This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

D3.2.2 Sign language multilingual knowledge base and multilingual subtitles final version

EasyTV Project

H2020. ICT-19-2017 Media and content

convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 30 months

Document. ref.:

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	Short name	Country
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES

PROGRAMME NAME:	H2020. ICT-19-2017 Media and content convergence. convergence. – IA Innovation action
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	UPM
INVOLVED UNITS:	UPM, CERTH, CCMA
DOCUMENT NUMBER:	D3.2.2
DOCUMENT TITLE:	Sign language multilingual knowledge base and multilingual subtitles final version
WORK-PACKAGE:	3
DELIVERABLE TYPE:	R
CONTRACTUAL DATE OF DELIVERY:	31-12-2019
LAST UPDATE:	19-02-2020
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
v.0.1	15/10/2019	Draft	María Poveda-Villalón (UPM)	Table of Contents definition and document structure
v.0.2	1/11/2019	Draft	Pablo Calleja Ibañez (UPM)	easyTV annotator section included
v.0.3	20/11/2019	Draft	Pablo Calleja Ibañez (UPM)	Multilingual knowledge base for sign language videos included
v.0.4	12/12/2019	Draft	María Poveda-Villalón (UPM)	easyTV semantic model, conclusions, introduction and
v.0.5	26/12/2019	Draft	Thanassis Kalvourrtzis (CERTH), Kosmas Dimitropoulos (CERTH), Kiriakos Stefanidis (CERTH)	Video, text and sign language representation, conclusions
v0.6	28/01/2020	Draft	Jordi Fabregat, Jordi Arraez & Francesc Mas (CCMA)	Multi-language subtitles
V0.7	31/01/2020	First vesion	María Poveda-Villalón (UPM), Pablo Calleja Ibañez (UPM)	Compilation and review
V0.8	13/02/2020	First vesion review	Pilar Orero (UAB), Stavros Skourtis (ARX)	Internal quality check
V1.0	19/02/2020	Final version	María Poveda-Villalón (UPM), Pablo Calleja Ibañez (UPM), Jordi Fabregat, Jordi Arraez & Francesc Mas (CCMA), Thanassis Kalvourrtzis (CERTH), Kosmas Dimitropoulos (CERTH), Kiriakos Stefanidis (CERTH)	Updated content according to internal review

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
ACM	Accessibility Content Management
API	Application Programming Interface
APP	Usually used to reference to a software application
DTT, DVB-T	Digital Terrestrial Television
DVB	Digital Video Broadcasting
EBU-TT-D	Specifies an XML based format for the distribution of subtitles over IP-based networks.
FullHD	Full high definition. Video content in 1920x1080 high resolution.
HbbTV	Hybrid broadcast broadband TV standard
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
Linear TV	It is a real-time television service that broadcasts scheduled programs, conventionally over the air or through satellite/cable, not streamed to a specific user.
MAM	Media Asset Management
NLP	Natural Language Processing
OTT	Over-The-Top: used in streaming media servicea offered directly to viewers via the Internet
OWL	Web Ontology Language
PoS Tagger	Part of Speech Tagger
RDF	Resource Description Framework

RDF(S)	RDF Schema
REST	Representational State Transfer
RF	Radio Frequency
RGB	Red, Green, Blue
SBT	Subtitle
SD	Standard Definition. Video content in 720x576 or similar standard resolution.
SPARQL	SPARQL Protocol and RDF Query Language
SPM	Subtitle Production Module
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniforme Resource Locator
VOD / VoD	Video on Demand
W3C	World Wide Web Consortium
WCM	Web Content Management

Table of Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

The present document is the deliverable “D3.2.2 Sign language multilingual knowledge base and multilingual subtitles final version” of the EasyTV project,[1] funded as an Innovation Action by the European Commission Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and Innovation Programme (H2020).

This deliverable documents and summarizes the results of transforming into Linked Data sign language videos using the EasyTV ontology. A second pillar of this deliverable describes the design and development of Multilanguage subtitle production.

The documentation covers, among others, the following topics:

- The final version of the **EasyTV ontology** is presented in this deliverable providing an overview of the evolution of the model along the project.
- The description of the process to transform sing language videos as Linked Data and publishing them on the web of data.
- **The Relationship and integration** with other **modules** of the project in which the multilingual ontology is used, more precisely within the crowdsourcing platform. The connection between the EasyTV annotation module with other modules as the Sign Language capturing system is also explained.
- A detailed description of the multilanguage subtitles production module is provided, giving an end-to-end vision from content production to final user consumption, and explaining the connection of EasyTV multilanguage module with other existing EasyTV components such as the crowdsourcing platform.

[1] http://easytvproject.eu/

1. INTRODUCTION

The EasyTV crowdsourcing platform relies on the EasyTV ontology in order to ease the translation between videos in different Sign Languages. In computer science, ontologies are defined as "formal, explicit specifications of a shared conceptualization" [8]. The EasyTV ontology is formal in the sense of following Description Logics and being implemented in the W3C Web Ontology Language standard OWL.[2] The EasyTV ontology is being used within the crowdsourcing platform as reference model to annotate Sign Language and link them through the meaning of the concepts encoded in the videos. Such links are exploited to facilitate the matching between Sign Language videos previously stored and annotated in the crowdsourcing platform. This document will describe in detail the final version of the ontology and the related services allowing its exploitation. This document provides an update on the work described in the EasyTV deliverable "D3.2.1. Enriched multilingual ontology with signs in different languages preliminary version" in which more information about the process followed and background knowledge on semantic models can be found. In addition, this document discusses the integration developments and status in the context of the Crowdsourcing platform describing the communication requirements and the interaction of the annotator service's API with the platform. Finally, the multilingual subtitles service integration with the crowdsourcing platform is detailed in this deliverable.

The remainder of this deliverable is structured as follows:

- **Section 2** described the final version of the EasyTV ontology.
- **Section 3** focuses on the video exchange format and the interaction of annotator with the capturing and crowdsourcing technology.
- **Section 4** details the features of the semantic annotator services by exploiting the EasyTV ontology integrated within the crowdsourcing platform.
- **Section 5** describes the implementation of the EasyTV multi-language module integrated within the crowdsourcing platform.
- **Section 6** presents a set of conclusions from the work carried out.

[2] https://www.w3.org/TR/owl-ref/

2. EASYTV SEMANTIC MODEL FINAL VERSION

This section describes the final EasyTV ontology including an example of use for data annotation. The ontology is publicly available under its URI https://w3id.org/def/easytv and is published following the best practices for publishing vocabularies defined by the web consortium W3C.[3]

During the development of the EasyTV ontology a number of existing models have been reused in order to maximize interoperability with existing linguistic resources and minimize development efforts avoiding the re-implementation of existing models. More precisely, the ontologies shown in the "Referenced Ontologies" box in Figure 1, also listed in Table 1, have been reused. Table 1 shows the relation of ontologies reused URI and the prefixes used to refer to them along the document.

Table 1. Reused ontology prefixes and namespaces

Prefix	Name	Namespace
bln	BabelNet model	http://babelnet.org/model/babelnet#
lemon	Lemon	http://lemon-model.net/lemon#
lexinfo	Lex Info	http://www.lexinfo.net/ontology/2.0/lexinfo#
owl	OWL	http://www.w3.org/2002/07/owl#
rdfs	RDF(S)	http://www.w3.org/2000/01/rdf-schema#
skos	SKOS	http://www.w3.org/2004/02/skos/core#
vartrans	Variation and translation module of OntoLex	http://www.w3.org/ns/lemon/vartrans

The final conceptualization implemented in the EasyTV ontology is depicted in Figure 1. In such figure the following graphical conventions are used:

Coloured rectangles are used for classes created in the EasyTV ontology while white rectangles for reused classes. For all the items (classes, object properties and datatype properties), it is indicated in which ontology they are defined by the prefix included before their identifier.

Arrows are used to represent properties between classes and to represent some rdf, rdfs and owl constructs, more precisely:

- Plain arrows with white triangles represent the `rdfs:subClassOf` relation between two classes. The origin of the arrow is the class to be declared as subclass of the class at the destination of the arrow.
- Plain arrows between two classes indicate that the object property has declared as domain the class in the origin and as range the class in the destination of the arrow. The identifier of the object property is indicated within the arrow.
- Dashed labelled arrows between two classes indicate that the object property can be instantiated between the classes in the origin and the destination of the arrow. The identifier

[3] https://www.w3.org/TR/swbp-vocab-pub/

of the object property is indicated within the arrow.

- Dashed arrows with identifiers between stereotype signs (i.e., "<< >>") refer to OWL constructs that are applied to some ontology elements, that is, they can be applied to classes or properties depending on the OWL construct being used.
- Dashed arrows with no identifier are used to represent the `rdf:type` relation, indicating that the element in the origin of the arrow is an instance of the class in the destination of the arrow.

Datatype properties are denoted by rectangles attached to the classes, in an UML-oriented way. Dashed boxes represent datatype properties that can be applied to the attached classes while plain boxes represent that the domain of the datatype property is declared to be the class attached.

Individuals are denoted by rectangles where the identifier is underlined.

Literals are denoted by rectangles where the value is included between quotation marks.

The representation of additional property axioms (functional, inverse functional, transitive, and symmetric) that are being used in the diagram are shown in the overview ontology legends.

In addition, each ontology overview picture includes a legend to remind the reader the graphics meaning.

Figure 1 shows the EasyTV ontology which focuses on the description of Sign Language videos and their annotation with linguistic information for which several linguistic models have been reused. This model is the final version of the one previously presented in D3.2.1, for this reason part of the concepts and properties described in this document have already been presented in the first version. However, the model is detailed in this document to provide a complete view for the reader.

The main concept defined in the EasyTV ontology is `etv:Video` which represents the videos uploaded to the crowdsourcing platform. For each instance of this concept, the language in which the video is recorded and at least 1 URL indicating the location of the video should be provided by means of the attributes `lemon:language` and `etv:url` respectively. If a video is composed by frames, each frame should be represented as an instance of the concept `etv:VideoFrame`. Each video frame should be linked to the instance representing the whole video by means of the property `etv:isComponentOf`. This model has been extended in this latest version to include a mechanism to identify the first frame (`etv:firstFrame`) and to point from each frame to the next one (`etv:nextFrame`). It is important to note that this model replaces the first version of the video composition model based on lemon (`lemon:ComponentList`). The new model is more accurate and less verbose than the previous version.

The order of the terms in written language sentences might not match the order of the corresponding Sign Language sentence, indeed, Sign Language allows representing terms or phrases information in a non-sequential way. For this reason, two concepts are define to transcript the order of the terms appearing in a sentence in Sign Language (`etv:SingedLinguisticExpression`) and in written language (`etv:WrittenLinguisticExpression`). These two concepts represent specializations of the `etv:LinguisticExpression` class.

Lexical entries can be represented by more than one grammatical realisation, that is, an entry can have more than one "form" (`lemon:Form`). According to the lemon model, individuals of `lemon:LexicalEntry` are linked to their forms by means of the property `lemon:lexicalForm` or its specialiations: `lemon:canonicalForm`, `lemon:abstractForm` and `lemon:otherForm`.

The EasyTV ontology extends the concept `lemon:Form` by adding as subclass the concept `etv:SignedForm`. In this case, lexical entries are linked to `etv:SignedForm` by means of the property `etv:signedForm` which is a sub-property of the `lemon:lexicalForm`. The signs are realised by videos or video frames and it is indicated by means of the property `etv:signedRep`.

As the model is intended to be used to annotate data that will be linked to BabelNet, some parts of the BabelNet model are reused in order to maximize interoperability. As BabelNet highly relies in the lemon model, such model is reused in the EasyTV ontology. For this reasons, the lexical entries are linked to their lexical sense (`lemon:LexicalSense`) by means of the relation `lemon:sense`. The lexical sense can be understood as a bridge term that unambiguously relates, by means of the property `lemon:reference`, the different meanings of a lexical entry with each ontological concept (`skos:Concept`).

In the previous version of the model two mechanisms to indicate the translation in a lexical level were taken into account, more precisely: a) the relation `lexinfo:translation` established between lexical senses, as used in BabelNet and b) the relation `vartrans:translatableAs` established between lexical entries. In this final version of the EasyTV ontology a new concept for representing translations has been included. In this case to represent the translation between videos. For this reason a new class `etv:Translation` have been created. Each instance of such a class would be linked to the two `etv:LinguisticExpression` instances involved in the translation. This concept also includes information about the use of such translation within the EasyTV crowdsourcing platform like the confidence of the translation (`etv:confidence`) which represents a percentage and whether the translation is suggested by a crowdsourcing user, therefore a signer, or it is inferred by the system (etv:`expertSuggestion`).

The model also allows the annotation of morphosyntactic information (`lexinfo:MorphosyntacticProperty`, `lexinfo:Gender`, `lexinfo:Negative`, `lexinfo:Number`) considering future use of the ontology in more detailed scenarios.

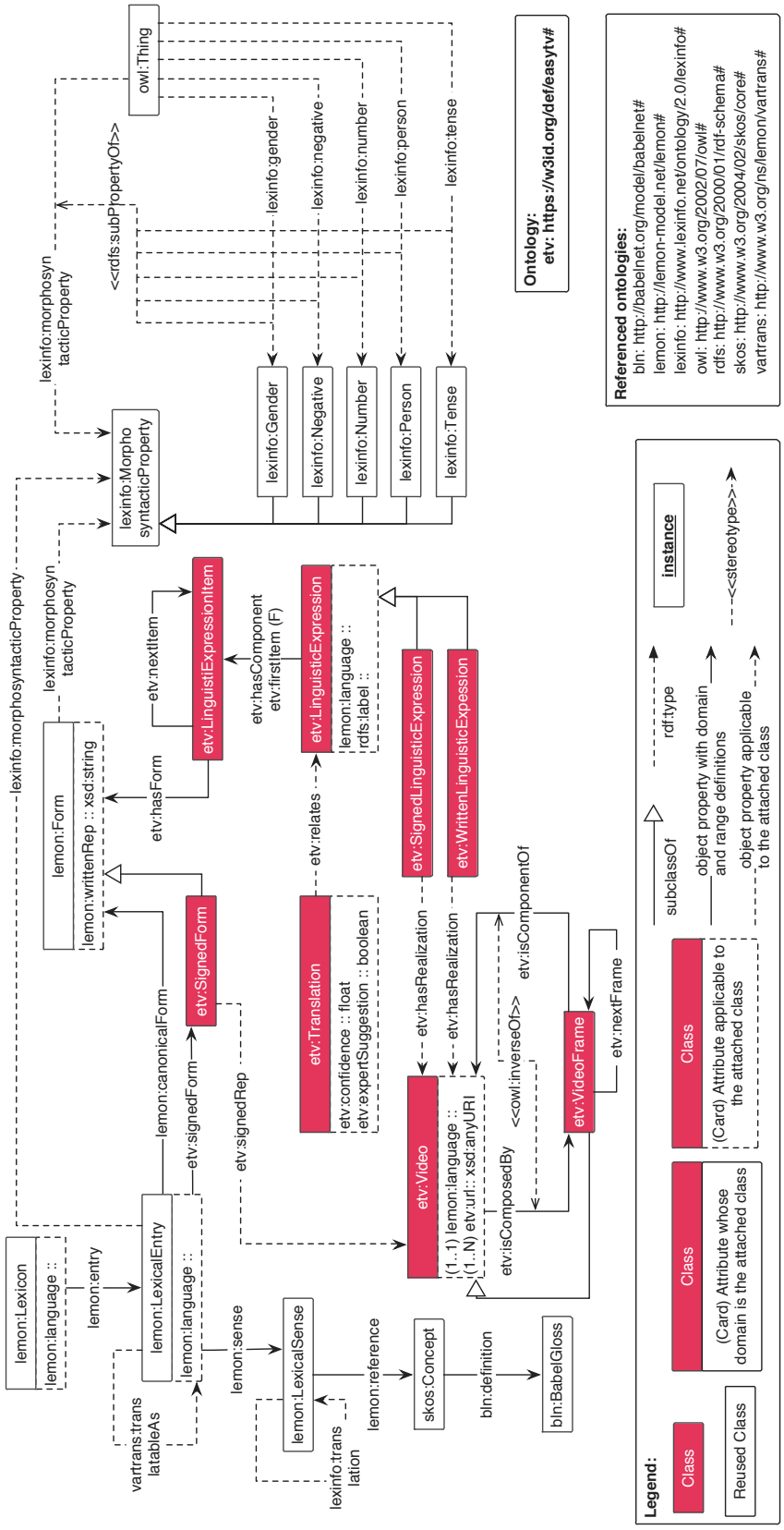

Figure 1. Conceptual model of the EasyTV ontology

Figure 2 shows an example of how the ontology is used to semantically annotate Sign Language videos provided by the crowdsourcing platform. Such figure shows an excerpt of an RDF graph in a graphical way. The example provided represents the RDF annotations that have to be generated to annotate a given video representing the clause "*Televisión en color*" which is composed of the sequence of sings of "*televisión*" and "*color*" in that order. It is worth noting that the sign representing "*televisión*" is linked to the lexical entry "television" in English provided by BabelNet as it can be observed by the prefix "bn" in the instance `bn:television_n_EN`. Such translation link is made by means of the property `vartrans:translatableAs`.

For readability concerns, only the complete RDF is shown for the term "*televisión*" being equivalent for the term "*color*".

