This document is part of the Research and Innovation Action "European Live Translator (ELITR)". This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 825460.

Deliverable D6.1

Publishing Platform

Dario Franceschini (PV), Chiara Canton (PV), Ivan Simonini (PV), Otakar Smrž (CUNI), Ondřej Bojar (CUNI), Adelheid Glott (AV), Franz C. Krüger (AV)

Dissemination Level: Public

Final (Version 1.0), 29th February, 2020

Grant agreement no.	825460
Project acronym	ELITR
Project full title	European Live Translator
Type of action	Research and Innovation Action
Coordinator	Doc. RNDr. Ondřej Bojar, PhD. (CUNI)
Start date, duration	1st January, 2019, 36 months
Dissemination level	Public
Contractual date of delivery	29th February, 2020
Actual date of delivery	29th February, 2020
Deliverable number	D6.1
Deliverable title	Publishing Platform
Type	Demonstrator
Status and version	Final (Version 1.0)
Number of pages	18
Contributing partners	AV, PV, CUNI
WP leader	PV
Author(s)	Dario Franceschini (PV), Chiara Canton (PV), Ivan Simonini (PV), Otakar Smrž (CUNI), Ondřej Bojar (CUNI), Adelheid Glott (AV), Franz C. Krüger (AV)
EC project officer	Alexandru Ceausu
The partners in ELITR are:	▪ Univerzita Karlova (CUNI), Czech Republic ▪ University of Edinburgh (UEDIN), United Kingdom ▪ Karlsruher Institut für Technologie (KIT), Germany ▪ PerVoice SPA (PV), Italy ▪ alfatraining Bildungszentrum GmbH (AV), Germany
Partially-participating party	▪ Nejvyšší kontrolní úřad (SAO), Czech Republic

For copies of reports, updates on project activities and other ELITR-related information, contact:

Doc. RNDr. Ondřej Bojar, PhD., ÚFAL MFF UK
Malostranské náměstí 25
118 00 Praha, Czech Republic
bojar@ufal.mff.cuni.cz
Phone: +420 951 554 276
Fax: +420 257 223 293

Copies of reports and other material can also be accessed via the project's homepage:

`http://www.elitr.eu/`

Contents

1 Executive Summary

This deliverable reports on the presentation platform for live automatic transcription and translation developed and extended during the first year of the ELITR (European Live Translator) project. As required by the T6.1 task, the alfaview® platform has been extended and integrated with the PerVoice Service Architecture to deliver live transcription and translation to remote meeting participants.

The feature has already been tested successfully in the field by alfatraining, an eductional provider who uses alfaview®. A participant with hearing impairment used the live transcript in order to follow the lessons and participate in discussions with the lecturer and other participants.

PerVoice has developed a web-based subtitle streaming platform, the Presentation Platform, which has been used during the Annual meeting of the EUROSAI Core Group 2, in June 2019 in Prague. During the event it delivered live and automatic subtitles to roughly 30 participants in 10 different languages.

CUNI have implemented another approach to presenting the transcribed and translated text, the Online Text Flow web application and command line tools. It is focused on providing the user with the complete history of the running text and improved translation into multiple languages at the sentence level. The tool set has been deployed during the Linguistic Mondays seminar series at CUNI (15–30 attendees weekly), as well as at the preparatory run of the EUROSAI 2020 Congress workshop on Using Language IT Technologies in Audits held at the SAO Czech Republic in February 2020 (21 participants).

In Section 2, the two main presentation views are described. Then in Sections 3 to 5, the three different implementation are presented with their technical details.

2 Presentation Techniques Overview

The last step in a spoken language translation (SLT) system is the delivery of the translated text to the user. The ELITR project focuses on the delivery of translation in the textual form. We experiment with two different types of view:

- The "subtitle view", optimized toward minimal use of screen space. Only two lines of text are available which leaves room either for e.g. a streamed video of the session or the slides, or for many languages displayed at once, if the use case requires a multi-lingual audience. Details in Section 3.
- The "paragraph view", optimized for context understanding. A scrollable long history of text is provided in a custom selection of languages, sacrificing space for the video. Nonetheless, a video pane can still be included, should that optimize the user experience. Details in Section 5.

Both views receive input in the same style, as text, but the different implementations result in different types of views.

