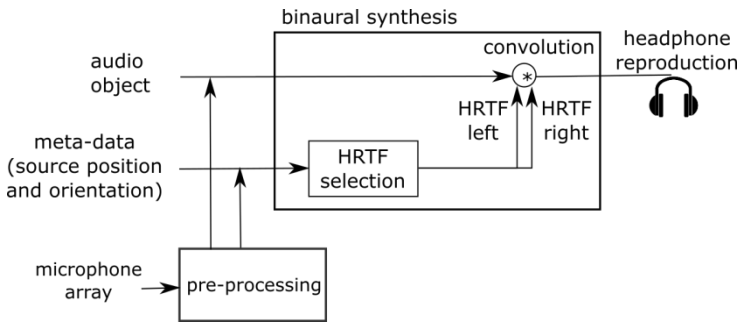


Figure 43: Block diagram of the audio rendering engine including pre-processing part for the microphone array recordings

4.1.1 Real-time binaural synthesis of a microphone array recordings

Microphone array recordings have been processed by using a beam-forming algorithm. Directional beams are applied to the multichannel recordings and the surrounded soundfield is sampled in discreet components each of which defines an audio object for a binaural synthesis. The optimization of a beam-forming process is done in the following terms:

- The experiments on the trade-off between single beam directivity on different frequencies (how narrow the beam is), a number of beams covering the full sphere and perceived spatial degradation are performed, and the optimal beam-forming solution is obtained. Figure 44 shows the obtained beam pattern with 18 beams.
- The design of the filters representing beams is optimized in order to increase the processing speed of a beam pattern creation.
- The length of the filters representing beams is optimized for a real-time binaural synthesis, allowing in that way a user to interact with a soundfield by rotating it, and thus changing the angle of listening.

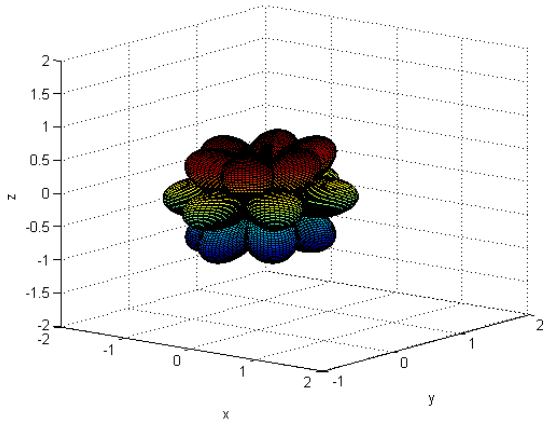


Figure 44: Beam pattern for a microphone array processing

4.1.2 Definition of 3D Audio Engine Interfaces

The inclusion of audio bridgets within the overall BRIDGET architecture entails the description of audio bridgets information, that needs to be packaged in a format understandable by any BRIDGET player. The interface defined, in order to support such scenario, is the following:

Input (bridget data package)

- Reference direction for matching 3D audio with video content (coordinate system of the scene);
- Beam pattern – layout of the beams arrangement for spatial segmentation of the audio scene (DSP filter that corresponds to a given direction);
- Spherical microphone array recordings – free field recordings of a bridget scene (32 channel 16-bit PCM audio file);

Output (computed in real-time at the player side)

- Binaural audio – mix of all input audio channels positioned in space around the listener (two channel audio file - 16-bit PCM with 48 kHz sampling frequency).

4.2 Optimized Rendering of Synthetic 3D Models

During the first half of the project's life, the techniques developed within T6.3 for rendering splat 3D models were ported into the mobile environment, namely the BRIDGET Player designed for embedded systems. For this purpose, a comprehensive analysis of the existing GPAC infrastructure (which is the basis for the BRIDGET Player) was performed. It focused on the data format used for the models, on model loading into a manageable memory structure, and on the efficient rendering process that enables the users not only to see and "understand in 3D" those models, but also to navigate around/inside them in an instinctive way.

The actions undertaken in the second half of the project aimed at improving this initial rendering approach. Textured-splat models can now be smoothly displayed in the app running on the mobile device thanks to the reduction of the general complexity of the rendering algorithms. More specifically, optimizations have been performed on the shaders which make the free-viewpoint rendering less computationally expensive and, therefore, more fluid. All these enhancements entail a raise in the users' satisfaction when interacting with the 3D content inside the BRIDGET Player, hence helping to achieve the global objective of the project.