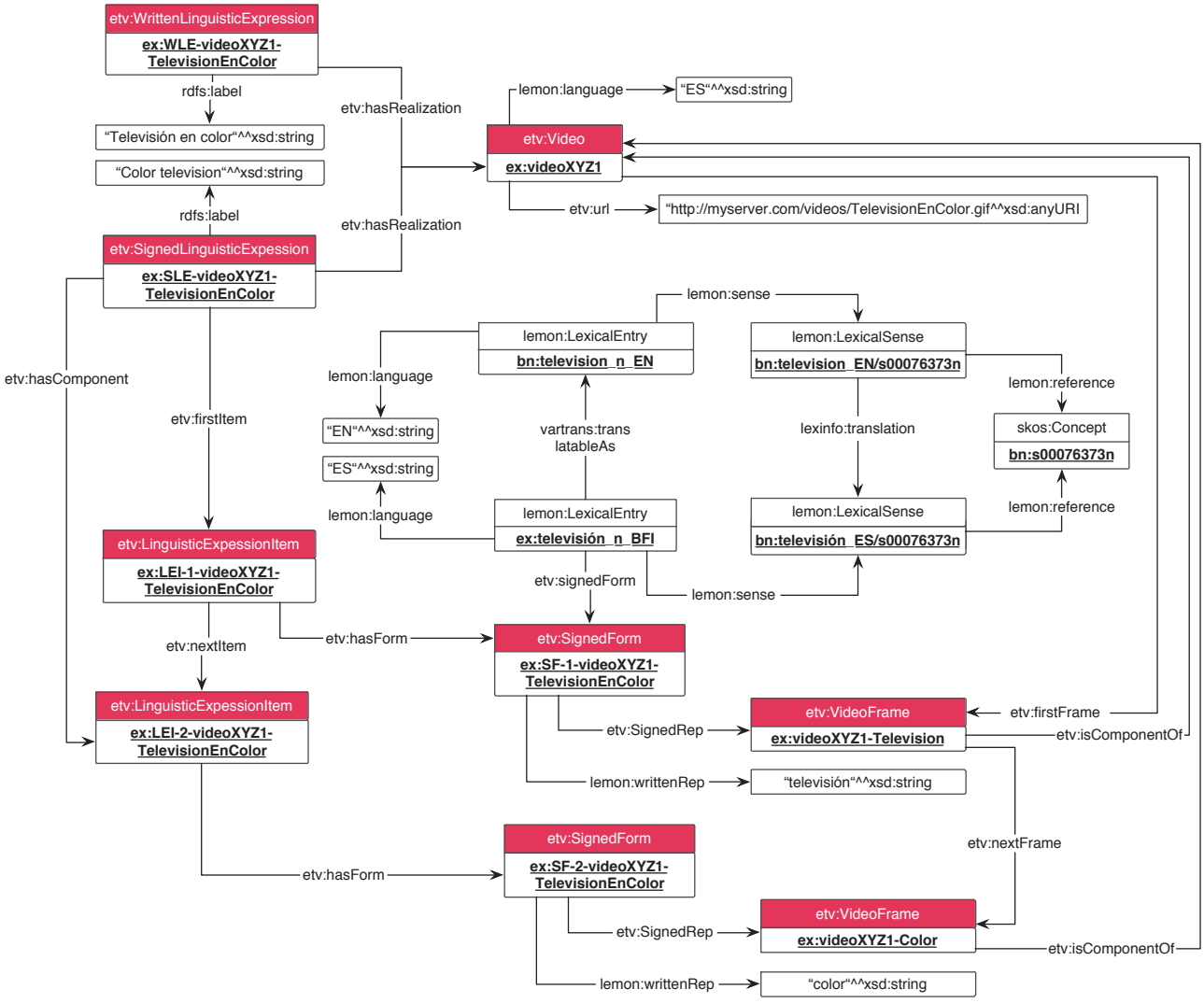

Figure 2. Graphical example of RDF triples

Finally, Figure 3 shows the RDF that should be generated when a Sign Language expert introduces a translation through the crowdsourcing platform. In this case, the expert links with a confidence of 90% the videos for “Televisión en color” and “Colour TV set”. As it can be observed, an instance of the class `etv:Translation` is created to link the signed languages expressions for “Televisión en color” and “Colour TV set” indicating the 90% of confidence and that that information has been suggested by an expert (`etv:expertSuggestion` value set to “True”)

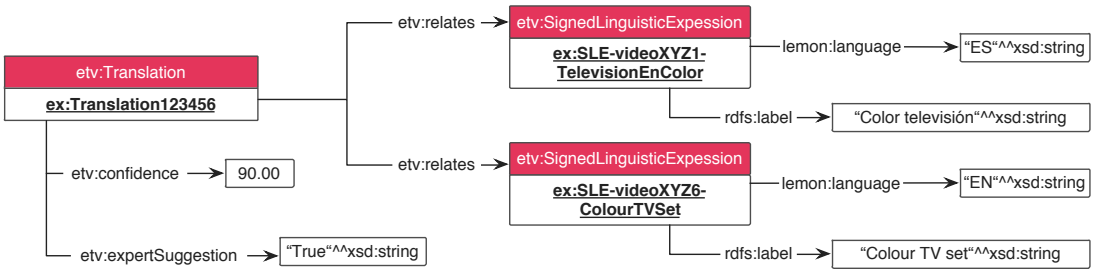

Figure 3. Graphical example of translation annotation from the crowdsourcing platform in RDF

3. VIDEO, TEXT AND SIGN LANGUAGE REPRESENTATION

3.1. Sing Language Content Generation with Capturing Module

For the capturing of Sign Language, we use the EasyTV capturing module, which has been developed in Task 3.1 and its final version is described in D3.7 [2]. The module is responsible for acquiring, i.e., capturing and storing, video frame sequences from a depth sensor and extracting the information needed for the construction of motion data, which is used for the realistic avatar playback (in Task 2.4). Usually, the capturing process takes place in a room with good lighting conditions and involves a signer speaking Sign Language in front of the depth sensor. Moreover, an administrator user that controls the capturing software gives instructions to the signer for a proper location, and marks the start and the end of the capturing process while checking the validity of the recordings. An example of the capturing procedure and the involved equipment is show in Figure 4.

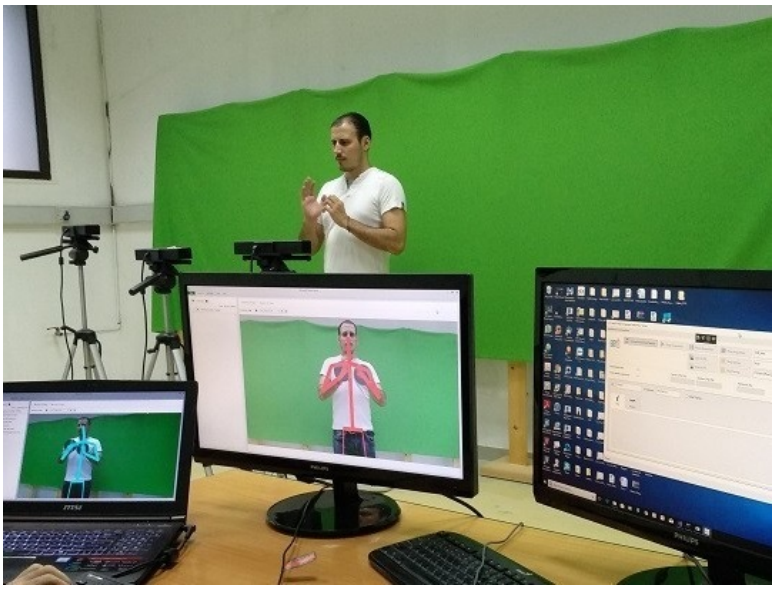

Figure 4. Testing a multi-view setup for sign language capturing

In order to produce accurate and noiseless motion data, a series of post-processing steps need to be applied after the image acquisition phase. More specifically, in the final version of the EasyTV capturing module uses Intel RealSense D435 sensor to capture high quality RGB and depth sequences. Both types of imaging data are important, with the former (i.e., RGB) being used by the keypoint detection algorithms while the latter (i.e., depth) being necessary for the generation of 3D data. The sequences of the recorded frames are merged into a single video in order to be in a compact format for further use.

The recorded video frames are fed as input into the motion analysis algorithms for the extraction of skeletal data. Skeletal data consist of 3D joints, i.e., keypoints, corresponding to the key visual content of each frame. Due to the nature of Sign Language, the capturing module detects 2D keypoints on the hands, face, and body of the signer. Since the detectors currently used in the EasyTV capturing technology are deep neural networks trained on extracting 2D-keypoints from RGB images, it is also necessary to infer the 3rd dimension for the motion data, that is, depth. An additional processing phase is required towards the generation of 3D keypoints. This process involves exploiting the depth images captured by the sensor.

The files generated by the capturing pipeline are exported in a compatible format that takes into

account the requirements of EasyTV services and modules, such as the multilingual ontology and the realistic 3D avatar which interact with the crowdsourcing platform.

3.2. Crowdsourcing platform and the ontology-based Annotator

The EasyTV Crowdsourcing Platform, whose current setup is described in deliverable D5.6 [3], is a web application developed to provide infrastructure for the creation, distribution and assessment of crowdsourcing tasks. More specifically the platform provides a number of functionalities targeting the creation of an online Sign Language knowledge repository with the collaboration of crowd workers. For this purpose the platform provides user interfaces that allow the creation and management of user profiles and web-based annotation tools for their participation in the Sign Language Production procedures through their browser. The full realisation of the pipeline is based on the integration of various EasyTV components which take part in the Sign Language production platform interacting as presented in the diagram Figure 5. The outcome of the crowdsourcing generated content is intended to be made available back to the users in two main ways, by the multilingual Sign Languages translations and by the retrievable 3D motion-data playable by animation engines.

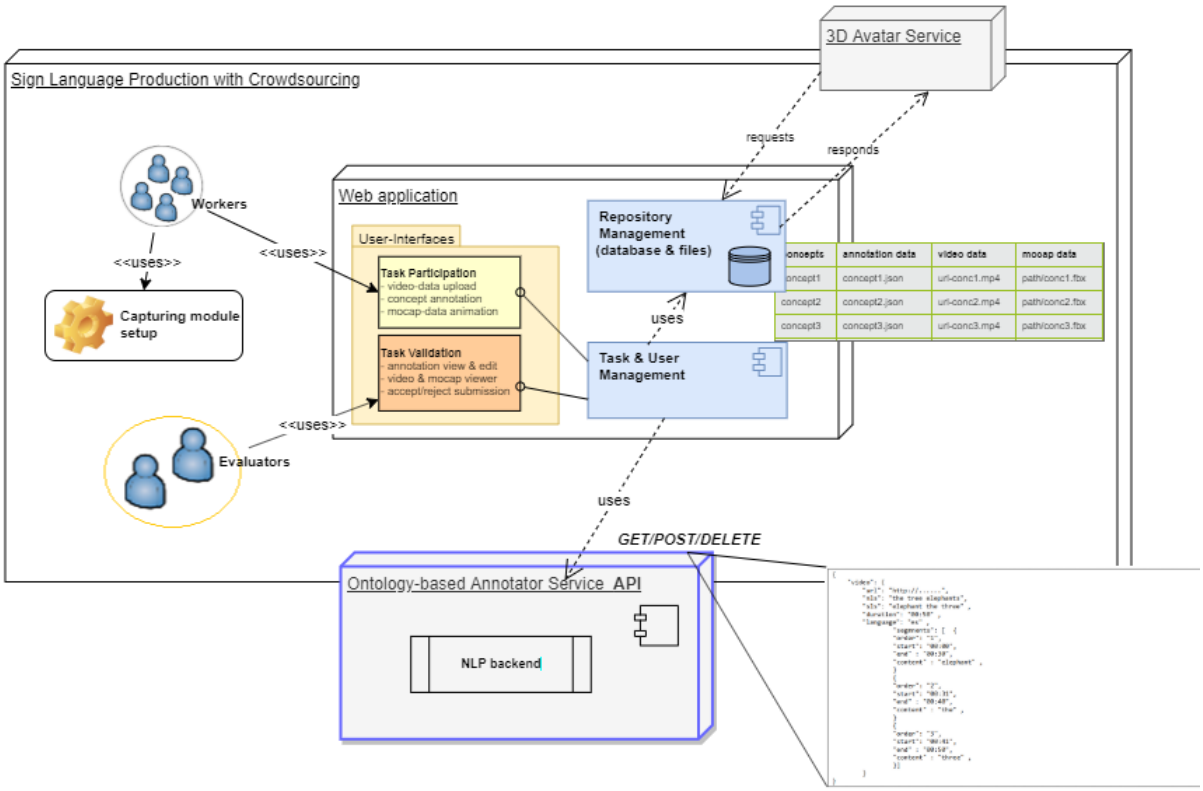

Figure 5. Overview of the integration with Crowdsourcing Platform and the Annotator Service API

In the context of a Sign Language task, a signer crowd-worker, after the end of a capturing session, obtains the output of the EasyTV capturing module which consists of a set of files containing both RGB videos and 3D motion for the recorded signs. These files are subsequently uploaded to the crowdsourcing platform as a response to the task that is being completed. The platform prompts the users to take a series of steps for their submission to be considered valid. For this purpose the platform presents the submitted content through a browser-based video-annotation tool and prompts

her/him to annotate and time-segment in two different sentence orders: one corresponding to Sign-Language Order and the second to Natural-Language Order. These annotations along with the recorded video are the input towards the Sign Language enrichment of the EasyTV Multilingual Ontology. The platform collects the users input through the video annotation web-application demonstrated in Figure 6. This is a browser tool that enables the crowdsourcing users to time-segment their video and to assign the appropriate subtitle annotations to it. After the user submission, the platform post-processes and formulates the inputs according to the specifications of the REST API developed for the EasyTV Ontology-based Annotator Service[4]. The platform interaction with the annotato API is implied in Figure 5, where the task-management component of crowdsourcing is indicated to execute HTTP requests to the Annotator Service API using appropriately JSON-formatted data.

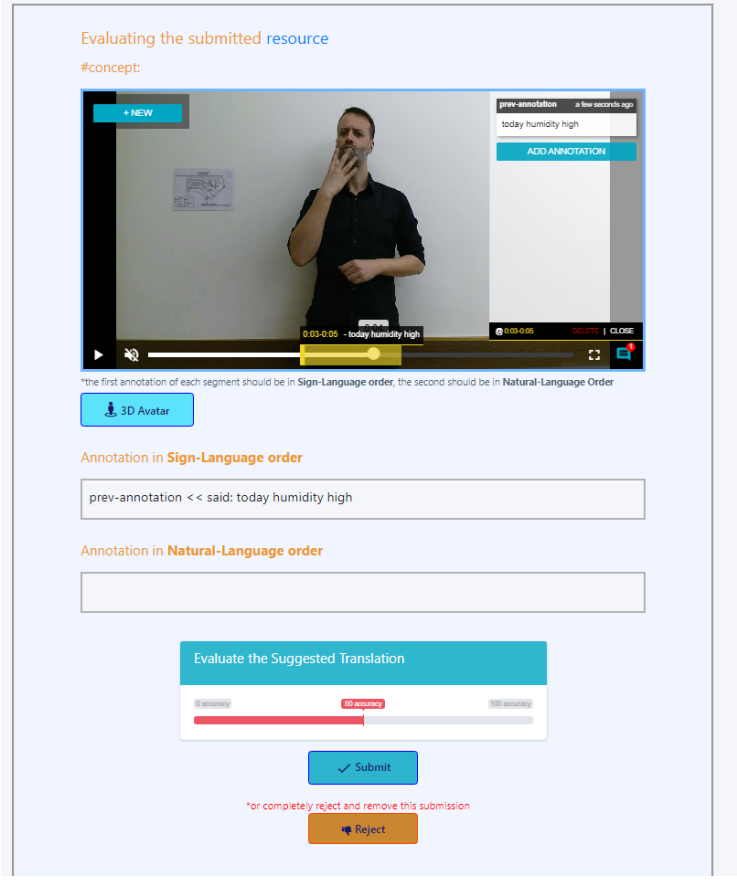

Figure 6. Web-based video annotation tool used in crowdsourcing tasks

It should be noted that the video-annotation tool presented Figure 6 is also being used during the validation step of the crowdsourcing process as a playback viewer adapted to support the evaluation process. The evaluator users of the platform are responsible for the final decision that instructs the platform to reject or accept and submit the content to the ontology. Finally the API of the Annotator Service has also been designed to enable the crowdsourcing platform to execute search queries based on concepts in specific languages. This allows the crowd-workers to retrieve annotated entries of the ontology and access their content (video and text) through the platform interface.

[4] http://api.easytv.linkeddata.es/easytv-annotator/swagger-ui.html

4. EASYTV ONTOLOGY-BASED ANNOTATOR SERVICE

The ontology is integrated in the EasyTV architecture through a web service that receives the HTTP requests and manages the information that is published or queried to the ontology. The service works under REST petitions and allows the communication with the needed functionalities of the crowdsourcing platform. This section presents the web service and the different modules that it needs as well as the project repositories.

4.1. Introduction

The information managed in the EasyTV project about the captured Sign Language videos has to be treated and processed before working with the Sign Language ontology. The captured information lacks of the required semantics to create an instance of the video in the Sign Language ontology. Moreover, an easy access to retrieve, manage or update videos in the ontology is needed. Thus, a web service has been created in order to provide functionalities to work with the ontology with a simple REST API. The web service is composed of different modules that provides a particular functionality. The developing language has been JAVA and each module belongs to a different repository.

4.2. Architecture

Figure 7 shows the architecture of the full **EasyTV Ontology-based Annotator Web Service.** The architecture is divided in 6 modules explained in this section.