3 Presentation Platform

In this section, we introduce the Presentation Platform which focuses on the face-to-face conferences use case (interpreting official speeches and workshop-style presentations). Usually on a main stage the speakers present their works together with slides and sometimes a live interpreting service is provided to participants. The goal of the Presentation Platform is to provide a web-based solution for translated live subtitles usage on handheld devices like personal laptops.

3.1 Main functions

The Presentation Platform provides the following main functionalities:

- Live translated subtitles to conference audience so that understanding, personal interaction and cooperation is increased.
- Live streaming of slides.
- The user is allowed to select the language of live subtitles displayed, choosing it from the available languages.
- The conference organization is allowed to define the available languages based on online automatic transcription and translation and on local interpreting services.
- The conference organization is allowed to define the most reliable source of translated subtitles for a specific language when more than one source is available.

Additional non-functional requirements are:

- The platform shall support HTTP Live Streaming protocol (HLS) for video streaming.
- The platform shall be a web-based solution.

3.2 Architecture

In the following, we introduce some basic information regarding the PerVoice Service Architecture and then we go through the Presentation Platform overview.

3.2.1 PerVoice Service Architecture Introduction

The ELITR architecture focuses on a distributed connection-based client-server application. The PerVoice Service Architecture provides a central coordination point called Mediator. Universities and research labs integrate their components through the implementation of software modules called Workers, each declaring the service it can provide (e.g. ASR in a specific language or SLT from a source language to a target one). Service descriptions and service requests are fully specified by their input and output fingerprints and types. Fingerprints are a code made of a two-letter language code (ISO369-1) followed by an optional two-letter country code (ISO3166) and an optional additional string specifying other properties such as domain, type, version, or dialect (`ll[-LL[-dddd]]`). Types are: audio, text, unseg-text (unsegmented textual data such as ASR hypotheses). For example, the ASR service which takes in input European English audio and retrieves European English unsegmented text adapted on news domain will be defined by "`en-GB-news:audio`" input fingerprint and "`en-GB-news:unseg-text`" output fingerprint.

3.2.2 Presentation Platform Architecture and Data Flow

The Presentation Platform is a Ruby on Rails web application hosted on an Nginx web server in the PerVoice Cloud infrastructure. The application provides an API to receive text streams from the PerVoice Service Architecture for all the possible languages. The clients establish a web socket connection with the Presentation Platform server to receive text streams updates. Web sockets offer a couple of crucial advantages for subtitle live streaming use case:

- low-latency communication to get subtitles updates as fast as possible,
- only one connection required per client.

The Presentation Platform integrates also a web player with HLS support for video or slides streaming.

During face-to-face conferences, audio is collected both directly from the speakers and from the live-interpreting service (where available). The production room hosts one or more Service Architecture Client instances which require specific services like:

- perform automatic transcription of this audio stream using a specific ASR language and publish the result in the Presentation Platform,
- perform automatic transcription of this audio stream using a specific ASR language and translate the obtained text into a specific MT target language and publish the result in the Presentation Platform.

The whole flow is managed through the fingerprint declaration in the client request.

In order to provide publication service, a Publication Worker is connected to the PerVoice Service Architecture. The Publication Worker sends data through the provided API to the Presentation Platform. End users' clients' browsers establish a web socket with the Presentation Platform server and obtain the text updates.

An example of workflow is represented in Figure 1. Let's suppose that:

- English audio is collected from the interpreting service,
- German audio is collected from the conference stage,
- ASR workers for both English and German are available,
- MT workers from English to Czech and from German to Czech are available,
- Publisher workers for all the required services are available,
- subtitles for English, German and Czech are required.