The creation of the textured-splat models has been extensively described in deliverable D6.3, using the uniformly-coloured splat models and a set of images from the scene in order to obtain the texture atlas and apply it to every splat. This technique obtains a higher subjective quality for this kind of models, as colour variations inside the splats are achieved and the final models resemble more accurately the original natural scenes.

Figure 45: Point cloud (left), base splat model (centre) and textured-splat model (right) from Torino's Arco Valentino

Figure 45 shows the results of the different processing stages a 3D model undergoes, all of which can be rendered in the player: from the original point cloud, a base splat model is created whose appearance is then improved by obtaining a textured-splat model. Figure 46 shows a close-up of the latter to help

understand the benefits of the splat-based (modelling and) rendering approach vs. the point-cloud-based one.

Figure 46: Close-up of the textured-splat model shown in Figure 45

The main objective of the optimizations performed on the GLSL vertex and fragments shaders has been reducing the amount of data transferred between CPU and GPU, and between the shaders themselves, during the rendering process. This has been accomplished by keeping down to a minimum the number of variables needed to represent the ellipses (e.g., using the fact that their major and minor axes are perpendicular to each other), and compacting the internal variables that are computed in the vertex shader, and then transferred to the fragment shader so that the right conversion factor is applied to the texture coordinates. These operations yield significant savings not only in terms of memory requirements, but also in terms of rendering time, which impacts directly on the quality of experience perceived by the users of the BRIDGET Player.

Another feature that is still being ported to the Android ecosystem is the rendering of hybrid SPLASH models, whose definition is also covered thoroughly in deliverable D6.3. Conceptually, they merge meshes and splats to represent a single 3D object and take advantage of the benefits of both kinds of primitives. While the core of the model is a textured triangle mesh covering smooth areas, textured elliptical splats are used to model the more irregular and intricate details that are typically present in a real-world object. Due to the difficulties of implementing this novel concept in modern handheld devices, and, more importantly, within the well-established structure of the GPAC application, this task has demanded more time than initially expected.

Figure 47: SPLASH hybrid model obtained from Lisbon's Torre de Belem

Nevertheless, the results of these hybrid models may be observed in the powerful PC standalone application that has been developed for this purpose, as no out-of-the-box solutions are available to show these SPLASH models to the user. One of such results may be observed in Figure 47, namely, for the Torre de Belem model. The façades of the tower are mostly modelled with a triangle mesh, whereas irregular zones like the towers or the windowsills are represented through textured splats.

5 T7.4 - Multi-screen player

The main design principles of BRIDGET player were already reported in D7.1 and they are still applicable in the second part of the project. Still several changes (mainly incremental updates) were performed in the player during the second part of the project. In this section we mention these principles and introduce the new elements when needed.

5.1 Architecture Design and Implementation

This task addresses the architecture design of a BRIDGET Player prototype, as well as its implementation in the reference mobile terminals. The starting point was GPAC, a multimedia interactive framework, a property of IMT (but distributed as open source) [5] and the activity was focused on integrating MPEG ARAF functionalities in OSMO4, the GPAC multimedia player. OSMO4 is an open source MPEG-4 compliant multimedia player which is supported on multiple platforms: Windows, Windows mobile, Android, iOS, Linux and MacOSX. OSMO4 supports different multimedia formats as well as a JavaScript execution engine based on the Mozilla SpiderMonkey engine. The implementation supports most of the Script

features defined in the MPEG-4 standard and includes full support for interaction with any object in a scene. Additionally, Osmo4 supports PNG and JPEG for static images, MP3 and AAC for audio, H.264/AVC and HEVC for video. Files can be accessed from the local drive or through HTTP. OSMO4 uses OpenGL and OpenGL-ES for rendering mixed environments including 2D and 3D graphic objects. The player has also access to a variety of sensors such as motion sensors, location sensors, microphones, cameras and others.

Apart from implementing support of MLAF in OSMO, the main modification needed in this first period, in order to enable full support of BRIDGET functionalities in GPAC, was the introduction of a mechanism for support and management of audio fingerprint (AFP) within the player architecture.

The schema presented in Figure 48 illustrates the AFP functional architecture.