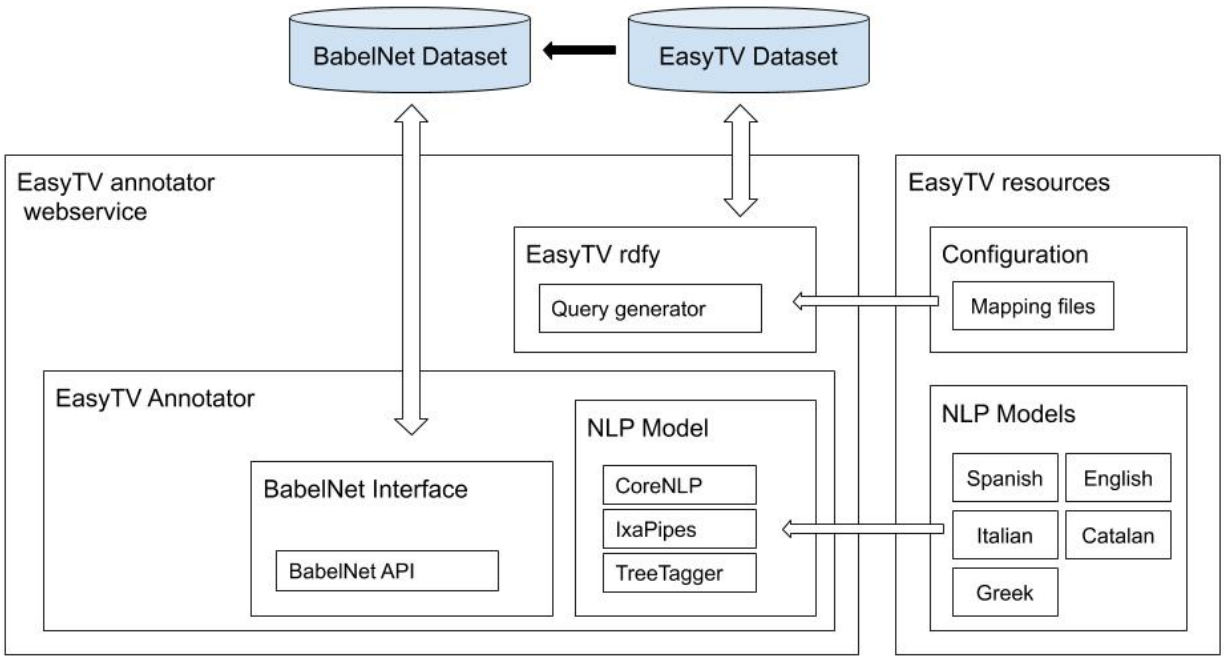

Figure 7. EasyTV Annotator architecture

EasyTV Annotator Web Service: This is the main module that manages the REST requests. The web service is constructed under the JAVA Framework Spring[5].

The web service receives information through parameters in GET in petitions or by already predefined JSON files, which corresponds with information provided by the crowdsourcing platform. The web service sends this information to the EasyTV Annotator module, to enrich the data with semantic information, and finally sends it to the EasyTV RDFY module to transform the data to RDF for the ontology.

The web service contains the necessary services to upload a Sign Language video, upload two Sign Language videos that are translations between them, retrieve translations from a particular Sign Language video and update the value of the accuracy of the translation between two Sign Language videos, among others. The complete list of services are detailed are presented in Section 4.4.

The developed repository is available in the following link: https://github.com/oeg-upm/easytv-annotator-webservice

EasyTV Annotator: The EasyTV Annotator is the module developed to work with the received Sign Language videos by annotating them with semantic information and by extracting suitable representations of the concepts of the video in the BabelNet ontology. The module uses the transcription of the video in a natural language sentence that is included in the Sign Language information for the two tasks. The module is divided into two submodules which are in charge of two different tasks: the Natural Language Processing annotator and the BabelNet Interface.

- **NLP Annotator**. This module processes the natural language sentence identifying its terms and words. It groups all the libraries used in the project for NLP tasks for the different languages with a common interface. These included libraries are coreNLP [6], Ixapipes [1] and TreeTagger [7] which are used for the languages of the project. CoreNLP is used for English, IxaPipes is used for Spanish and Italian and TreeTagger for Greek and Catalan. All these libraries perform the same tasks: they receive a natural language sentence and they execute the tasks of tokenization, part of speech tagging and lemmatization over it. The information obtained in these tasks is needed to find a suitable representation of the terms of the video in the BabelNet ontology.

- **BabelNet Interface**. This module contains the methods of the JAVA API to access BabelNet Dataset to find a suitable representation of the terms or words in the natural language sentence and to associate the video frames to such representations. BabelNet API methods can retrieve synsets of the ontology specifying language, part of speech of the word and the source of the sysnset. Moreover, the method can retrieve the most relevant sense of a synset for a specific language.

The JAVA API library and its specifications are presented at: https://babelnet.org/guide

The developed repository is available in the following link: https://github.com/oeg-upm/easytv-annotator

EasyTV RDFY:

The purpose of the module is the creation of the RDF triples necessary to upload or modify

[5] https://spring.io/

information in the Sign Language ontology with the information provided in the web service and transformed by the EasyTV Annotator. Moreover, the module contains the necessary queries to interact with the ontology. The module is based on the library RDFY and adapted to the project use case.

The developed repository is available in the following link: https://github.com/oeg-upm/easytv-rdfy

EasyTV Resources:

Most of the resources used in the EasyTV Annotator are static models for each specific language and each specific task for the different integrated libraries. Moreover, the EasyTV RDFY module needs mapping files that contains the configuration parameters to transform the annotated data to RDF following the specifications of the ontology. All of these models and configuration files are static and big in terms of memory. Thus, these resources should not be deployed inside the web service.

The module of resources is deployed outside the web service architecture and referenced through a configuration file. This solution allows to easily compile and deploy new versions of the web service without modifying the static content.

The developed repository is available in the following link: https://github.com/oeg-upm/easytv-resources

BabelNet Dataset. The BabelNet Dataset is a multilingual encyclopedic dictionary and a semantic network that contains more than 16 million of multilingual entries stored in RDF triples. BabelNet covers more than 284 languages and integrates information from different sources such as WordNet, Wikipedia or Wikidata.

This module does not belong to the EasyTV-annotator library. However, it is present in the architecture because it is used through the communication service of the BabelNet API in the EasyTV Annotator.

The public SPARQL Endpoint is available at: https://babelnet.org/sparql/

EasyTV Dataset. This module is the SPARQL endpoint where the results of the EasyTV RDFY module are stored and consulted. Each dataset entry contains the Sign Language information that will enrich the multilingual ontology BabelNet, storing Babelnet synset identified for each word, language, part of speech and the video frame that involves its representation.

http://easytv.linkeddata.es/sparql

4.3. Sign Language Videos and their population into the ontology

The main purpose of the web annotator service is to work with the Sign Language videos captured, annotated and sent from the crowdsourcing platform. The communication of these videos from the crowdsourcing platform to the web service is given by JSON. To upload a Sign Language video into the ontology the web service provides the service *annotateVideo*.

The Sign Language videos are annotated in the following format:


```
{
   "video": {
      "url": "http://nlp-easytv.oeg-upm.net/video/en/2.mp4",
      "nls": "air",
      "sls": "air" ,
      "duration": "00:03" ,
      "language": "en" ,
                  "segments": [{
                  "order": "1",
                  "start": "00:00",
                  "end" : "00:03",
                  "content" : "air" }]
    }
}
```


The fields of the JSON are:

- **url**: the URL of the video where it is saved.
- **nls**: the concept or sentence that is represented in the video written in natural language.
- **sls**: the concept or sentence that is represented in the video written in the order in which the signs appear in the video.
- **duration**: duration of the video.
- **language**: Language of the video.
- **segments**: List of segments. Each sign represented inside the Sign Language video is annotated as a segment, which includes:
    - **order**: position of the segment inside the video.
    - **start:** time when the segment starts.
    - **end:** time when the segment ends.
    - **content:** the word or words of the natural language sentence (nls) represented in the segment.

The need to include the field of the natural language sentence is because the video segments of the Sign Language videos may not correspond to the literal representation in the same order, as Figure 8 shows.

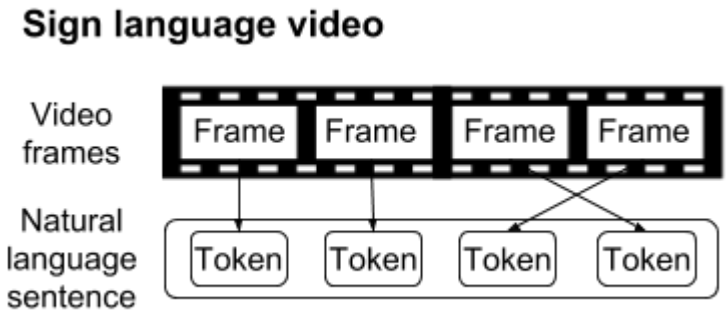

Figure 8. Sign Language video

The web service receives the Sign Language Video represented in the JSON format and starts the annotation process. Firstly, the video is sent to the annotator module (**easytv-annotator**) to extract the semantic information necessary to load the video into the Sign Language ontology.

The EasyTV annotator sends the natural language sentence to the NLP annotator module to execute the corresponding NLP library for the specific language. The libraries included are Stanford CoreNLP for English, IxaPipes for Spanish and Italian and Treetagger for Greek and Catalan. Each library is prepared with the tasks of tokenization, part of speech tagging and lemmatization. When the libraries are executed, they read and process the natural language sentence (nls). The results are tokens with the information retrieved in each task. As the libraries work differently, a common objects have been created for the representation of the sentence and its tokens in a uniform way. This object is ESentence and it is comprised of ETokens.

Figure 9 shows the structure of the ETokens in the ESentence and the information retrieved by the libraries at the end of the three tasks: the information about the word, the lemma and the part of speech.

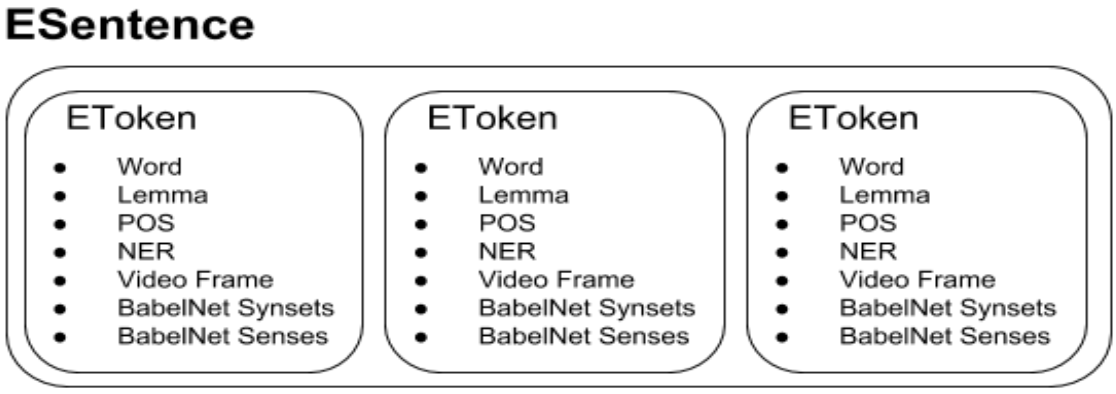

Figure 9. Structure of the ESentence and ETokens

Then, the ESentence is sent to the BabelNet Interface module. The purpose of the BabelNet interface is to retrieve the best candidate class in the ontology for the words (tokens) of the natural language sentence. Using the information retrieved by the NLP libraries for each token (the lemma of the words, the part of speech and the language), the BabelNet interface retrieves the most accurate class (synset) in the BabelNet Dataset that represents the token (e.g., for the word “house” in English, the best representation in the BabelNet Dataset is https://babelnet.org/synset?word=bn:00044994n).

Usually, the BabelNet Interface retrieves more than one synsets in this process due to the ambiguity of certain words and the different lexicons used in BabelNet. For the disambiguation of the synsets, the module orders them by its relevance and stores up to three synsets. For each synset, the best sense for the language is captured. Finally, all the retrieved information is stored in the ETokens.

Once the semantic information has been captured for the Sign Language video, it is sent to the EasyTV RDFY module to create the RDF triples to upload the Sign Language video to the Sign Language ontology with all the information retrieved in the previous processes.

Moreover, it is possible to receive two videos at the same time from the crowdsourcing platform to be annotated as mentioned before, and stored as direct translations. The service is *annotateTranslatedVideos* and it covers the use case in which a user in the crowdsourcing platform knows a concept in two different Sign Languages and wants to upload both videos. The format of the JSON file is similar to the one of the *annotateVideo* service, except that two videos are declared instead of only one video as shown in the following JSON excerpt:

```
{
   "video1": {
      "url": "http://nlp-easytv.oeg-upm.net/video/es/2.mp4",
      "nls": "aire",
      "sls": "aire" ,
      "duration": "00:05" ,
      "language": "es" ,
                  "segments": [{
                  "order": "1",
                  "start": "00:00",
                  "end" : "00:05",
                  "content" : "aire" }]
    },
      "video2": {
      "url": "http://nlp-easytv.oeg-upm.net/video/en/2.mp4",
      "nls": "air",
      "sls": "air" ,
      "duration": "00:03" ,
      "language": "en" ,
                  "segments": [{
                  "order": "1",
                  "start": "00:00",
                  "end" : "00:03",
                  "content" : "air" } ]
    }
}
```

4.4. Implemented Services

The EasyTV-annotator-webservice provides a list of services to interact with the Sign Language Ontology. As introduced before, the two principal ones are the services related with the population of the ontology with Sign Language videos:

- **annotateVideo**: to annotate and populate the ontology with a Sign Language video.
- **annotateTranslatedVideos**: to annotate and populate the ontology with two Sign Language videos that are translations in different languages.

Moreover, other services have been developed to work with the ontology. The list of services are:

- **verifyTranslations**: to modify the value of the accuracy between a pair of translations. This is modified by an user through the crowdsourcing application.
- **getTranslation:** to retrieve a translation of a Sign Language video in a particular language.
- **getAllGraphs**: to retrieve the list of graphs created in the ontology. Each graph corresponds to a Sign Language video.
- **getAllTranslations**: to retrieve the list of translations created in the ontology between Sign Language videos.
- **getAllVideos:** to retrieve the list of videos created in the ontology.
- **getAllVideosOfLang:** to retrieve the list of videos created in the ontology in a particular language.
- **getGraphOfVideo**: to get the graph of a particular Sign Language video.
- **getVideosOfConcept**: to get the Sign Language video of a particular concept.
- **deleteGraph:** to delete a particular graph in the ontology.
- **deleteGraphFromVideoURL:** to delete a particular graph in the ontology using the URL of the Sign Language video.

To visualize all the services and the required input to users, a swagger interface has been included in the web service. Swagger shows the services in a web page as Figure 10.

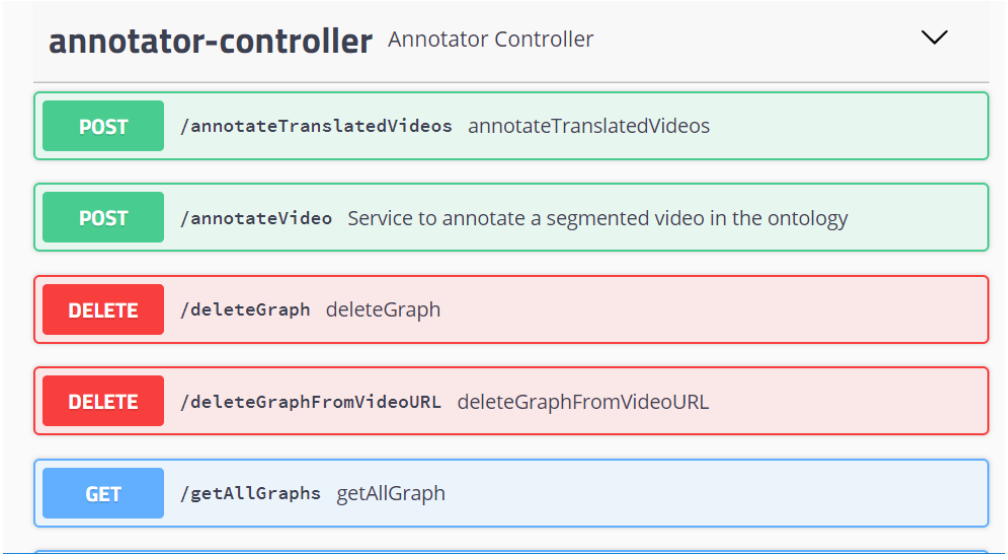

Figure 10. Swagger Interface

4.5. Deployment

The project is composed of different modules that are coded in different JAVA projects. The main module, the **EasyTV Annotator Web Service**, joins together all the other modules as libraries. In the context of the project, the management of the needed libraries is given by Maven. Each project

contains a POM file where all the needed libraries and the compilation properties are declared. Thus, the compilation of the other developed libraries before the web service is needed to be available in the Maven local repository.

Maven uses its own central repository to access the required libraries declared in the POM file and download them to the local repository. However, not all the libraries are available on the central repository. For these particular cases, a folder (**lib**) with all the necessary libraries, compiled in jar files, are included. Moreover, in the POM files, the instructions to include these libraries to the local Maven repo are also included.

The compilation result of the main module, **EasyTV Annotator Web Service**, result is a .war (in the last version of the code, the file has the extension .war.original) file which can be deployed in a web content server. In the context of the project, the web content server has been Apache Tomcat (version 9). The war is allocated in the folder "*webapps*" and it is automatically deployed when the server is started. The deployed web service is configured through the configuration file (allocated in *easytv-annotator\WEB-INF\classes* once it has been deployed) to reference to the different resources needed for each module (**EasyTV resources**). Thus, if the location of the module **EasyTV resources** is changed, the configuration file has to me updated with the new location of this module.

The web services is public from the URL: http://api.easytv.linkeddata.es/easytv-annotator/swagger-ui.html#/

4.6. Data population

Currently, the Sign Language ontology has been populated with a set of Sign Language videos representing concepts of the weather domain in four languages (English, Spanish, Italian and Greek). This set of concepts is described in Table 2. The concepts have been included though the web service and populated into the ontology.

The videos for Spanish have been provided by recordings of CERTH by Emilio Ferreiro Lago (CNSE) in the context of the project. The videos in English, Italian and Greek have been extracted from the web SpreadTheSign (SpreadTheSign.com).

Table 2. List of Sign Language concepts in the ontology

Language	Nº Videos
English	31
Spanish	21
Italian	29
Greek	12

5. MULTI-LANGUAGE SUBTITLES

Most of today's TV programs are available in only one audio language. However, the language spoken in the program content constitutes a huge barrier for people who do not understand it. This is especially problematic for those countries with more than one official language (like in Spain, where Castilian, Catalan, Bask and Galician languages are official) or with an important level of immigrant population (e.g. the Arabic in Spain). As an example in Catalonia with 7M of inhabitants, 0.5M don't understand Catalan. Providing multiple-language subtitles for the same content can be a vehicle of social integration and promote social cohesion.

EasyTV multi-language subtitle module was born with this aim, to help the social integration of immigrants and tourists, allowing to understand the local TV programs in the native language while, simultaneously, having an opportunity to learn Catalan.