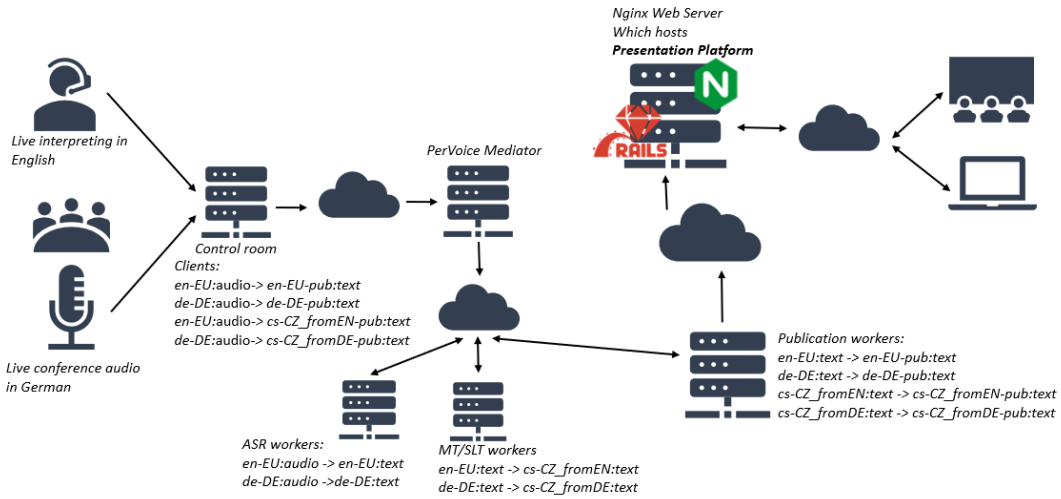

Figure 1: Presentation Platform data flow.

In the control room, two clients will be instantiated, one per audio input. The first one will perform the following requests:

- publish English subtitles starting from English audio (`en-EU:audio`→`en-EN-pub:text`),
- publish Czech subtitles starting from English audio (`en-EU:audio`→`cs-CZ_fromEN-pub:text`)

The second one will perform the following requests:

- publish German subtitles starting from German audio (`de-DE:audio`→`de-DE-pub:text`)
- publish Czech subtitles starting from German audio (`de-DE:audio`→`cs-CZ_fromDE-pub:text`)

The Mediator, verifies the feasibility of the received requests based on its knowledge of available services, and then it instantiates the following paths:

- `en-EU:audio`→`en-EU:text`→`en-EU-pub:text`,
- `en-EU:audio`→`en-EU:text`→`cs-CZ_fromEN:text`→`cs-CZ_fromEN-pub:text`,
- `de-DE:audio`→`de-DE:text`→`de-DE-pub:text`,
- `de-DE:audio`→`de-DE:text`→`cs-CZ_fromDE:text`→`cs-Cz_fromDE-pub:text`.

The four Publication workers provide the subtitles to the Presentation Platform and the end user browser receives them via web socket. The user's view filters displayed subtitles based on user's preferences. What happens when more than one stream is available for a specific language (like the Czech in the provided example) will be explained in Section 3.3.4.

3.3 User Interface

The Presentation Platform provides 4 web pages:

- Login page. Due to SAO privacy reasons, only SAO workshop and conferences participants are allowed to access the video and subtitles streams. See Figure 2.
- Main page with video and subtitles streaming. See Figure 3.
- Settings page. See Figure 4.

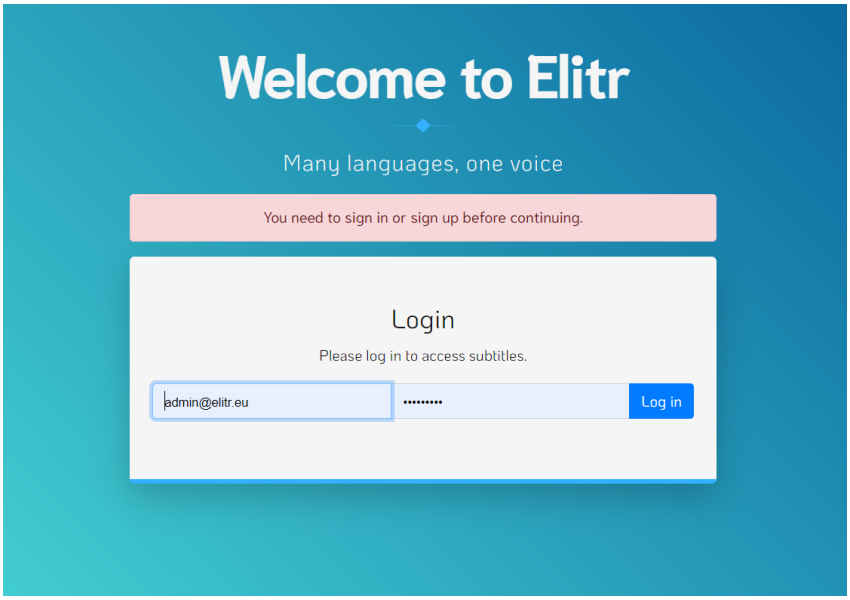

Figure 2: Presentation Platform user types.

- Language stream setting page. See Figure 5.

Please note that the actual appearance of the web application may change during the project. We are constantly improving the usability of the solution, taking into account feedback from partners and users.