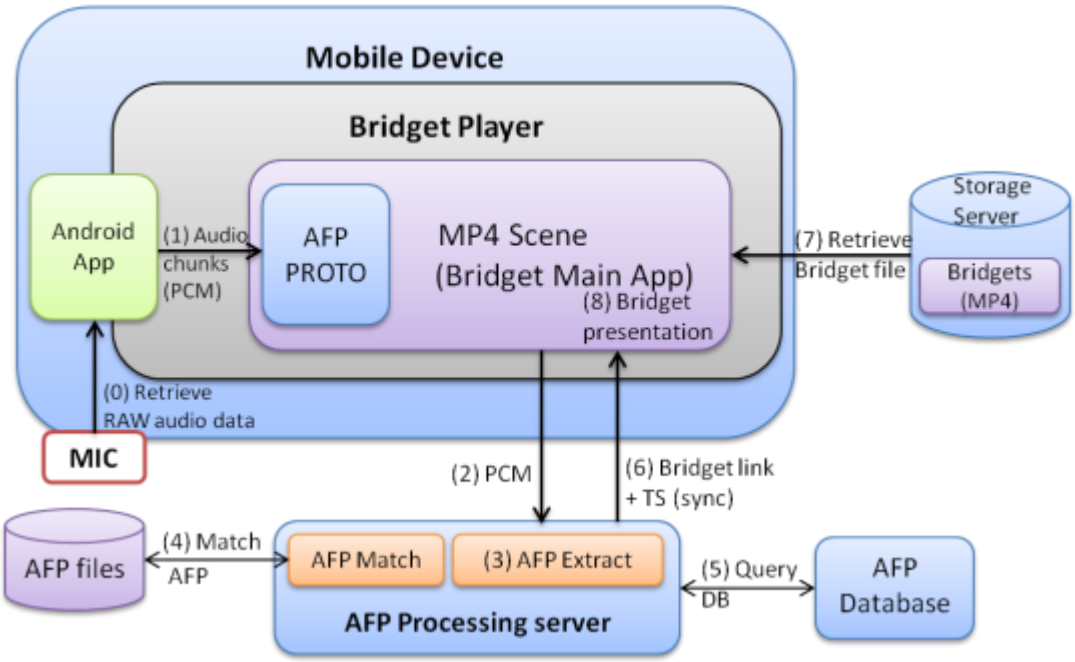


Figure 48: AFP Architecture

In order for the BRDIGET Player to retrieve bridgets and synchronize with the main screen, the following functionalities, described below, have been implemented:

- An AFP PROTO, which is a scene level feature implemented in the Bridget Player.
- A processing server where the audio fingerprint functionalities are running.

The AFP prototype has been designed to connect to the microphone of the device, through the Android application that acts as a wrapper over the native code of the BRIDGET Player. As long as the prototype is enabled, the raw audio data of the microphone is intercepted, chunked to a given size and sent to the BRIDGET scene (the main application).

A simplified schema of the AFP prototype is depicted in Figure 49.