At the same time, a second but not less important objective of EasyTV was to increase the number and quality of subtitle production for everyone, without a linear increase of costs. So, to define this objective in one sentence, to do more and better with less.

Having multiple-language subtitles will also allow to increase the number of spoken languages for the content through audio-subtitles, which can also help people with literacy issues, or have problems when reading subtitles.

In the next sections, we will describe the implementation of the EasyTV multi-language module, from the description of the whole workflow to the detailed definition of every process necessary to complete it from production to final users consuming the content.

5.1. Content Production

Each broadcaster has its particularities and production systems. Depending on the size of the company, they have their own-development, market based systems or they outsource for some processes. Nevertheless, we can remark a common element that is a Media Asset Manager (MAM), the software solution used to manage high-volume video and multimedia files. Assuming a MAM with multi-language support, the integration with the EasyTV platform will be easier.

When two separate and independent systems need to communicate, there is always the requirement to implement some kind of module or gateway that allows broadcaster systems to communicate with the EasyTV platform, to talk a 'common language' or protocol that allow the broadcaster or content owner to make requests, commission or consult tasks on the EasyTV platform.

The EasyTV platform is developed in a Docker virtual machine with a Web interface, so it was recommended to implement and communicate through an API (application programming interface), and develop the API methods to request and gather the desired assets and tasks.

In this way, the Content Owner MAM can communicate with the EasyTV platform, request the tasks, upload and download the required files, such as video assets or metadata, and finally index all the information to be ready for OTT publishing or broadcasting.

In the case of CCMA, the internal publication chain was not originally multi-language compatible and required an extra adaptation. To understand better the work performed, more details are given in the following sections.

5.1.1. Subtitle production chain

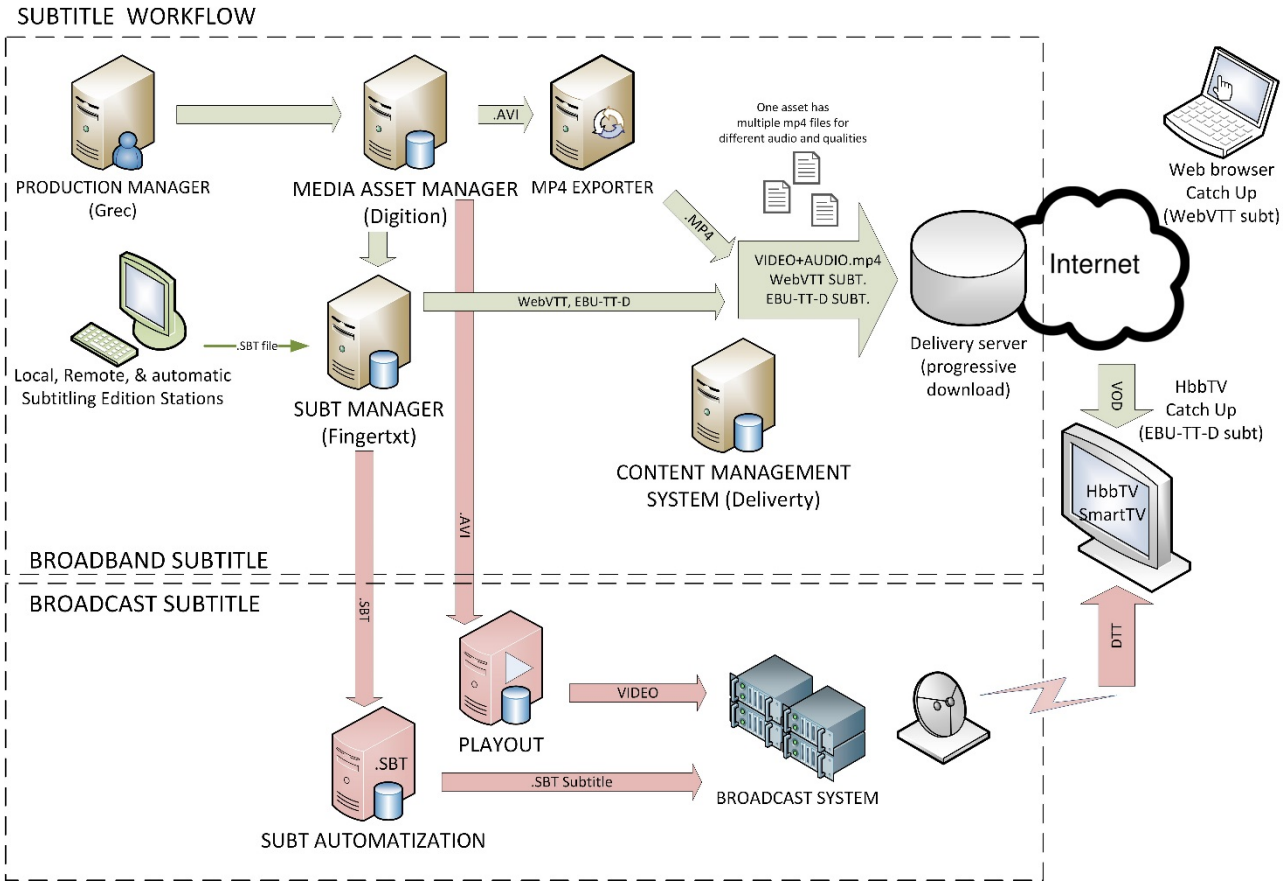

Figure 11. Subtitle workflow.

CCMA internal subtitle production chain is composed in several modules as shown in Figure 11:

- **Production Manager** – this module is responsible of managing all contents that will be broadcasted/broadbanded, it allows to define the requirements for every content, including the rights to publish this content for every platform, or the required accessibility contents to be included, like subtitles or audio description. This module has been adapted to support multi-language subtitles.
- **Subtitle Manager (Fingertext – ACM Accessibility Content Manager)** – it's not only a repository for accessibility metadata like subtitles or audio description, but it also allows to manage, import and export the metadata in several formats. The internal subtitles format is a proprietary format called .SBT but it can import and export other subtitle formats like WebVTT, EBU-TT-D or TTAF). It has been adapted and improved to support multiple language subtitles for one video asset. The audio description format is also a proprietary one.
- **Local, Remote & Automatic subtitling edition stations** – At CCMA , subtitle authoring is mostly outsourced and also produced with the help of semi-automated systems that allow to achieve high quality subtitles mainly in Catalan. The EasyTV platform enables to extend multi-language subtitling tasks using specially developed for this project, such as EasyTV integration module, and the Subtitle production module from the EasyTV Platform.
- **Media Asset Management (Digition)** – This is the CCMA MAM and keeps all audiovisual productions preserved, meanwhile the accessibility content is stored separately in the Subtitle Manager (Fingertext).

- **Web Content Management System (WCM-Deliverty)** – This module is the CCMA software application that is used to manage the creation and modification of digital content for web portal, device apps and digital services like HbbTV.

In the graphic shown (Figure 11) there are two independent chain, one is called “broadband subtitle” while the other is called “broadcast subtitle”. “Broadband subtitle” is the service that will allow to distribute the subtitles for Live & VoD content through internet, while “Broadcast subtitle” shows the workflow that allows to publish the real-time subtitles that are shown on CCMA DTT channels.

We will explain, in the next chapters, how CCMA has adapted its workflows, from the request on the Easy TV platform for the creation of new multi-language subtitles in the "Production Manager" module, as well as the collection of the finalized and revised subtitles that will be stored in the Subtitle Manager repository (Fingertext) and finishing with the publication and distribution of the subtitles to the end user.

5.1.2. **Multi-language subtitle production**

Starting from the current production systems CCMA has developed workflows (as can be seen in Figure 12) to request subtitle translation tasks to the EasyTV platform and receive the outcome.

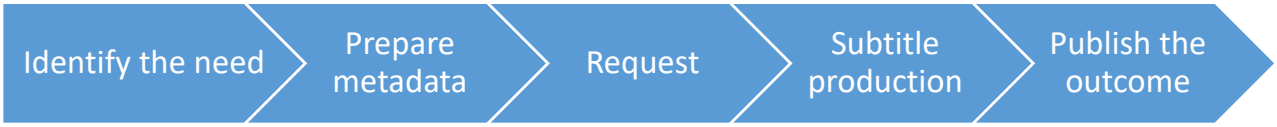

Figure 12. Subtitle production pipeline.

Production Manager (Grec) helps linear broadcast schedulers to manage the programme catalogue, they can modify each production and mark the ones that need multi-language subtitles. This action informs the EasyTV Integration Module, the in-house CCMA’s component in charge of managing the subtitles translation requests. The role of the EasyTV Integration Module is to prepare the necessary metadata and store the EasyTV outcome.

The general scheme, presented in Figure 13, is comprised of the following components:

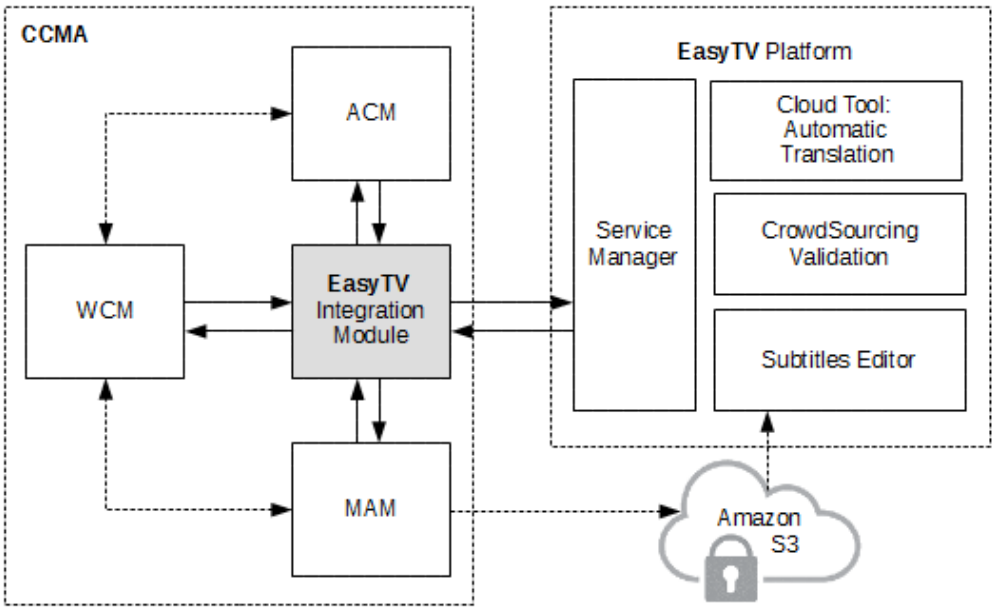

Figure 13. Multi-language subtitle production Overview.

In order to become a candidate to be requested to the EasyTV translation platform, a video must accomplish the following requirements:

- Video content metadata initialised (title, synopsis).
- Available Catalan subtitles with a source reference with accurate timecodes.
- Watermarked low-resolution video version ready to be uploaded to EasyTV platform.

The original production material needs to be protected against piracy, so that is why it is required to create a low quality watermarked version of the video to work with EasyTV platform.

An engine prepares the watermarked videos tailored for EasyTV editing tool, using the following basic parameters:

- h.264/mp4 coded
- Resolution 640x360
- Single Definition progressive
- Average bit rate about 300-400kbps
- Watermarked with EasyTV logo

For security reasons, the low quality content is also published into Amazon S3 storage with a private cloud URL, with the aim to protect it against possible early broadcasting (Figure 14). Only the subtitle web-based tool has the required permission to access to that content.

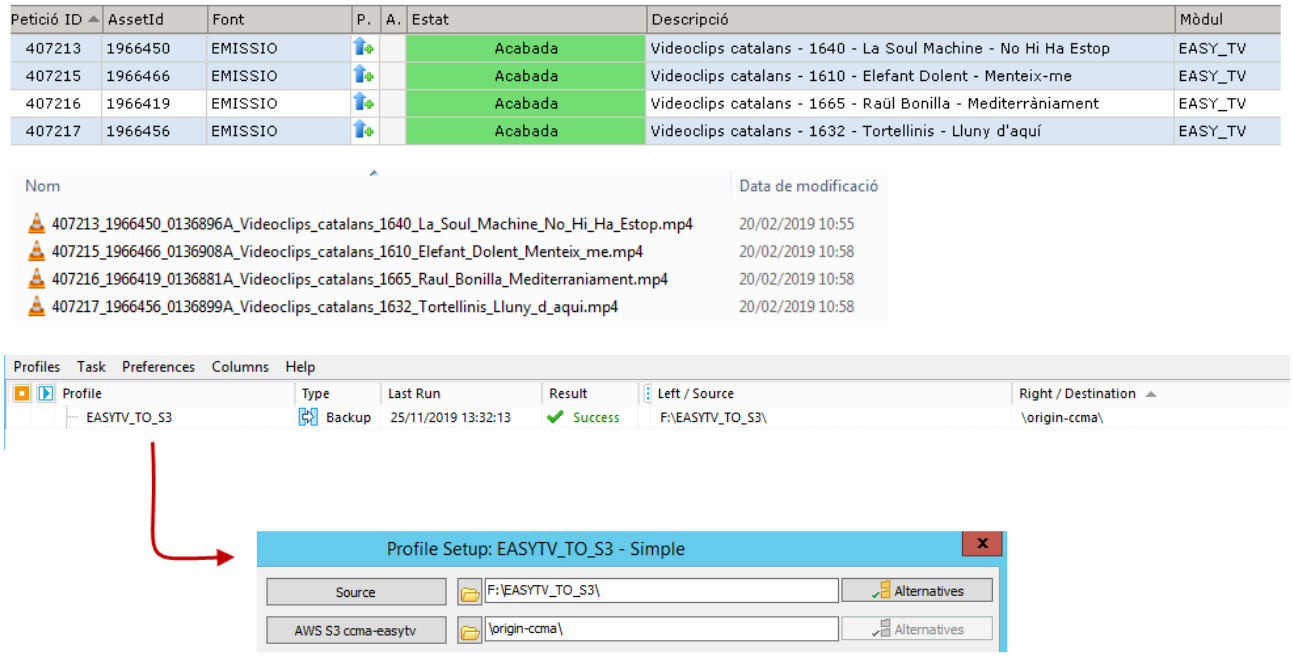

Figure 14. Low-resolution exporter and Amazon S3 uploader.

Finally, the Integration Module requests the Service Manager to register a new translation job into the EasyTV Platform, sending the Catalan Subtitles, video and metadata.

Afterwards, the EasyTV Integration Module checks periodically the status of all processing jobs sent to the Service Manager. In case a translation job is completed, the Integration Module gets the new subtitles metadata generated from the EasyTV Platform, registers them in the Accessibility Content Manager repository and publish them to the consumers.

Since we have detailed the process until the request to the platform, next section gives insights of how the EasyTV platform handles the request to generate the multi-language subtitles.

5.1.3. **Subtitles Production Module**

The Subtitles Production Module (SPM) is a module of the EasyTV Platform. It is composed by the back-end and front-end editing tool. Its goal is to generate the subtitles translated to the requested language. In addition, it interacts with other components such as the Service Manager (platform orchestrator) and Crowdsourcing Platform (CP).

SPM receives subtitling requests from Service Manager and prepares a first version of the subtitles translated through cloud service translation tools. However, to improve the accuracy and quality of subtitles this automated translation requires a human revision to analyse the context and improve the translation.

Some languages are not available from automatic translation services. In those cases where automatic translation is not possible, the subtitling tool allows also authoring new subtitles from the scratch.

EasyTV crowdsourcing platform plays the role to manage reviewers and evaluators, in order to complete the work giving the required access to the subtitle editor UI.

To follow better the case, the implied roles are briefly summarized in next lines, although they were described on D1.2.

Three different roles are defined to manage the CP, according to the confidence level awarded: administrator, reviewer and evaluator. Each user manages different permissions:

- The **Administrator** manages the platform, with full access and control.
- The key function of reviewers and evaluators is to access a list of pending and assigned jobs, and edit subtitles.
- The **Reviewer** can pick an assigned or pending job and check if subtitles are properly translated by the automatic system, or translate them from scratch. If the automated proposal contains errors, it must be fixed and refined by the reviewer.
- The **Evaluator** supervises the work done by the Reviewer. Small errors can be directly fixed by them, otherwise the work could be rejected marking the conflicts and re-assigned to another reviewer if necessary.
- Finally, once the evaluator agrees, the task is finished and ready to be published.

Figure 15 shows the critical path of the information, in a success scenario. Obviously, a real case would be more complex, and would require more status updates, replies, etc. This path covers from the initial request from broadcaster or content owner, the delegation to the corresponding module and the interaction from a reviewer and an evaluator. Once finished and approved for being published, the assets become available to be recovered from the content owner.

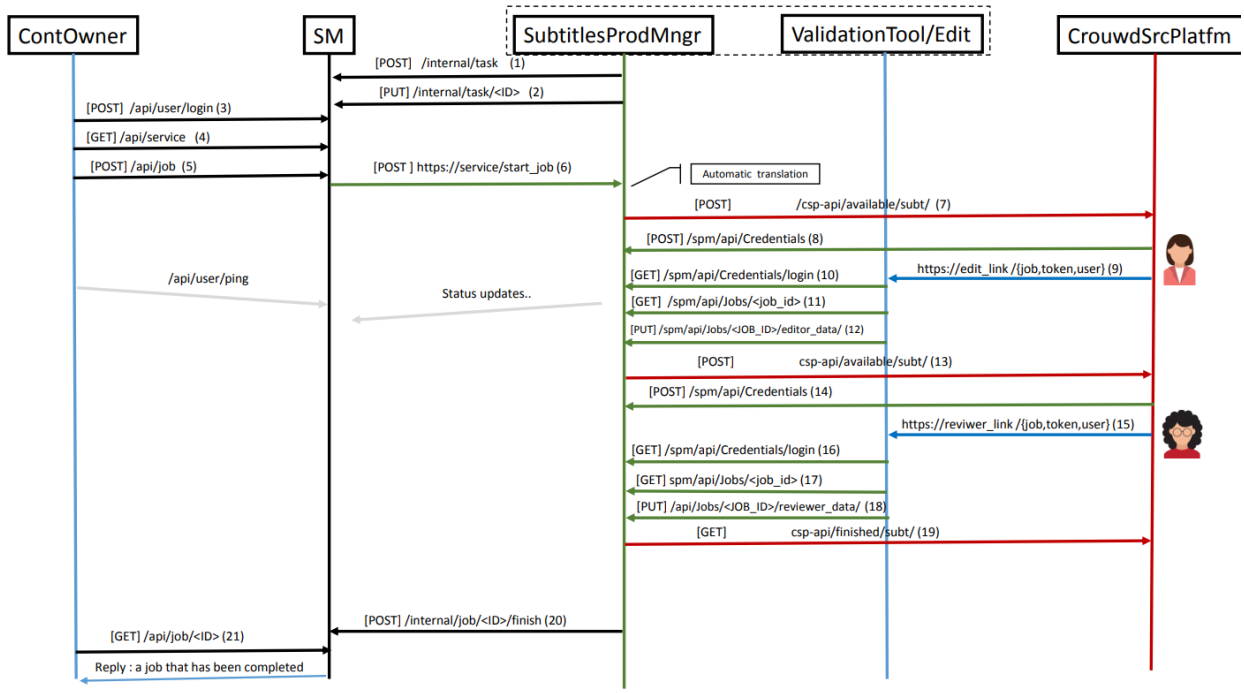

Figure 15. Multi-language Subtitle translation end-to-end workflow.