3.3.1 Login Page and User Types

During events, a username and password is shared with the audience.

Presentation Platform has two user types:

- guest user,
- administrator.

A guest user is allowed to access the main page and see video and subtitle streams. They are able also to display one or more subtitle streams based on their linguistic knowledge and preferences. The guest user is not allowed to access to the settings and to language stream setting page.

The administrator has the guest user permission and, in addition, they are allowed to access also to the settings and to language stream setting page.

3.3.2 Main Page

The main page contains:

- video player for HLS slides or video streaming,
- dynamic black boxes which host the subtitle streams,
- a configurable number of buttons to enable/disable specific subtitles,
- logout button,
- settings button.

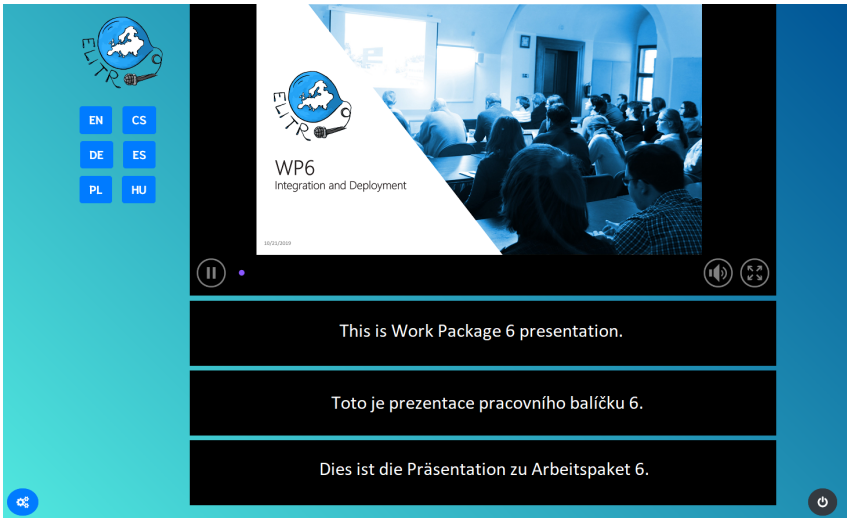

Figure 3: Presentation Platform main page.

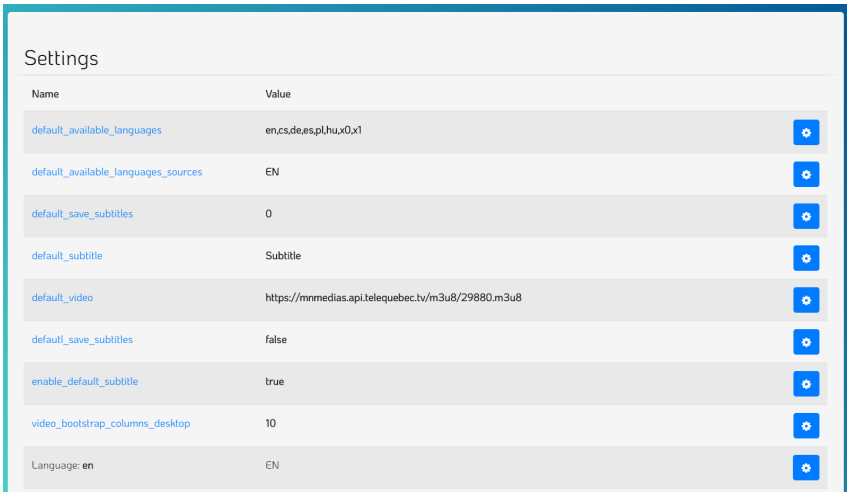

Figure 4: Presentation Platform settings page. Only EN is available as a source language. The picture is cropped to display just the en language source configuration.

When a user first loads the main page, the video/slides streaming starts. The user is allowed to enable subtitle streaming in a specific available language by clicking the corresponding language button. The user can enable one or more languages based on their preferences. To disable one subtitle stream, it's sufficient to click again the corresponding language button. When an administrator logs in, the settings button also becomes available.

Based on the example in Figure 1, the available languages will be English, German and Czech.

3.3.3 Settings Page

Only the administrator is allowed to access the settings page. The main functionalities provided are the following:

- default_available_languages: list of subtitle stream languages available for end users. The administrator determines available subtitles, based on his knowledge on available services, editing this list. Example in Figure 1 should have en, de and cz as default available languages.