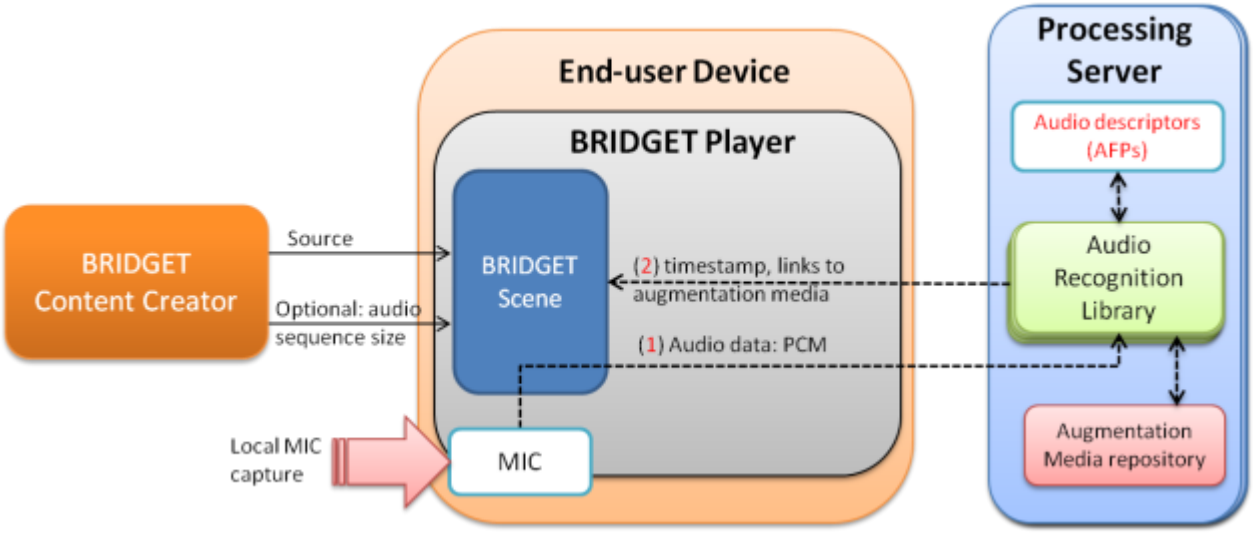


Figure 49: Bridget AFP PROTO

Further, the AFP PROTO interface is presented:

```
EXTERNPROTO AFP [
      exposedField MFString          source                  []
      exposedField SFInt32           buffer_mic              40000
      exposedField SFBool            isEnabled               FALSE
      eventOut     MFInt32           mic_data
] "urn:inet:gpac:builtin:AFP"
```

The **source** field of the prototype specifies the URL of the device's microphone. The application connects to the microphone and intercepts raw audio data (PCM) as long as the **isEnabled** field is TRUE. The PCM chunk size can be also controlled through the prototype instance by using the **buffer_mic** field. The recorder sample rate is hardcoded at 8000 samples per second, therefore a 5 second audio chunk corresponds to a buffer_mic value of 40.000. Whenever a new audio buffer is filled with data, the PCM array is sent to the Bridget Scene through the **mic_data** field of the prototype. A JavaScript XMLHttpRequest is used to send the audio PCM to the AFP Processing Server where the audio fingerprint services are running. The first operation performed by the processing server is to extract the audio fingerprint by using the Huawei AFP library (section **Error! Reference source not found.**) and then to match the result against all the extracted audio fingerprints that are already stored on the storage machine. The Huawei AFP matching library computes the number of common features between the reference fingerprint and the stored ones. If matchings with fingerprints from different programmes are found, a ranking algorithm is applied on the result in order to decide which of the programmes exposes higher audio similarity.

Following the same approach, if several timestamps are returned by the matching library, the algorithm filters the results and returns a single timestamp. Once the program name and the timestamp are identified, the processing server interrogates the database and gets the link pointing to the corresponding BRIDGET file (MP4). The link and the timestamp are sent back to the BRUDGET Scene as a result of the HTTP request previously initiated by the scene. Once the result is available, the BRIDGET Player downloads the BRIDGET file and starts presenting the content considering the timestamp received from the processing server. As long as the AFP proto is enabled, the same process runs continuously. Unless the server returns a link to a new BRIDGET file, the scene only considers the timestamp for synchronisation purposes. If a new link is returned (meaning that another program is detected), the same principle applies as described above.

The architecture and the prototype that have been designed to implement BRIDGET audio fingerprinting functionalities are compliant with one of the prototypes that have been introduced in the second edition of ARAF. The name of the PROTO is **RemAud** and it provides remote audio recognition support in an ARAF Player. The architecture of the RemAud prototype is presented in Figure 50.