5.1.3.1 Back-end

The SPM back-end consists of three applications and a database as shown in Figure 16:

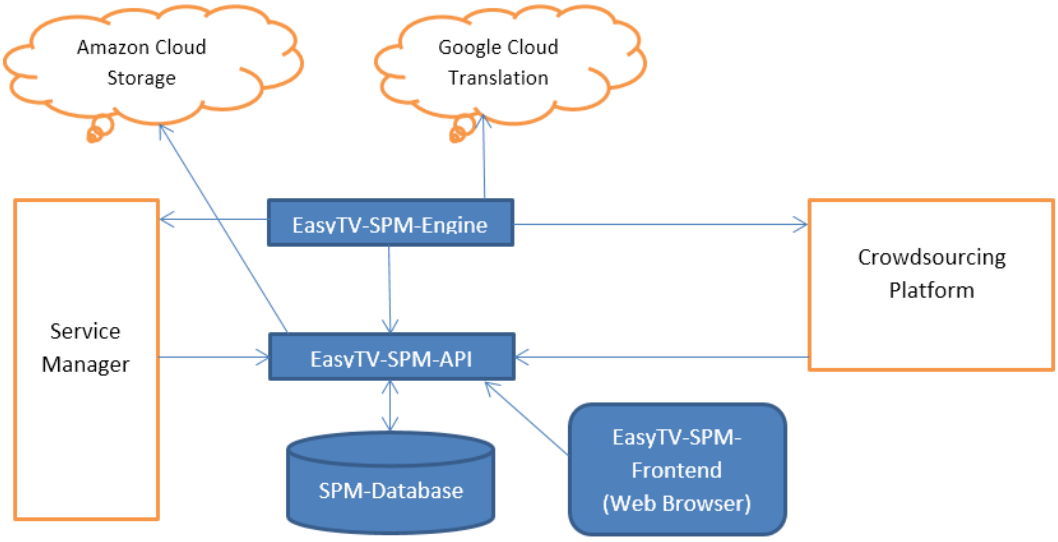

Figure 16. SPM back-end

- **SPM-Database**: Repository where the translation tasks (Jobs) and other module configuration data are stored. Allows storing the required information for each "job":
  - Origin and destination language codes
  - Program Title
  - Synopsis
  - Subtitles (EBU-TT-D format)
  - MP4 video file path of the content, stored on the Amazon cloud servers.
  - Deadline (airing date)
  - Expiration date
- **EasyTV-SPM-API**: REST API that acts as back-end to manage translation tasks (Jobs) by accessing the SPM database. It also provides temporary video URLs to access the content during translation tasks, which are obtained through calls to the Amazon API web services (AMAZONWS).
- **EasyTV-SPM-Engine**: console application that runs periodically. It communicates with the EasyTV-SPM-API and manages automatic translation of subtitles through the Google Translation API. It also manages the changes in the status of the translation tasks and communicates with the EasyTV-Service-Manager (SM) and crowdsourcing platform (CRWD) through their respective APIs.

  The engine has defined a list of states for each job. It helps to coordinate the communication with other modules and follows the defined workflow. The list of states is presented in Table 3.

- **EasyTV-SPM-Frontend**: HTML-JavaScript web application for the edition and review of subtitles. It communicates with the EasyTV-SPM-API API.

Table 3: SPM - internal job states

state	description
Awaiting For Automatic Translation	• Pending to send Google translation request. Title, synopsis and subtitle texts translated.
Auto Translated	• A reviewer can start working. • The Service Manager and the crowdsourcing platform are informed about the new status.
Awaiting For Edition	• A reviewer user of the crowdsourcing platform will be assigned to the job. • When a user is assigned the job, the crowdsourcing platform calls the API to save credentials in the SPM-Database • One-token Credentials for single use only.
On Edition	• An authenticated reviewer obtains the associated job data and edits the content. • The system saves partial editing results.
On Edition Saved	• The crowdsourcing platform and the Service Manager are informed about the progress of the edition.
Edited	• When the user finishes the edition, the crowdsourcing platform is informed that the Job is available for a evaluator
Awaiting For Revision	• A evaluator user of the crowdsourcing platform will be assigned to the job • When a user is assigned the job, the crowdsourcing platform calls the API to save credentials in the SPM-Database • One-token Credentials for single use only.
On Revision	• An authenticated evaluator obtains the associated job data and edits the content. • The system saves partial editing results.
On Revision Saved	• The crowdsourcing platform and the Service Manager are informed about the progress of the evaluation.
Revised	• When the evaluator finishes the revision, the crowdsourcing platform is informed.
Crwd End Notified	• The translated items are brought back to the Service Manager.
Finished	• End state.

Rejected	• An evaluator has rejected a job. Crowdsourcing platform and the Service Manager are informed that the job has been rejected.
Cancel Requested	• The API receives a DELETE action.
Cancelled	• End state.

Back-end Technical implementation information

This solution is multiplatform and can be deployed on Windows, Linux and OSX platforms. It can also be deployed on the Docker container system.

- *EasyTV-SPM-API*
  - Web application developed with C # language with Microsoft ASPNET CORE technology.
  - Provides a REST type API that allows to create and manage tasks or subtitle translation jobs.
  - SQLite database to save the status of translation tasks, credentials, configuration, etc.
  - Amazon Web services (AWS) API client to obtain video URLs associated with a task with limited access time.
  - Documentation and API test website available Swagger [6]

- *EasyTV-SPM-Engine*
  - Console application developed with VB.NET language with Microsoft NETCORE technology.
  - Interact with other EasyTV modules through:
    - EasyTV-SPM-API.
    - EasyTV Service Manager module API.
    - Crowdsourcing platform module API.
    - Google Translation Services API.
  - It has three operating modes, selectable via command line options:
    - As a task programmed through CRON or AT: In each execution, the status of Jobs is checked and executes the necessary actions.
    - As a program that is running permanently. In this mode an internal timer is activated. Every 15 seconds, the status of the Jobs is checked and it executes the necessary actions.
    - Interactive mode: Shows a menu that allows to execute different commands.

5.1.3.2 Front-end editing tool

The editing tool is exclusively accessed from the crowdsourcing platform, once a user is registered. The CP is in charge to manage users, and basically assign users to available jobs.

Reviewer view:

[6] https://spm-api.easytv.eng.it/index.html

Once the reviewer has chosen or has been assigned to a job, the access to the subtitle editor is started as shown in Figure 17:

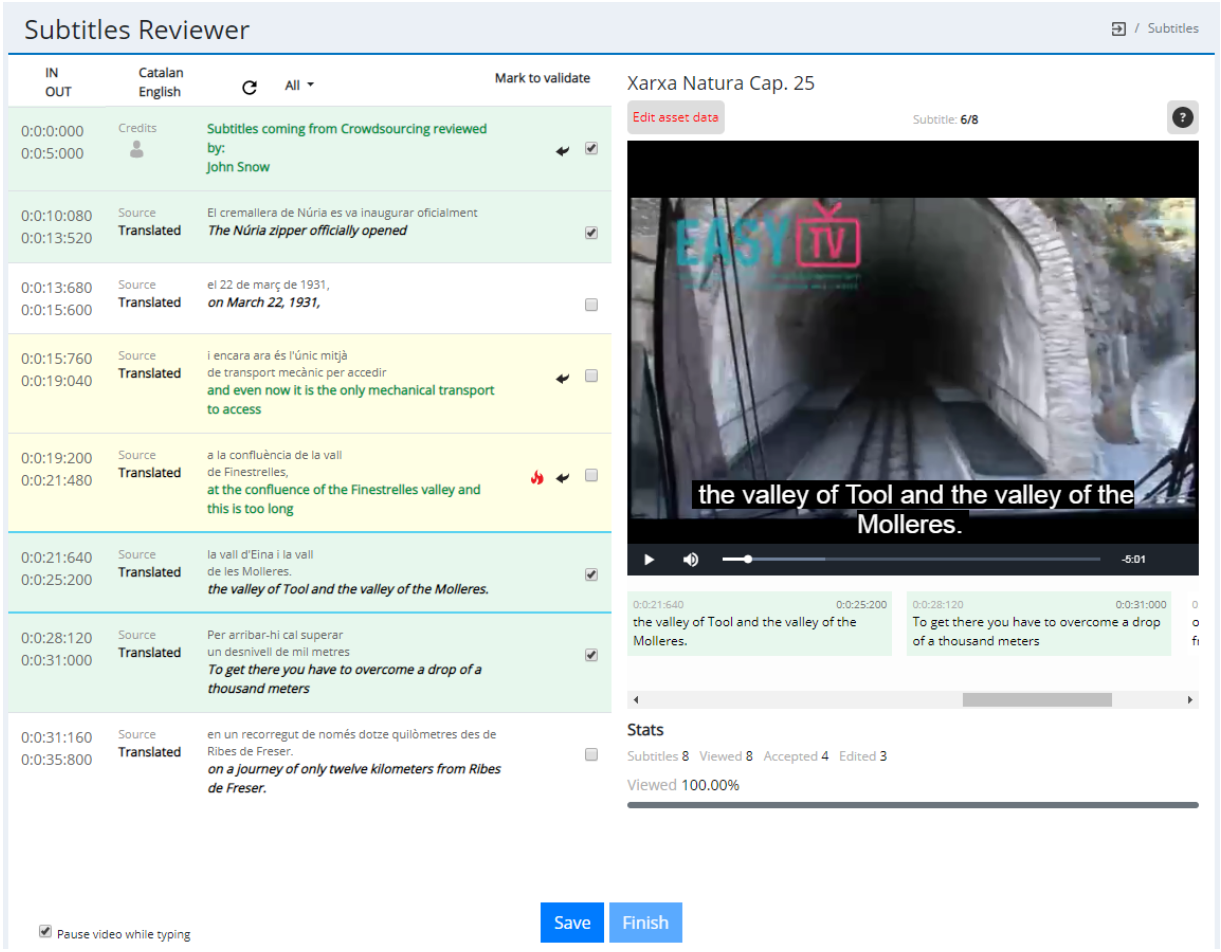

Figure 17: SPM Editing tool UI - Reviewer

This screen is primarily comprised by a list of subtitles at the left side and the player and stats at the right. Each subtitle is located in a box containing various information: its associated input and output timecodes, the original subtitle (source), and the translation generated by the cloud service (translated) shown below.

In each subtitle box there are also several elements:

- **Check box**: To indicate if translation is correct.
- **Edit text field**: It appears when a subtitle is clicked and enables the edition.
- **Undo button**: It allows undoing changes if the subtitle has been modified.
- **Text overflow icon**: It warms the user about the excessive length of a text in relation to the time length this is going to be displayed.
- **Add/Remove line buttons**: When the user clicks a subtitle box to edit it, a plus or minus button appears, depending if the subtitle is a single line one, or double line.

Besides the editing tools, there are some more tools to help with the whole process. The drop down menu enables to choose which list of subtitles are shown:

- All – all subtitles
- Accepted – Subtitles that are accepted (so reviewed).
- Edited – Subtitles edited.
- Pending – Subtitles with edition pending.
- Conflict – Subtitles that are rejected by the evaluator and need to be reviewed again.
- Empty – Subtitles with no text.
- Text overflow – Subtitles too long for the space assigned that need to be shortened.

The reviewer can access also to the stats section to check the progress status of the task with some information added. The fields shown are the number of existing, viewed, accepted and edited subtitles and the percentage and progress bar for each option. To change the stats bar between fields, the user just has to place the mouse over each one and the bar will change its value accordingly.

The video player displays synchronously the edited subtitles at the same time they are being edited or modified. This means that anytime the reviewer clicks a subtitle, the player moves forward or backward to adjust to the selected moment. The same happens when the video is played; so subtitles boxes are focused progressively synchronously to the video. The player has also a control option to allow or not the possibility of pausing the video while the user is typing, to ease the edition of subtitles.

The timeline also shows the previous, present and next subtitle, allowing the reviewer to jump directly to this subtitle just clicking on it.

There is a help modal (Figure 18) with some tips to help the new users, accessible through the help button above the player box. This help modal shows a legend that explains the meaning of every icon, the colour code status of the texts, and also the keyboard shortcuts to interact with the editor.

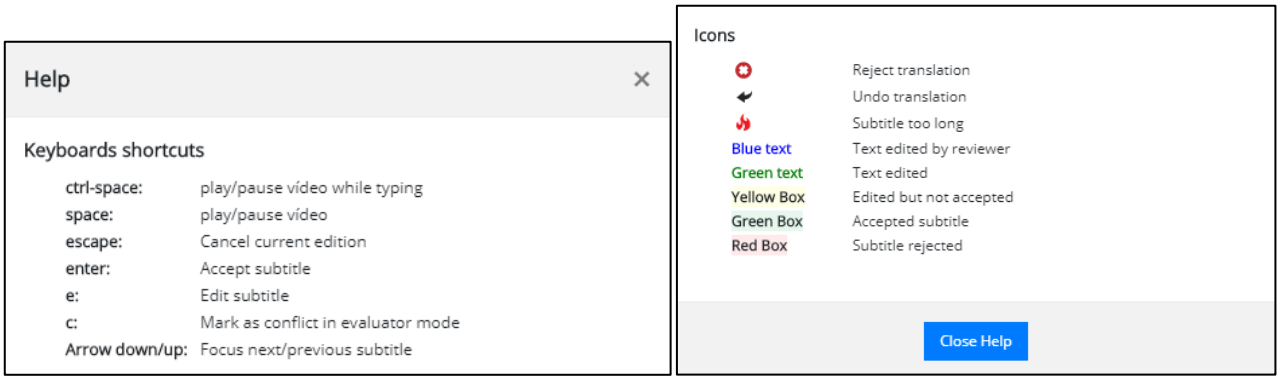

Figure 18: SPM Editing tool UI – help legend

To finalize the edition, all the subtitles must be accepted regardless if they have been edited or not. If the reviewer does not accept all the subtitles, including the title and description, the finish button keeps disabled and the job cannot be finished.

To validate the title and synopsis of each asset, there is a specific button called “Edit Asset Data”. This button shows a modal with the title and synopsis and the possibility to accept or edit each one.

In addition, there is also the option to save the work without finishing the job, just to make sure the changes are saved. This can be done at any moment by clicking in the corresponding SAVE button.

Evaluator view:

Once a task has been finished by the reviewer, an evaluator can get it to evaluate and improve the accuracy, granting a quality assurance before being accepted to be broadcasted (Figure 19). The main screen of the evaluator is very similar to the one used by the reviewer:

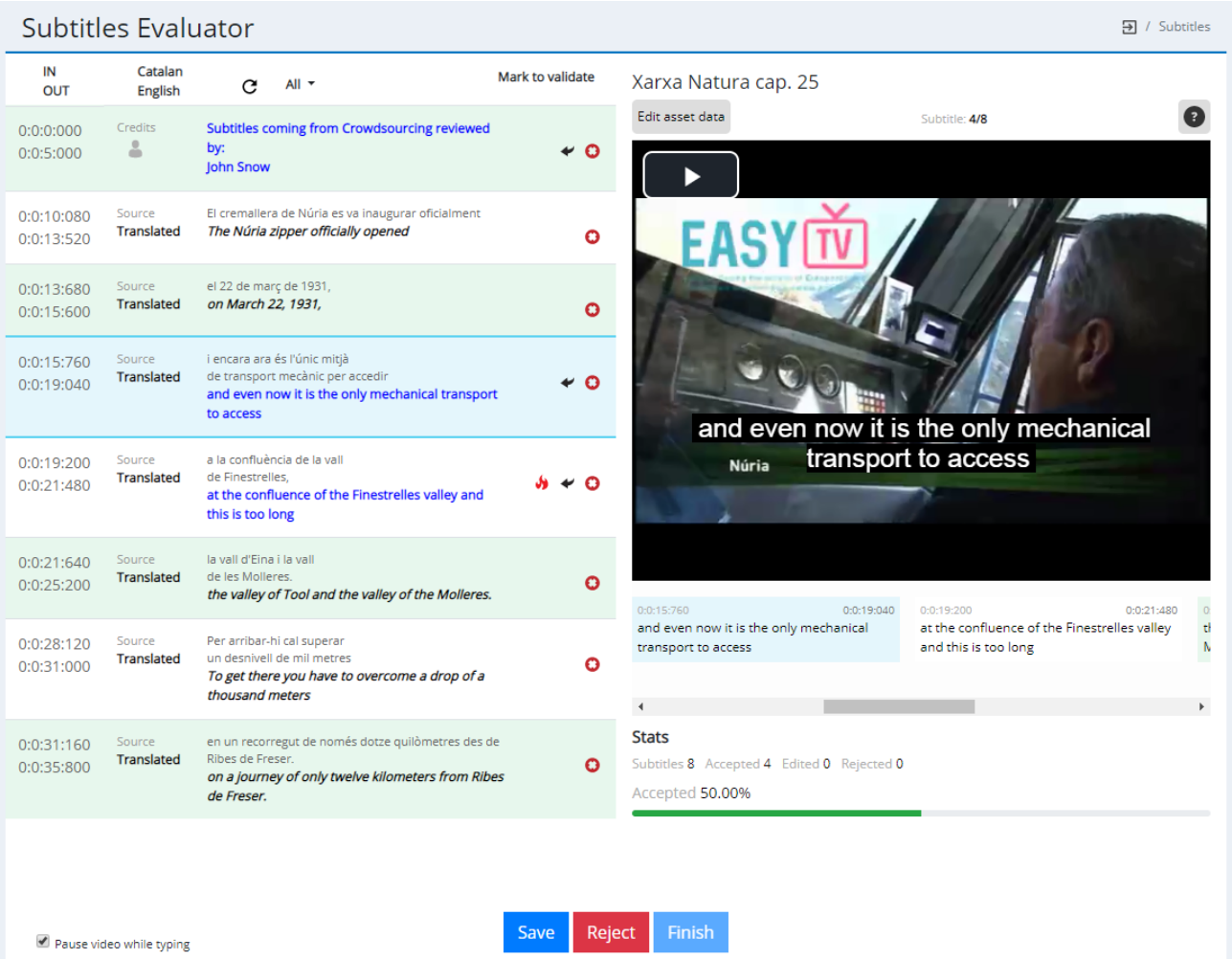

Figure 19: SPM Editing tool UI - Evaluator

The evaluator must check that the reviewer job is correct. This is done by checking all the subtitles list and marking them as accepted.