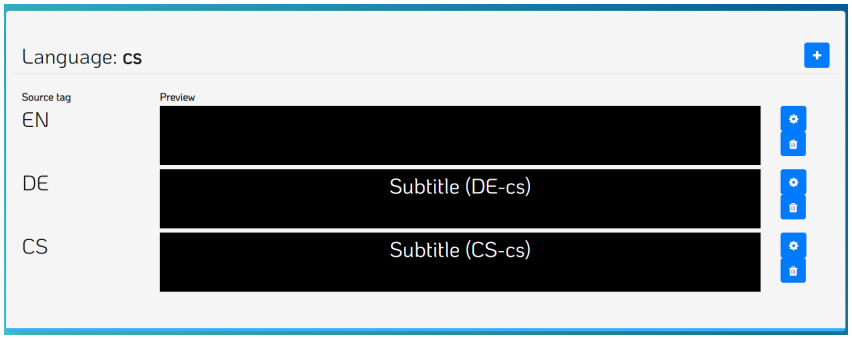

Figure 5: Presentation Platform language stream setting page for Czech language. Available sources are English, German and Czeach. Selected source is English.

- default_available_languages_sources: list of subtitles stream sources available. The example in Figure 1 would have EN and DE as default available languages sources.
- default_video: URL for HLS video streaming. The URL is provided by the conference slide streaming service.

For each available language, a language stream settings page becomes available.

3.3.4 Language Stream Setting Page

Only the administrator is allowed to access the language stream setting page from the settings page. Each configured language has its own setting page which allows the administrator to configure the source of the subtitle to be displayed. It is also possible to watch a preview of sources output so that the administrator is allowed to define the most reliable source of translated subtitles for a specific language when more than one source is available.

Based on the example in Figure 1, English subtitles will have English source language since the subtitles will be the output of the interpreter's audio automatic transcription. German subtitles will have German source language since the subtitle will be the output of the main stage microphone's audio automatic transcription. The Czech subtitle instead will have two possible sources: the translations of the English or the German automatic transcription. In the language stream setting page, the administrator configures both of the options (English source and German source) and, looking at the quality of the provided translation, defines which one has to be displayed as Czech subtitle.

4 alfaview

alfaview® is an online video conferencing solution with focus on the educational use case. Its goal is to provide effortless audiovisual communication over an extended period of time without deteriorating in quality. Tailor-made for their own demand and developed from the ground up by alfatraining, one of the market leaders in government funded education in Germany, the platform enables users to manage virtual conference rooms and conduct online meetings via cross platform client software.

alfaview® provides additional features to augment conference and classroom situations e.g. additional cameras, screen share or text based chat.

As a result of the ELITR project, alfaview® now offers a live transcript and translation feature, that integrates seamlessly into the alfaview® user interface. When activated, the transcript of the spoken word is displayed in a dedicated sidebar element. Meeting room participants can easily follow an ongoing presentation while simultaneously reading over the transcribed text as it flows top to bottom in a chat-like manner. Additionally participants can choose to display a translated version of the transcript.

4.1 Main functions

The alfaview® platform provides the following main functionalities:

- Intuitive, easy-to-install client software for all major platforms,
- Allows online meetings with 100+ participants,
- Low latency, high quality audio and video streaming technology,
- Simple user role concept (Moderator and Participant),
- Latest security standards for transport encryption,
- Includes screen sharing, secondary camera and group chat,
- Manage conference rooms, users and guests via a web based administration interface,
- Resilient and highly available microservice architecture,
- API allows integration into third party software, e.g. learning management systems.