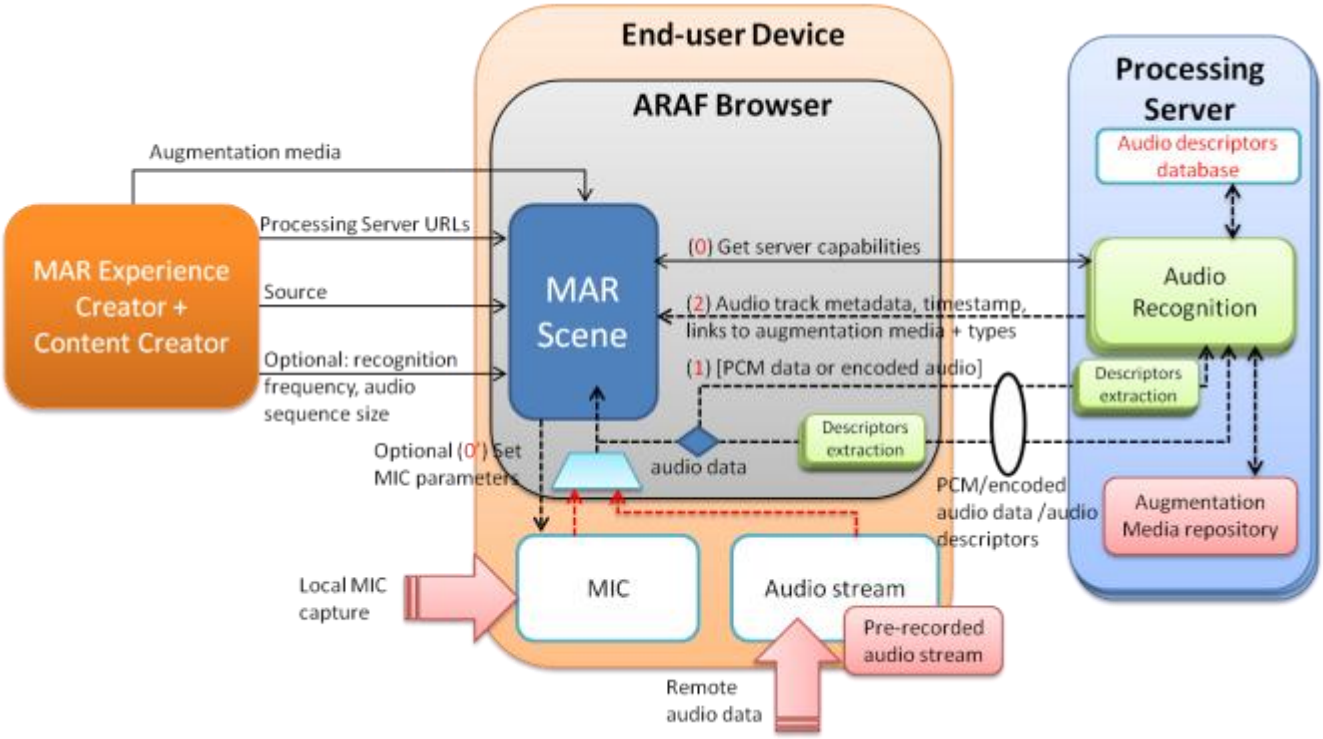


Figure 50: RemAud PROTO (ARAF)

The full description of the prototype can be found in [6], section 4.2.4.12.3.2.
The source, the audio sequence size, the way how the audio data is sent to the processing server and the server response are also defined here, and they have exactly the same meaning as described in the AFP prototype functionality.
RemAud provides more functionality than what is necessary in the BRIDGET scenario so far (AFP related); nevertheless RemAud can easily replace the simple prototype that has been specifically developed for the BRIDGET use cases.

6 T7.5 – Standardization

This task was active in this reporting period between November 2014 and July 2016. The effort has been mainly devoted to ensure compliancy of the AT and player formats to the new MLAF standard, and implement single components of player and ATs as MXM engines, as specified in WP3.

As a result of the activities carried out within WP7 (in some cases, jointly with WP6), several proposals co-authored by BRIDGET researchers were submitted to MPEG (the full list is available in WP9 report).

We also extended the ARAF standard to include a mechanism allowing audio fingerprinting, as described in section 5.1. At present, however, the format of audio fingerprint itself is not standardised.

7 Conclusions

This document summarises the research outcome of WP7 during the BRIDGET's life This work started already in the first phase of the project and a first version of the report was delivered as D7.1. In this version of the report we integrated the modifications and updates obtained in the second part of the project. WP7 progress is in line with project plans, and achieved expected outcomes, in particular a fully functional Authoring Tool and BRIDGET Player.

References

[1] http://www.mymultimediaworld.com/

[2] http://facebook.github.io/react/

[3] ISO/IEC SC29WG11 23000-13 (MPEG-A) N15293 "Study Text of ISO/IEC CD 23000-13, Augmented Reality Application Format", Geneva, February 2015.

[4] http://www.wimlabs.com/en/wimtv.html

[5] http://gpac.wp.mines-telecom.fr/

[6] A. Gabrielli, Y. Lehiani, T. Lavric, M. Preda (IMT), "ARAF: remote recognition – analysis and preliminary results", MPEG contrib. M34360, 109th MPEG mtg., Sapporo, JP, July 2014.