Evaluator Options

The evaluator can perform three main actions in each subtitle job: accept it, edit minor changes and then accept it, or reject it (asking some reviewer to rework on it).

Marking conflicts is used for the rejecting process. If the evaluator finds that the translation is not good enough, he/she can reject the job and send it back to the reviewer with the possibility to add some tips about how to improve the translation. In this case, the reviewer will receive the tips and will see all the conflict subtitles displayed in red to let him/her know which ones must be checked.

However, if everything is correct and all the subtitles are accepted (modified or not), the job can be

finished. Nevertheless, there cannot be any subtitle marked as conflict including the title and synopsis, which must be also accepted.

The editing workflow is very similar to the reviewer's, but there are some little changes when editing. First, the text shown in blue means that the reviewer has modified this subtitle. And the second one is that a subtitle cannot be in the "edited" state. If a subtitle is modified, it is set automatically as accepted because we assume that evaluator's editions are correct.

In order to get a general view of the task, the evaluator can check the stats where the accepted, edited and rejected subtitles are counted. This information is useful for the evaluator to decide when to fix the conflicted subtitles by himself and finish the task, and when to reject it.

- **Finish:** If the evaluator decides to finish a task, this finalises the whole circuit and the finished task will be available again for the content owner.

- **Reject:** Once a task is rejected by the evaluator, it goes back to the reviewer, who will receive a message in the crowdsourcing platform and an email with the reasons of the rejection and clues to improve the translation. With this information, the reviewer can modify and correct again all the subtitles, especially those ones that have a conflict, which are displayed in red to facilitate their location.

Technical implementation details

The editing tool is a webpage designed following the 'Single Page Application' pattern, so the user does not have to communicate regularly to request pages from the server and allows him/her to have a more fluent and dynamic experience while working on the edition. The underlying architecture is MVC (Model-view-controller) based on the Backbone library, which enables to develop this structure with JavaScript. Other tools and frameworks are used, such as *Underscore* for variable typing or *Handlebars* for creating the HTML templates.

The player consists of two layers: one for the video itself rendered with the *VideoJS* plugin, and another for subtitling using the subtitles render plugin.

The Editor Tool is accessible only from the Crowdsourcing platform where is embedded in an *iframe*. It uses "postMessage" function to dialogue, when the iframe needs to be closed.

5.2. Consumption of Multi-language subtitles

In previous sections, we have explained how EasyTV Project has developed a complete solution to allow the authoring of multi-language subtitles and how this complete solution could be easily integrated with the broadcaster system and workflows.

The broadcaster is usually responsible of the storage, management and deployment of these multi-language subtitles to the end user.

This section explains how the developed applications allow users to enjoy the content with multi-language subtitles, using standard protocols, where possible, or proposing the best solutions in those environments where it is still difficult to find adequate implementations.

5.2.1. HbbTV

In 2010, CCMA started its HbbTV service for SmartTV, the European standard for TV hybrid (broadcast - broadband) applications. The service, named "TV3alacarta", opened a new way of TV

program consumption as VoD on SmartTV, where the users could enjoy most of TV3 programs at any time in their HbbTV compatible SmartTV.

The participation of CCMA in the Hbb4all European project allowed for improving the HbbTV service with customized subtitle and audio-description services, which were deployed in February 2017. CCMA has been always very sensitized to the difficulties of impaired people for accessing to the audiovisual content, so all CCMA services aim to be accessible to all users, whether they are consumed through TV's, PC's or mobile devices like smartphones.

While very new HbbTV v2.0.1 SmartTV support EBU-TT-D standard subtitles, HbbTV v1.1 and HbbTV 1.5 have no support for subtitles (Figure 20). Hbb4all results allowed incorporating EBU-TT-D subtitles to all HbbTV versions, thanks to an accurate and successful development that provided an analysis of the EBU-TT-D protocol to re-generate the subtitles as HTML object, required to render it over the video on any HbbTV screen.

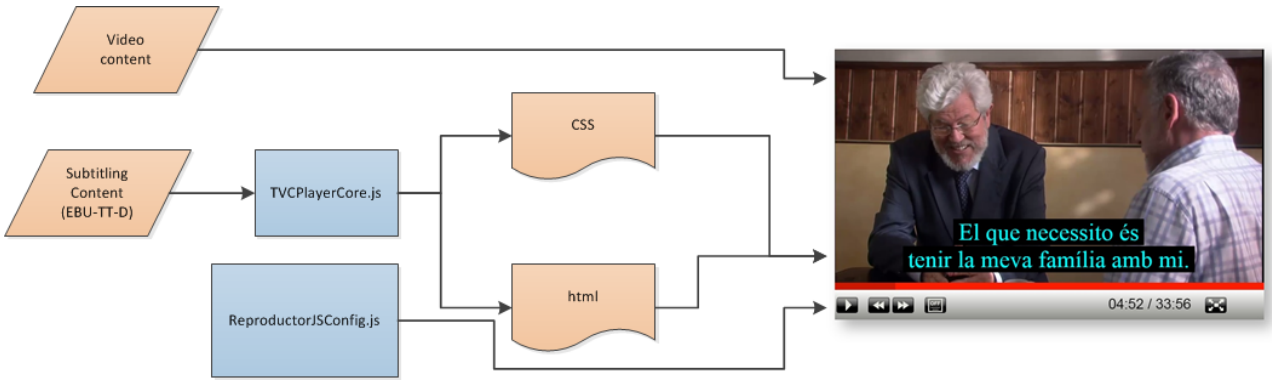

Figure 20. EBU-TT-D subtitles development for HbbT v1.1 & v1.5

The Hbb4All solution was improved with customized subtitling which allowed users to change font size, position and background and keep their own personalization for future use.

Since then, "TV3alacarta" HbbTV service only offered subtitles in Catalan language when available, and the end user could choose the best personalized configuration through an easy to use customization menu (Figure 21).

Figure 21. Subtitle default conf. / Customization menu / Customized subtitle

EasyTV multi-language subtitles

One of the goals of EasyTV is to offer **multiple language subtitles** to the end user while allowing the available tools to personalize the size, position and background of the subtitles, for VoD contents available on "TV3alacarta" service.

Moreover, after all the effort on the production stage, and taking advantage of the fact that subtitles are required before being broadcast, we've tried to maximize the exploitation of the assets and also to enable multi-language subtitles not only for VoD content but also for pre-recorded programs that are broadcasted through linear TV.

As consequence, the applications' settings were re-designed to host multi-language subtitles and other accessibility settings.

5.2.1.1 Multi-language Subtitles on pre-recorded content Broadcasted as Linear TV

The current document has presented the great challenge assumed in EasyTV for authoring multilanguage subtitles that remain available at the Accessibility Content Manager.

This section describes the developments carried out on the broadcaster premises to offer multi-language subtitles synchronously to pre-recorded content broadcasted as linear TV keeping the best user experience in order to offer accessibility not only to people with hearing issues but also to people with other needs. The developments affect all the processes involved from the production to management and storage, and the final distribution of the content.

The problem

The radiofrequency bandwidth for TV broadcasters has experienced significant reductions over the last years, due to bandwidth required for 4G and 5G telecommunications infrastructures. Nowadays CCMA is broadcasting all its TV channels through only one RF Multiplex, including:

- TV3 HD Channel (main channel in FullHD)
- TV3 SD Channel (main channel in SD)
- C33/Super3 (cultural and kids channel)
- Esports3 (sports channel)
- 3/24 (24h News channel)
- 4 Catalunya Radio audio channels
- IB3 Balear Islands TV channel

Due to this situation, there is a bandwidth limitation for new services, like for example, multiple language subtitles broadcasted through DTT.

There are two methods to deploy subtitle service on DVB, which can be used for Terrestrial (DTT), Satellite and Cable Digital TV:

- **Subtitling by DVB teletext:** It is inherited from the teletext service for analogue television, where the information was embedded in the video vertical signal interval (VBI). This information, including subtitling, is digitally transmitted in a component of the DTT Transport Stream according to the EN 300 472 standard. The look of subtitles is exactly the same as the teletext look, as it's exactly the same service.
- **DVB subtitling:** consists of a graphic representation of the subtitles that is transmitted in a component of the DTT Transport Stream according to the EN 300 743 regulations [4]. It presents a series of advantages over teletext subtitling as shown in Figure 22:
    - Better definition than teletext font.
    - Control of the letter font by the broadcaster (as it is a bitmap or graphic representation).
    - It allows to include several subtitles per home service (for example, several languages).

Figure 22. Comparison between 'Teletext subtitles' (left) and 'DVB subtitles' (right)

DVB Teletext is limited to the Teletext font only, based on the original Teletext character set, so it would be impossible to deploy, for example, Arabic subtitles.

DVB subtitling allows to deploy any font and any language symbol, as soon as the broadcaster is able to represent the graphic conveniently, and allows multiple subtitles simultaneously, but it needs more bandwidth to send the bitmap (graphic representation) than the bandwidth needed for Teletext codes (AscII codes).

While DVB subtitling is the best option for Digital TV for quality and flexibility reasons, the current RF bandwidth limitations make this multi-language deployment difficult.

Nowadays CCMA is offering its subtitling services over DTT according to the DVB subtitling standard EN 300 743 [4]. Unfortunately, the handicap of this solution is that the use of graphics implies a waste of bandwidth, so each subtitle stream requires about 100 Kbps for its transmission. Increasing the bandwidth for subtitles would imply to decrease the bandwidth for video and audio or, in other words, to decrease the quality of the broadcasted channels and, in the same way, decrease the user experience.

Therefore, it was necessary to find a solution to allow increase the number of subtitle services without growing the required RF bandwidth.

The solution

CCMA team studied the available technologies for sending subtitles in a lighter way which should allow to implement the multi-language subtitling with the use of a small amount of bitrate in the DTT channel and, at the same time, keep the highest font definition and quality. This work led us to a solution based on HbbTV.

HbbTV allows activating events synchronously to the RF TV reception thanks to a form of trigger called 'stream event' which is a signal received through RF Multiplex channel.

This means that once an HbbTV app is activated and associated to a TV channel, this app can keep listening to this 'stream event' which would include information to execute an action.

This behaviour is represented in Figure 23, where an HbbTV SmartTV receives a TV Channel through RF (DVB-T, DVB-S or DVB-C) and loads the HbbTV App through IP Internet connection, following the URL referenced within the RF channel information. The stream event broadcasted in the TV channel stream will activate some pre-programmed HbbTV app actions.

This approach needs less bitrate than using DVB subtitles, and it offers additional advantages such as the personalization of the subtitle presentation by the user: font size, position and use of background.

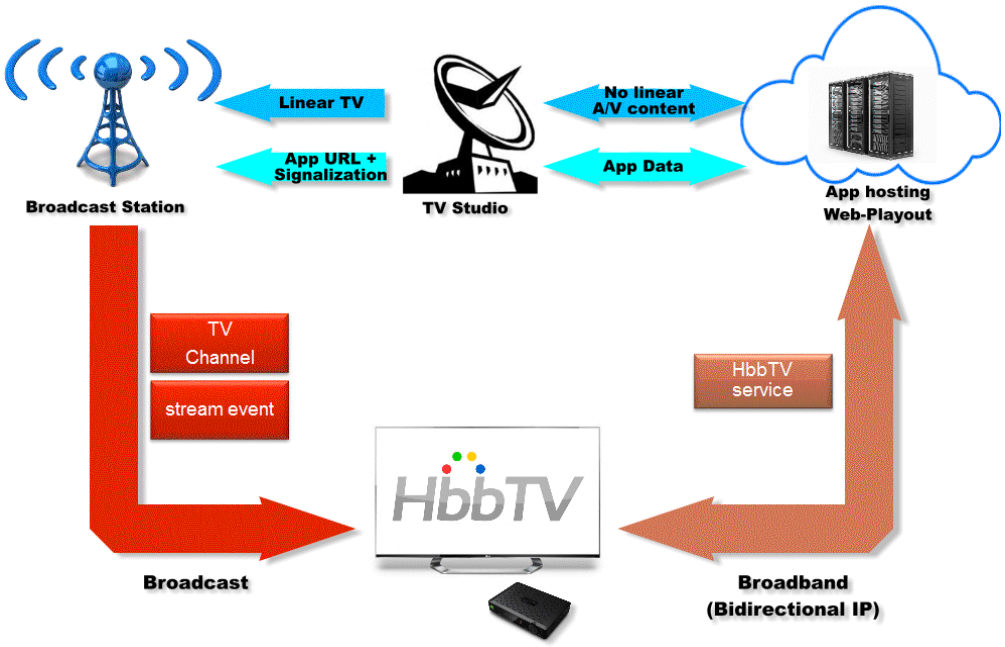

Figure 23. Hybrid reception on HbbTV SmartTV

Figure 24 explains how this action takes place in a time line, where the first row shows the TV program playing, the second row represents the TV signal where the various 'stream events' are also broadcasted, and the third and last row shows the actions of the HbbTV application:

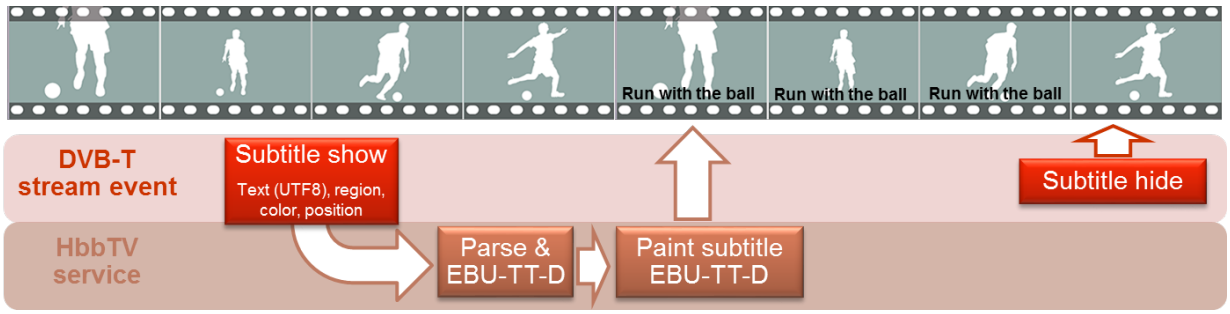

Figure 24. Subtitle solution using stream events and HbbTV app

When the user plays a new TV channel, the SmartTV will look for the HbbTV linked app (signalised in the DVB multiplex) and will load it through its IP Internet connection to start execution.

From there, if the user activates multi-language subtitles, every time that a 'stream event' is detected, the app will load the subtitling information included in it, will parse this information to generate an EBU-TT-D document and finally will re-generate the subtitles as an HTML object, required to render it over the video on the HbbTV screen.

It was important to keep the same EBU-TT-D subtitling standard, as it was already developed in

Hbb4all project for HbbTV[7] v1.0 and V1.5, and included as standard in HbbTV v2.0.1.

While EBU-TT-D is a good protocol for production and distribution, it is too big in terms of memory to be sent through 'stream events' with a limited payload and with narrow bandwidth. Therefore, a new light and proprietary protocol was designed, which only loads the most relevant information:

- Subtitle text (8-bit Unicode Transformation Format)
- Region
- Colour (white by default)
- Position (center by default)

Example of a 'simplified' subtitle:

```
{
"c": "P",
"p": "10 72 80 24",
"a": "c",
"i": [{
"c": "#FFFF00",
"t": "I, a més, vol proposar"
}, {
"c": "#FFFF00",
"t": "que em contractin. Seria genial,
eh?"
}   ]   }
```

Example of the same subtitle once translated to EBU-TT-D with the HbbTV app:

```
<tt:region xml:id="r0" tts:origin="2.50% 72.00%" tts:extent="95.00% 24.00%" tts:padding="0%" tts:displayAlign="after"
style="sr2" tts:writingMode="lrtb" tts:showBackground="whenActive" tts:overflow="visible" />
<tt:style   xml:id="sr2"   tts:fontFamily="Tiresias"   tts:fontSize="180%"   tts:lineHeight="normal"   tts:textAlign="center"
ebutts:linePadding="0.5c" tts:wrapOption="wrap" />
<tt:style   xml:id="ss2"   tts:fontStyle="normal"   tts:fontWeight="normal"   tts:textDecoration="none"   tts:color="#FFFF00"
tts:backgroundColor="#000000" />
<tt:p xml:id="p20" region="r0"begin="00:02:24.760" end="00:02:27.760">
<tt:span style="ss2">I, a més, vol proposar</tt:span>
<tt:br />
<tt:span style="ss2">que em contractin. Seria genial, eh?</tt:span>
</tt:p>
```

The light protocol designed to be transported with 'stream events' allows to reduce dramatically the weight of subtitles metadata, while maintaining the most relevant information to allow the regeneration of the subtitles on the HbbTV screen.