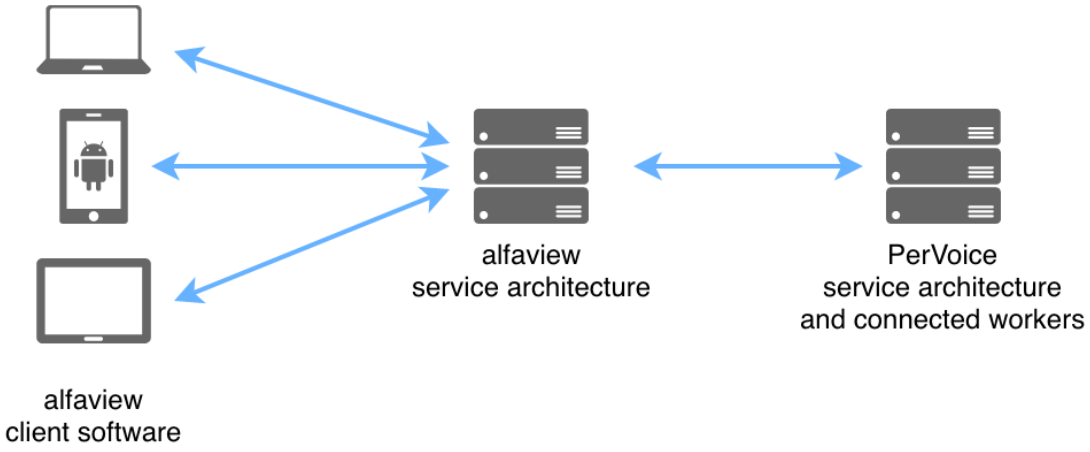

Figure 6: Live transcript and translation data flow.

4.2 Architecture and Data Flow

- The alfaview® client sends an audio stream to the alfaview® service architecture.
- Dedicated microservices re-stream the audio to all connected participants and also forward it to configured third party services such as the PerVoice service architecture for further processing.
- In the PerVoice service architecture, the audio stream is processed and transformed into a textual transcript and translation.
- As a last step, the transcript and the translation is dispatched to all connected alfaview® clients for display.

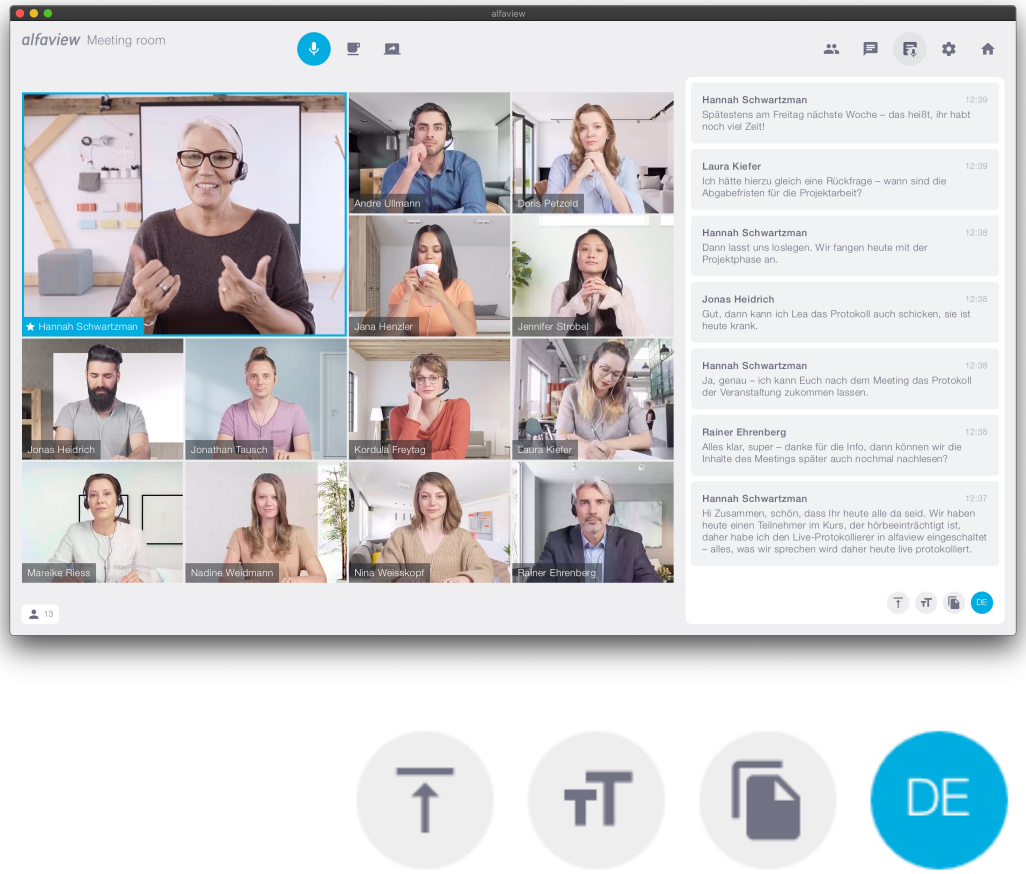

Figure 7: User interface of the alfaview® client in a typical meeting situation. The sidebar displays the live transcript.

4.3 User Interface

The alfaview® client primarily displays a grid of video streams in a meeting room. In addition, the sidebar serves as a placeholder for secondary content such as the live transcript or translation.

The sidebar can be adjusted in width, and offers the following controls:

- Flip text flow direction,
- Toggle text size,
- Copy to clipboard,
- Select the display mode (transcript or translation).

5 Online Text Flow

Online Text Flow is a web application and a set of command line tools focused on providing the user with the complete history of the transcribed and translated text. It reorganizes the data into sentences, discerns stable (confirmed) and unstable (preliminary) text output, and keeps the history of the text flow. It improves the user experience by translating at the sentence level and allowing the user to follow at their own pace and understand the multilingual context.

5.1 User Interface

The Online Text Flow web application has been developed primarily to improve the users' chance to recover from translation errors. Thanks to the distinction of the stable/unstable information in the incoming messages, the unstable text is being replaced by the the stable one as it is received. This lets the users decide if they want to read the immediate output, or rather wait for the final message.

The web application view is illustrated in Figure 8. Sentences in black are stable or "complete", no update is possible. Sentences in dark gray with yellow index number are tentative or "expected", their segmentation and content, and thus translation, still may change. The last "incoming" sentence fragment in light gray with red index number is still being uttered and is thus highly unstable. Each language, or rather text flow of an intuitive name, is shown in its own column. The ordering, naming, and selection of columns is up to the user and can be changed by opening a hyperlink to a dedicated endpoint of the app, or clicking a button in the sidebar.

The layout of the interface allows for multiple languages or translation versions to be displayed at once. The screen scrolls automatically to the bottom unless the user scrolls up to read more of the history, then back to the end to resume the automatic scrolling. The alignment of sentences is not fully parallel by design, as we prefer contiguous columns within each language over tabular layout, which may even be inefficient to render for long histories. One important aspect is however coupled across columns, and that is the floating alignment level for finalized sentences. The complete text is aligned at the floating level in the lower part of the page, while

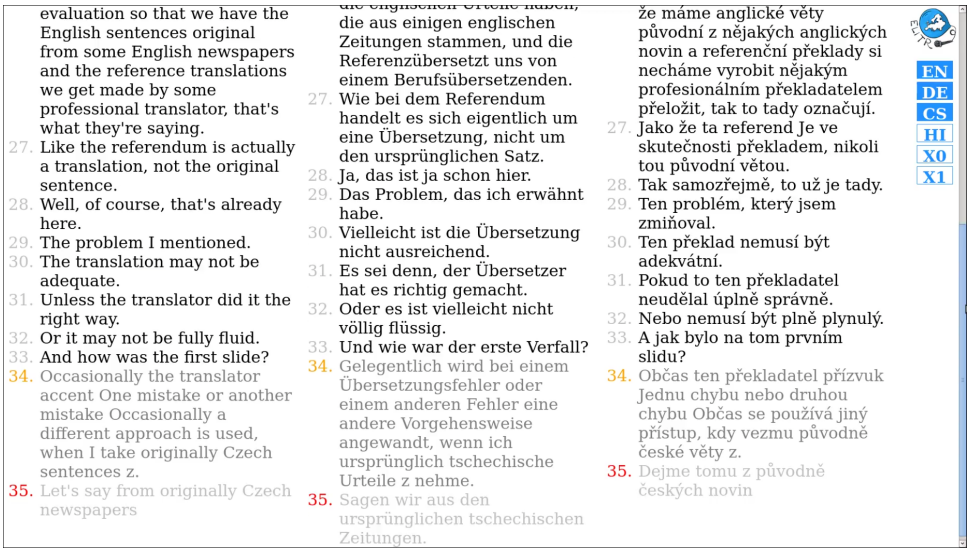

Figure 8: Screenshot from the Online Text Flow paragraph view of simultaneous translation output. The talk was given in Czech (right), automatically recognized and translated into English (left) and then into German. Sentence indices correspond to each other across languages. There are multiple types of translation errors visible in the output, starting from excessive and missing capitalization, wrong sentence segmentation up to mistranslations.

the unstable hypotheses flicker below the level downwards, varying in their vertical length.

Users appreciate the persisting availability of the text and the context. A possible drawback of this interface is that all errors in the output of the system remain on the screen for a long time, and may needlessly distract the user. Imperfections in the transcription or translation as well as the overflow of visual information determine the user satisfaction and the viability of the solution the most.