The average bandwidth required to carry five subtitles is approximately 11kbps, which is perfectly compatible with the limitations of the TV channel multiplex and allow the transport of multi-language subtitles.

Along the next sections, the implemented subtitling protocol over Stream Events will be explained, as well as the developments and integrations carried out to deploy this service in the TV3 DTT channel that belongs to CCMA.

[7] HbbTV specifications https://www.hbbtv.org/resource-library/specifications/

CCMA subtitling protocol over Stream Events

The Stream Events is a mechanism of messages for encapsulating private data into its payload and send them over DVB transmission systems based on Transport Stream carriage, including DVB-T. This mechanism allows distinguishing different types of messages through its '*Event_id*' identifier. This section describes the protocol defined by CCMA to implement a multi-language subtitling service based on these messages.

The protocol defines different types of message over Stream Events with the necessary information to allow an HbbTV app running in a Smart TV in the current tuned channel determining, without knowing prior information, which subtitling messages it has to listen to. There are three types of messages:

- **'Channel Info' messages:** this type of message is the entry point for the HbbTV application and for this reason it uses a predefined '*Event_id*' identifier. They carry the list of DTT services identifiers to let the HbbTV app know if the current tuned DTT service has any multi-language subtitle service associated and which '*Event_id*' is carrying the corresponding 'SBT Info' messages.
- **'SBT Info' messages:** this type of message specifies the details of the multi-language subtitling service in a specific DTT channel. For each subtitling component, it particularizes the language, the currently subtitling status and the '*Event_id*' carrying the corresponding 'SBT Data' messages.
- **'SBT Data' messages:** this type of message carry data of the subtitling corresponding to a specific DTT channel and language. There are three types of subtitling actions that could be executed by the HbbTV app as soon as the message is received:
    - To draw the received subtitle erasing all the content previously presented on screen
    - To draw the received subtitle without erasing the screen
    - To completely erase the screen.

The first version of the protocol implemented these messages as JSON-format [5] placed directly in the payload of a Stream Event due to the fact that the maximum allowed size of 245 bytes seemed enough for the purpose. However, we found during testing that some HbbTV devices do not work properly when a Stream Event message is split into two Transport Stream packets. This issue was solved by partitioning each JSON structure to be encapsulated in multiple Stream Event messages where each of them fits in a single Transport Stream packet, including a pair of header bytes for the sequence control.

The use of the JSON format has the required flexibility for including the necessary subtitling data, and also it has the advantage of the native support of UTF-8 encoding, that allows to include text in nearly any language such as Arabic. Anyway, the Stream Events mechanism has low space for data, so the subtitling protocol only includes the basic parameters provided in Table 4 and left to the HbbTV application other details of the presentation.

Table 4. Parameters of a subtitle message

Parameter	Usage
"r" (region)	It defines the region with 4 integers that represent a percentage of the screen, from left to right: • Horizontal position of the beginning of the region with respect to the left edge of the screen. • Vertical position of the beginning of the region with respect to the upper limit of the screen.

	• Width of the region with respect to the width of the screen. • Height of the region with respect to the height of the screen.
“a” (alignment)	Alignment of the text within the region with three possible values: • "l” Left • "c" Centred (default) • "r" Right
“i" (item)	Text element to be placed in a region.
“fc” (font colour)	Colour to draw this text item (default is white).

Subtitling service over Stream Events deployment

The Subtitling service over Stream Events has been integrated into the CCMA DTT broadcasting playout, following a workflow that is similar to the already existing one for DVB subtitling service. The publication chain is represented in Figure 25:

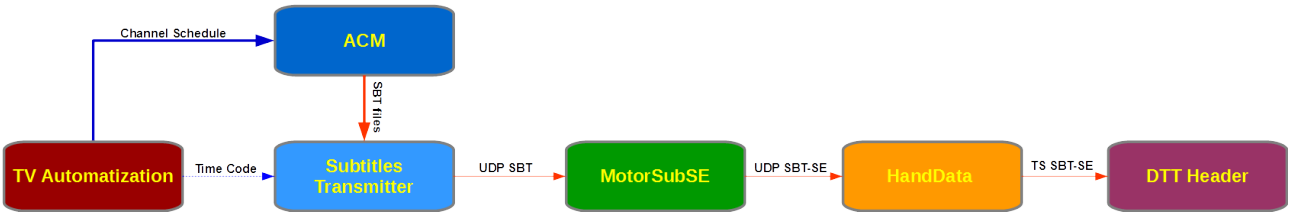

Figure 25. CCMA workflow of multi-language subtitling from ACM to DTT Broadcasting

The role of each element from the figure is:

- **TV automation:** this is the element that manages the broadcasting of contents over the DTT channel.
- **Accessibility Content Manager (ACM):** The ACM is the key element where the multi-language subtitles produced by the EasyTV crowdsourcing platform remain available. It is connected to the TV automation to get the schedule of the channel and to check which programs have attached subtitles and in which languages, in order to provide the associated subtitle files to the subtitle-transmitter equipment.
- **Subtitles-Transmitter:** this equipment sends each individual subtitle fragment extracted from a file towards multiple clients. The subtitling is synchronized with the video content by means of the time code also provided by the TV automation.
- **Engine for Subtitling over Stream Events (MotorSubSE):** the subtitling messages sent by the subtitles-transmitter equipment use a propietary protocol that needs to be transformed into the subtitling protocol over Stream Events. To achieve this task, CCMA has developed the module named ‘MotorSubSE’ which receives the messages from the subtitles-transmitter and sends the transformed subtitles towards the HandData module.
- **HandData:** it consists on a CCMA development for injecting different kinds of data into a DVB headend such as DVB subtitles, EPG, Teletext, HbbTV signalling, etc. It also has a module that allows injecting section structures which is used as a gateway for transmitting the Stream Events

received by UDP from the MotorSubSE.

- **DTT Header:** this module is the set of equipment to generate a valid multiplex for its transmission over a DVB network. The CCMA multiplex is broadcasted as a DVB-T signal that covers the Catalonian region.

CCMA Linear TV multi-language subtitles service

In the particular case of CCMA, when the users access to its TV channels from an HbbTV compatible SmartTV, 'TV3alacarta' HbbTV app will show automatically an overlaid message box (called hook) inviting end users to activate the app by pressing the red button on the TV remote control (Figure 26) In parallel, a second overlaid message box will also present alternative options like:

- Yellow button – to activate the start-over option that allows users to watch the current broadcasted TV program from the start.
- Green button – to activate multi-language subtitles, only if they are available for the current TV program.

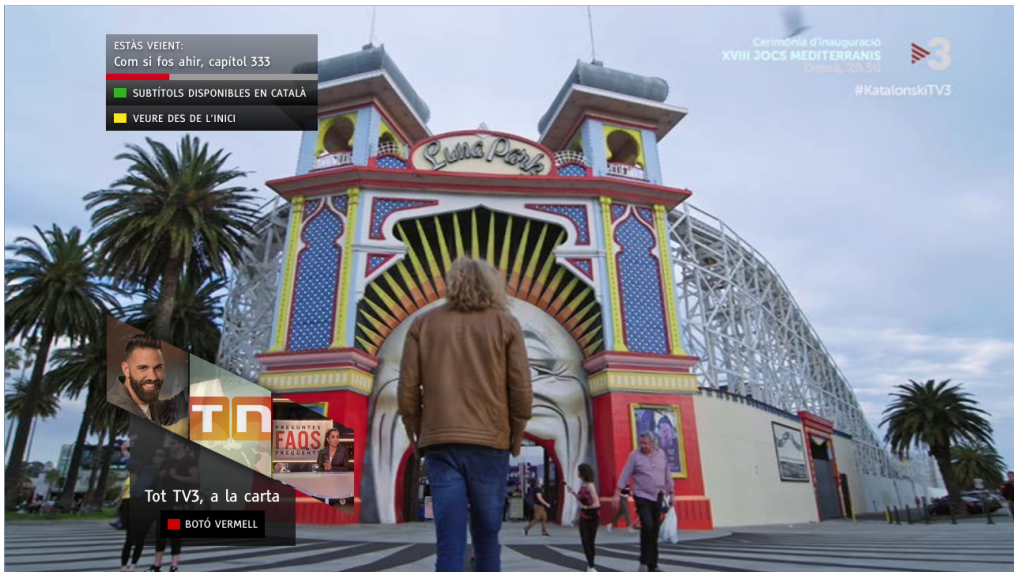

Figure 26. 'TV3alacarta' hook for HbbTV app

Figure 27. Multi-language subtitle configuration

If green button is chosen, the basic live subtitle configuration menu is presented with the list of available language options (Figure 27). It is also possible to access to advanced setup options to customize the subtitles position, background and size.

All subtitle configuration options are stored so anytime a content from the same channel has subtitles that matches with the chosen language, they will be automatically shown as in Figure 28.

Figure 28. Subtitle customization menu

As the multi-language subtitle service is a result of the EasyTV project, end users can also consult information about the project included in the HbbTV portal help section as shown in Figure 29.

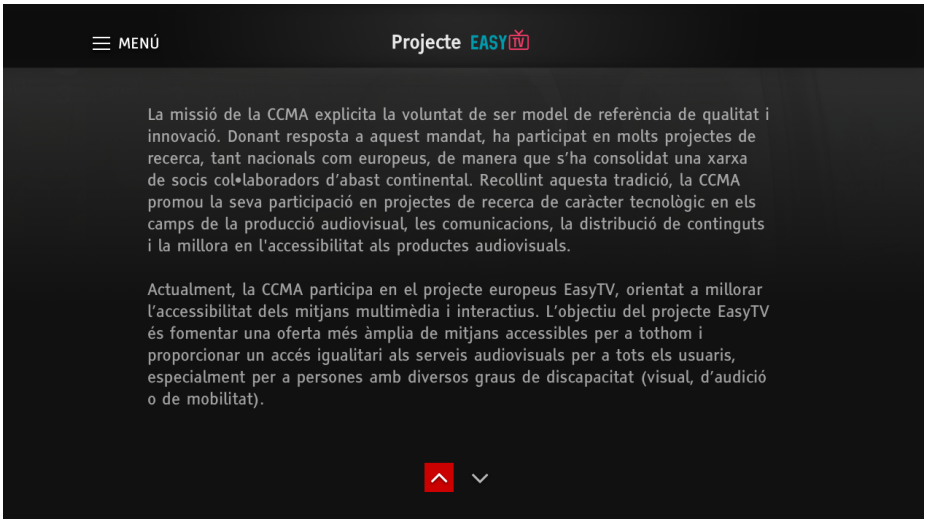

Figure 29. Information about EasyTV project included in the HbbTV app

5.2.1.2 Multi-language Subtitles on VoD "TV3lacarta" HbbTV Service

This section describes the developments carried out on the broadcaster premises to offer subtitles synchronously to VoD contents consumed from the "TV3alacarta" HbbTV service.

As it was explained in section 5.2.1, CCMA deployed the subtitles service for HbbTV in February 2017. This first development allowed the users to enjoy only one subtitle in Catalan for VoD content, and some English kids content with English subtitles only.

The developments also affected the whole accessibility services workflow, which included a repository of subtitles that was ready to distribute the subtitles using several protocols in parallel, making it compatible with broadcast, broadband, VoD and Linear TV workflows (Figure 30).

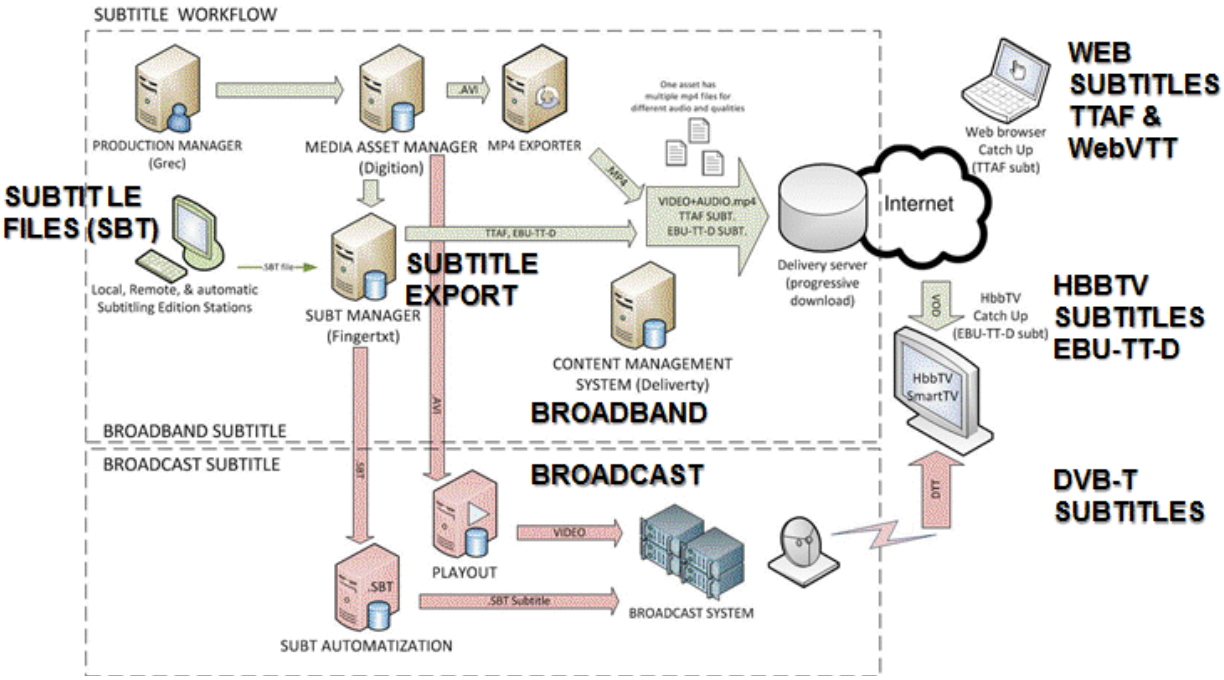

Figure 30. Information about EasyTV project included in the HbbTV app

When a TV program is broadcasted, main Catalan subtitle is transcoded from internal proprietary format (SBT) to DVB subtitle. At the same time, all included multi-language subtitles are transcoded to the proprietary light protocol to be broadcasted through 'stream events' and interpreted by the HbbTV app on compatible SmartTV's (as explained in section 5.2.1.1).

For VoD TV programs, the processes are executed off-line, so the internal subtitle protocol (SBT) is transcoded to WebVTT for "TV3alacarta" web service and mobile app, while the same subtitle is transcoded to EBU-TT-D for "TV3alacarta" HbbTV service.

CCMA HbbTV VoD service

In the particular case of CCMA, when the users access to its channels from an HbbTV compatible SmartTV, 'TV3alacarta' HbbTV app will show automatically an overlaid message box (called hook) inviting end users to activate the app pressing the red button on TV remote control.

If the red button is pressed on the remote control, HbbTV "TV3alacarta" VoD app will start showing main screen (Figure 31) from where is possible to navigate to main menu (upside button) with 8 options (Figure 32):

- Start ("*Inici*") – move to main screen, where the most relevant programs are directly linked.
- Search ("*Cercador*") – allows to search programs by name.
- Last days ("*Últims dies*") – offers access to a scheduler where all TV programs are linked and sorted by day.
- Archive ("*Arxiu AZ*") – direct links to programs sorted by name.
- Last four options offer access to submenus where TV Programs are organized per thematic: TV main programs, series, News and Sports.

Figure 31. CCMA HbbTV start screen with access to main TV programs.

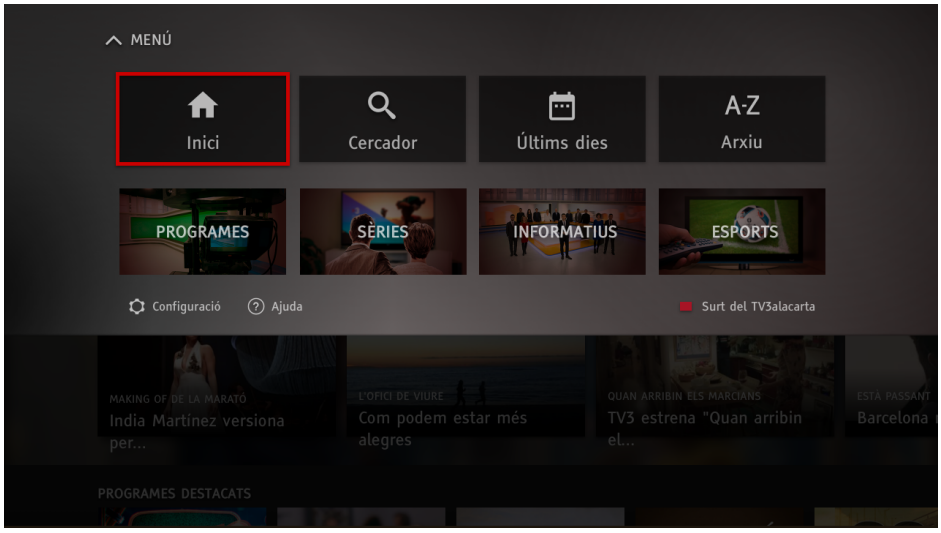

Figure 32. Access to main menu.

In the TV program archive menu, a new section has been created to setup multi-language subtitles. End users can choose to look only for TV programs, for example, with English or Arabic subtitles (Figure 33 and Figure 34).

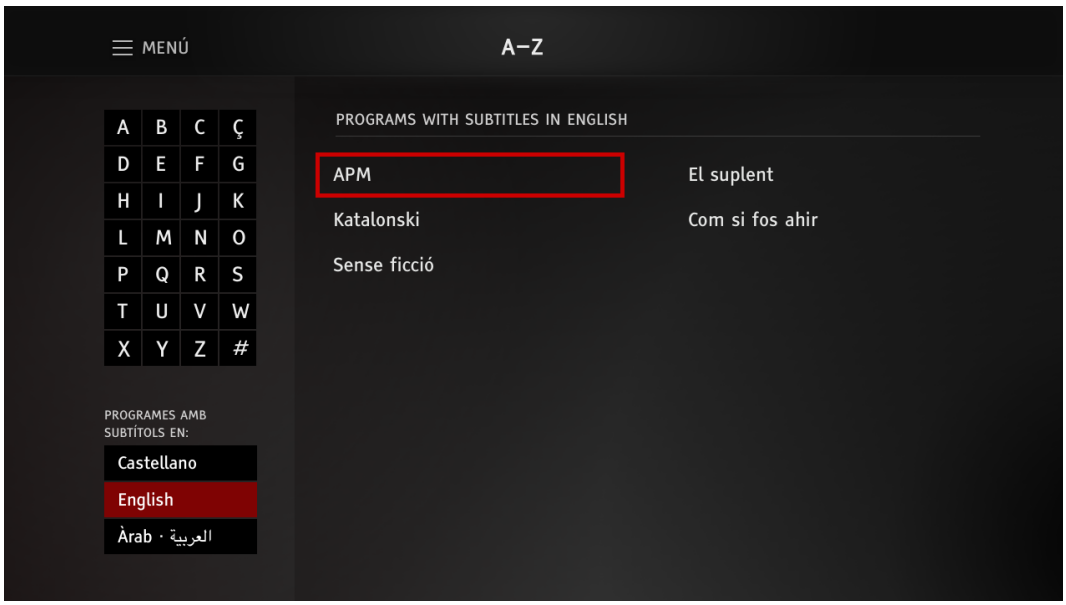

Figure 33. Example of CCMA HbbTV app showing programs with English subtitles

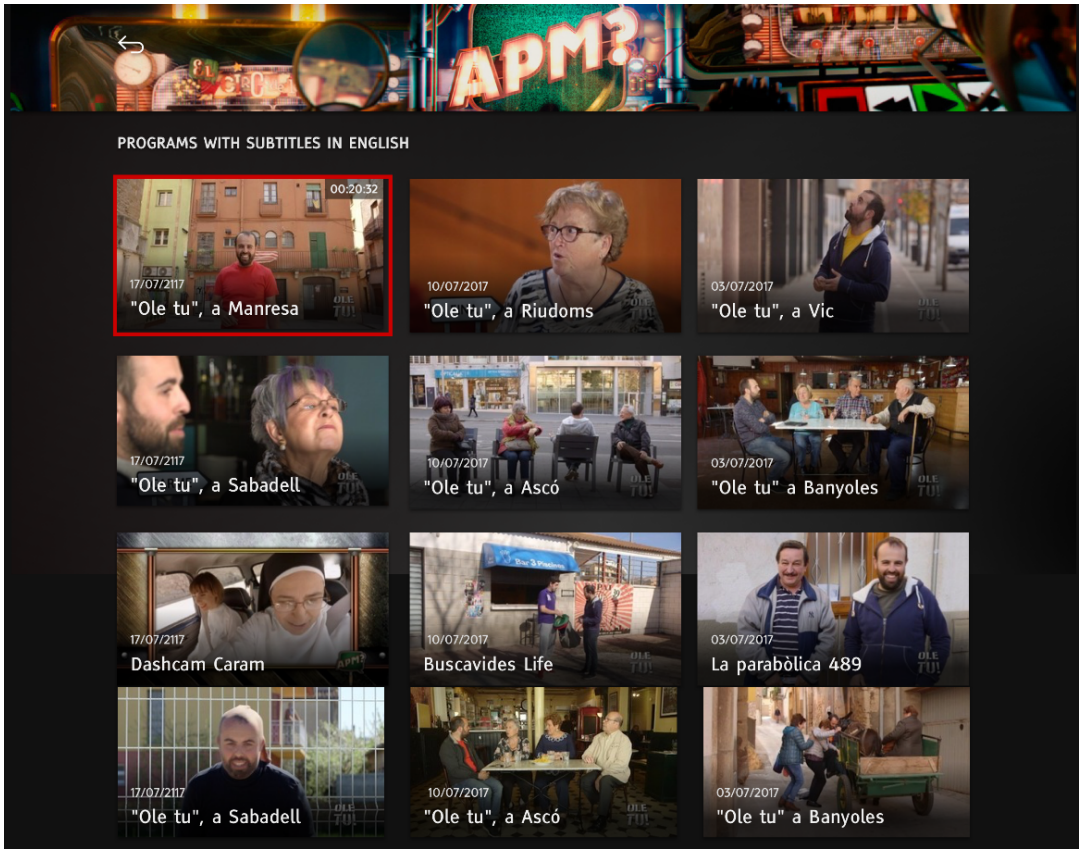

Figure 34. Example of CCMA HbbTV app showing only VoD contents with English subtitles

In the same way as for Linear TV content, end users will be informed when they choose a content enriched with multi-language subtitles, inviting to press the green button to activate and setup (Figure 35).

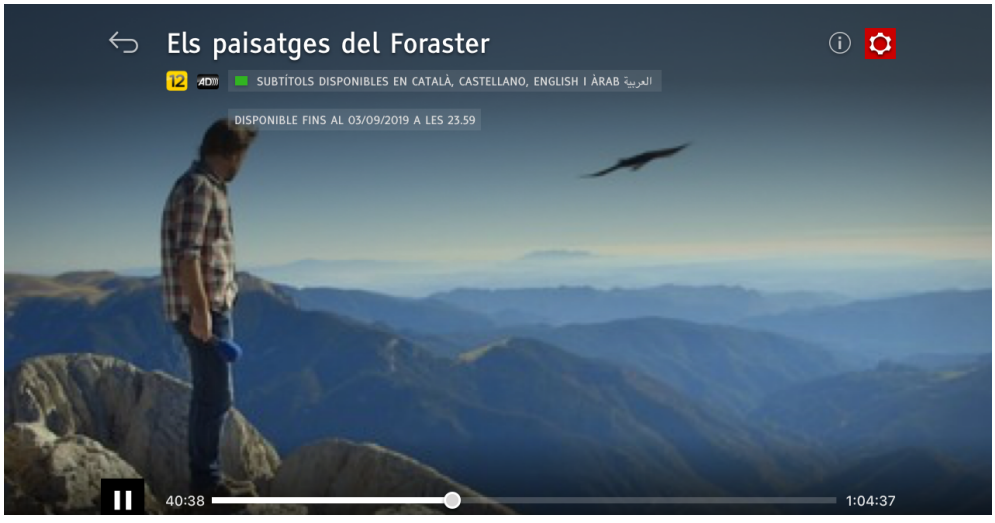

Figure 35. TV Program "Els paisatges del Foraster" offers multi-language subtitles.

Multi-language subtitles can also be customized to suit the user preferences, and the configuration is stored for future use on VoD contents (Figure 36).

Figure 36. Customization for VoD through CCMA HbbTV app

5.2.2. Multi-language on Website OTT

The participation of CCMA in the Hbb4all European project allowed improving the CCMA web services and mobile app's with subtitle service, which were deployed in February 2017.

The adaptation of web services for multi-language subtitles has been carried out, so the web player integrated in CCMA OTT service supports a list of multi-language subtitles which is deployed to end user on the player user interface to allow choosing the desired language when possible.

The adaptation for CCMA mobile apps will need further developments that are foreseen in the roadmap for new future implementations.

CCMA OTT web services

CCMA web services are compatible with all main web navigators (Figure 37): Google Chrome, Microsoft Edge, Firefox and Safari. Main web page allows users to search for concrete contents, look for last days TV programs or directly link to most important content.

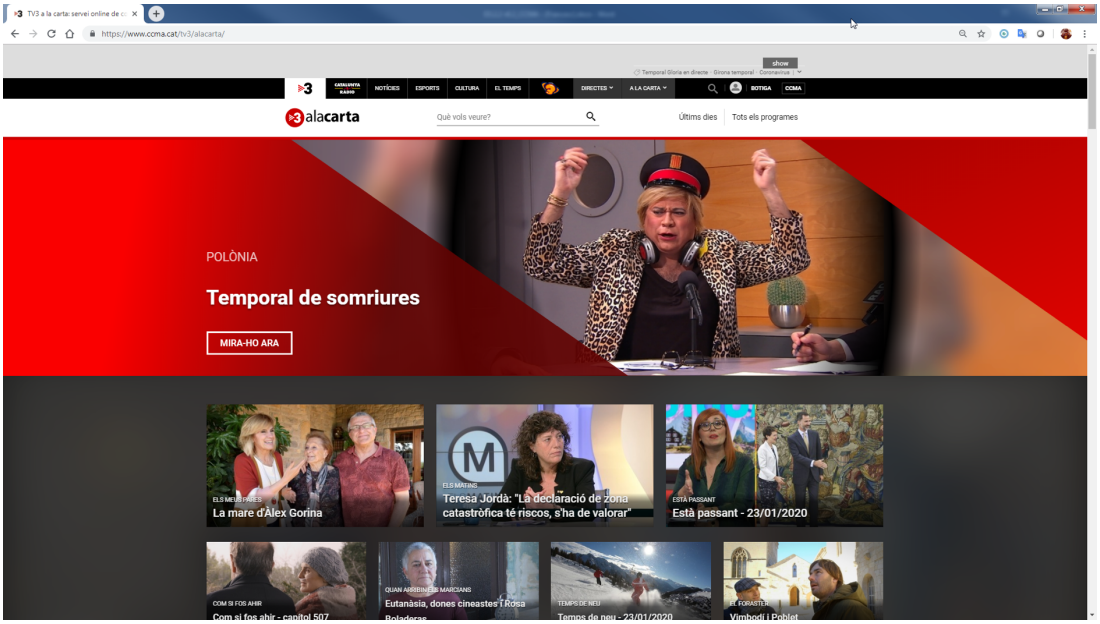

Figure 37. CCMA website

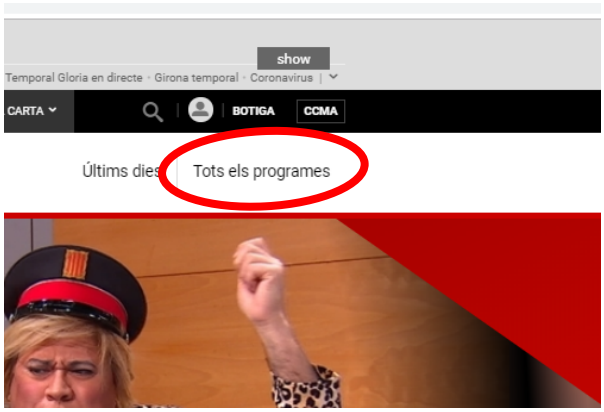

Figure 38. CCMA website access detail

"*Tots els programes*" link (Figure 38) drives users to a special search webpage where TV programs are organized in alphabetical order by its name. In this webpage it is also possible to choose the programs list by topic: series, news, entertainment, sports, culture or musical programs (Figure 39).

Two new sub-sections are presented in the left-bottom side of the webpage, from where it is possible to filter the programs list by the availability of audio-description or multi-language subtitles.

As an example, if a user clicks in the button to see English subtitles, the TV programs list will only show those contents that have English subtitles.

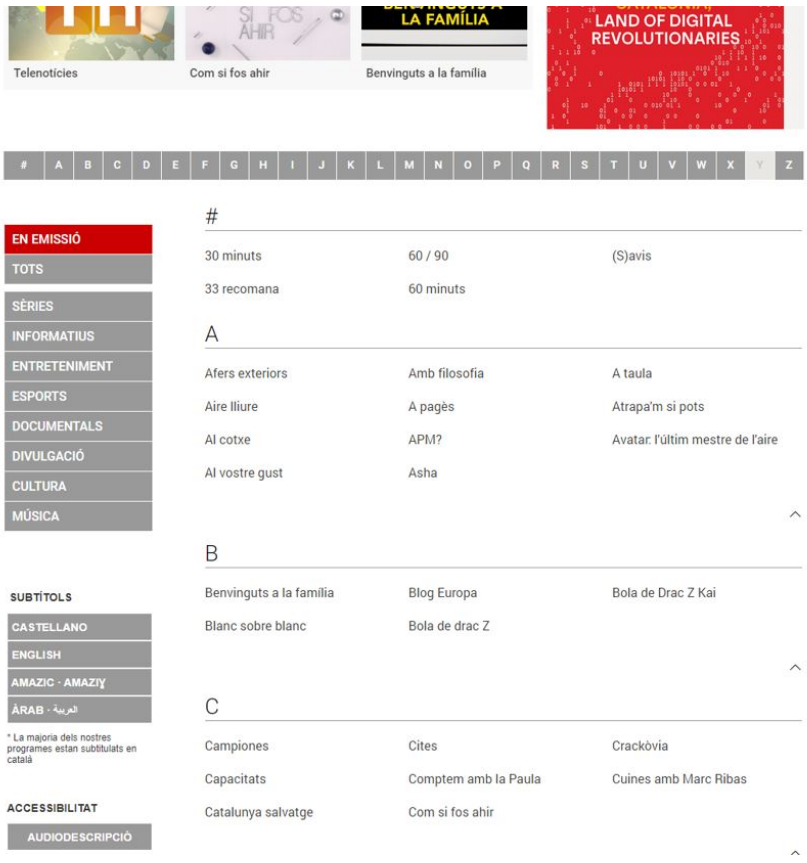

Figure 39. CCMA "Tots els programes" website

In the same way, if the user selects a content with multi-language subtitles from any "TV3alacarta" webpage and starts to watch it, the web player will offer a list with the subtitles available in the control panel (Figure 40).

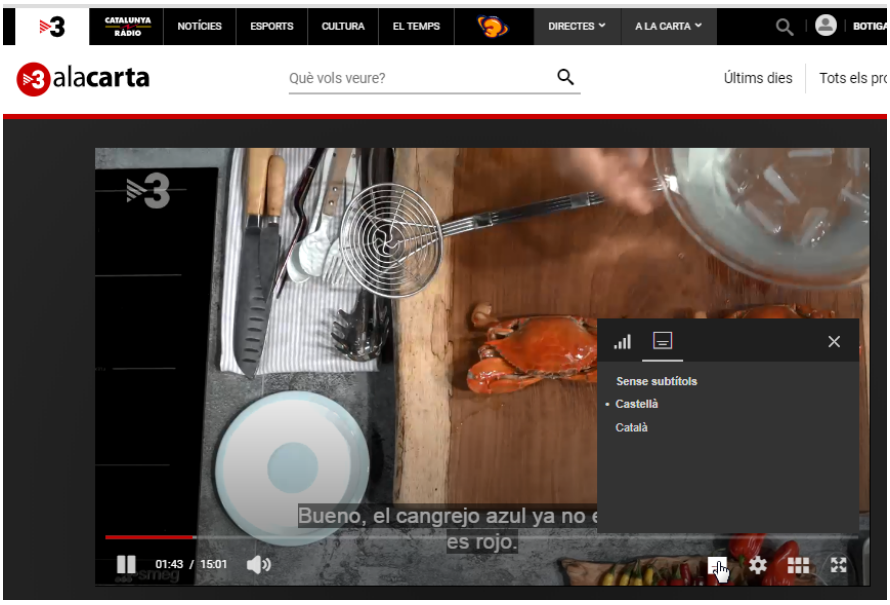

Figure 40. CCMA web player with multi-language subtitles support.

6. CONCLUSIONS

Accessibility in audiovisual industry is not something nice to have, but a needed feature to be considered by design, something that mostly everyone will need sooner or later, due to a sensorial disability, or to learn a new language or, why not, to avoid noise while someone is sleeping. EasyTV explores how accessibility services and assistive technologies can be improved thanks to the latest technological advances that allow to develop new, revolutionary and innovative solutions.

One of the main goals of EasyTV is to design and develop a solution to boost the production of multi-language subtitles and Sign Languages translations. This deliverable has explained the technologies developed to: a) capture Sign Language in multiple languages and b) annotate such videos with semantic information in order to c) store and deduce translations; and d) to provide multilingual-language subtitles.

The EasyTV Ontology-based annotator web service allows an easy population of the Sign Language videos into the developed EasyTV ontology. Moreover, the web service has been fully integrated with the crowdsourcing platform developed in the context of the project. The different functionalities of the platform that requires the access to the ontology are provided through web services, such as creation, modification, retrieval and removal of the Sign Language videos.

The crowdsourcing platform together with the web service have achieved a new scenario in the project in which the users can now work recording Sign Language videos, annotating them and creating new knowledge in the ontology. This processes will require from expert users that know at least one Sign Language which will be in in charge of the enrichment of the knowledge that is reflected through the crowdsourcing platform. Incrementally, new videos and concepts will be stored as well as new translations.

Providing multiple-language subtitles for the same content can be a vehicle of social integration and promote social cohesion. At the same time, multi-language subtitles allow also to offer the local contents internationally, achieving a wider audience and expanding the local culture around the world, promoting, for example, the tourism.

EasyTV pretends also to increase the number and quality of subtitle production for everyone; without a linear increase of costs, in an audiovisual industry where, in general, the budget dedicated to accessibility services is low, (Access Services Pan European Survey 2016 showed that the annual budget for access services was only 0.44% of organizations total annual budget).

Thanks to the audio-subtitles module integrated in the EasyTV platform, it will also be possible to increase the number of spoken languages for the content, helping people with limited literacy abilities to hear the content in their spoken language.

7. BIBLIOGRAPHY

[1] Agerri, R., Bermudez, J. and Rigau, G. (2014): "IXA pipeline: Efficient and Ready to Use Multilingual NLP tools", in: Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014), 26-31 May, 2014, Reykjavik, Iceland.

[2] D3.7: Sign language capturing technology final version – EasyTV deliverable

[3] D5.6: Final report on the set up and implementation of the EasyTV crowdsourcing sign language platform and repository – EasyTV deliverable

[4] ETSI EN 300 743: "Digital Video Broadcasting (DVB); Subtitling systems".

[5] ECMA-404 The JSON Data Interchange Standard.

[6] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60). DOI: http://dx.doi.org/10.3115/v1/P14-5010

[7] Schmid, H. (2013). Probabilistic Part-of-Speech tagging using decision trees. In New methods in language processing (p. 154). https://doi.org/10.1007/BFb0026668

[8] Studer, R., Benjamins, V.R., and Fensel, D.: Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25(1-2) (1998) 161-197