We are exploring how to integrate and simplify the interfaces. We may include a side pane for the slides or a video, show the flickering expected and incoming text as subtitles, and display the complete text with its scrolling features in a single column or two if possible. Further implementation and usability tests are needed.

5.2 Architecture

Online Text Flow connects to the PerVoice Mediator which provides the data after speech recognition. Both systems then exchange the machine translation texts. There are four main components in Online Text Flow, connected linearly to form a processing chain:

Events Turn data from speech recognition into text for machine translation.

Client Emit data from the Events to the Server with a custom name of the event stream.

Server Run the web application to merge, stream, and render online text flow events.

Browser Display the text flow graphically in a web application with its interactive features.

Let us give an example of the raw output of speech recognition and how it is processed by the Events component (run by the command `online-text-flow events`) into a stream of complete, expected, and incoming text events, as discussed in Section 5.1. The `$` below denotes the command line input. We look into the first 30 lines of updates from the speech recognition, and how they are transformed by Events:

```
$ head -n 30 data/en.txt
130 480 You...
130 840 You should...
130 1200 You should...
130 2280 You should...
130 10200 You should...
130 10560 You should...
130 13080 You should...
130 14160 You should thank.
130 16680 You should. Thank there have...
130 17040 You should. Thank there have been...
130 17400 You should. Thank there have been many...
130 17760 You should. Thank there have been many revel...
130 18120 You should. Thank there have been many revolution.
130 18480 You should. Thank there have been many revolutions...
130 18840 You should. Thank there have been many revolutions over the...
130 19560 You should. Thank there have been many revolutions over the last century.
130 20280 You should. Thank there have been many revolutions over the last century. But perhaps...
130 20640 You should. Thank there have been many revolutions over the last century. But perhaps none...
130 21000 You should. Thank there have been many revolutions over the last century. But perhaps none as...
130 21180 You should. Thank there have been many revolutions over the last century, but perhaps none as sick...
130 21720 You should. Thank there have been many revolutions over the last century, but perhaps none as significant...
130 22080 You should. Thank there have been many revolutions over the last century, but perhaps none as significant as...
130 22440 You should. Thank there have been many revolutions over the last century. But perhaps none as significant as the law...
130 22700 You should. Thank there have been many revolutions over the last century. But perhaps none as significant as the large...
130 3655 You should. Thank
3655 23150 there have been many revolutions over the last century, but perhaps none as significant as the longevity red...
3655 4759 there
4759 23520 have been many revolutions over the last century, but perhaps none as significant as the longevity revolution.
4759 23750 have been many revolutions over the last century, but perhaps none as significant as the longevity revolution.
4759 24600 have been many revolutions over the last century, but perhaps none as significant as the longevity revolution. We...

$ head -n 30 data/en.txt | online-text-flow events | (head -5; echo ---; tail -7)
100 101 You...
100 101 You should...
100 110 You should thank.
100 110 You should.
200 201 Thank there have...
---
100 200 You should.
200 210 Thank there have been many revolutions over the last century.
300 301 But perhaps none as significant as the large...
200 201 Thank there have been many revolutions over the last century, but perhaps none as significant as the longevity red...
200 210 Thank there have been many revolutions over the last century, but perhaps none as significant as the longevity revolution.
200 210 Thank there have been many revolutions over the last century, but perhaps none as significant as the longevity revolution.
300 301 We...

$ head -n 30 data/en.txt | online-text-flow events --text
You should.
```

```
Thank there have been many revolutions over the last century, but perhaps none as significant as the longevity revolution.

We...
```

The Events component copes with the evolving timestamps and the associated fragmentary text of speech recognition, regroups the text and classifies it with respect to its stability using artificial timestamps, which number the sentences and encode their status. Specifically:

- Sentence index is multiplied by 100 and provided in the first ("start") timestamp.
- The difference between the "start" and "end" timestamp encodes the sentence status:
  - The difference of 1 corresponds to the "incoming", highly unstable sentence.
  - The difference of 10 corresponds to the "expected" sentence, a complete sentence which can nevertheless change.
  - The difference of 100 corresponds to the "complete" sentence. The final version which will not change and will not be sent again.
- Whenever a sentence of index i is sent (i.e. when a line starting with $100i$ arrives), all previous versions of sentences of higher indices $j > i$ are invalidated. Usually, their updated versions arrive shortly.

At the final moment of the example, the user can see three sentences, one of each stability class.

The Client component gets its data in the Events format via standard input on the command line, and connects to the Server either through HTTP POST requests, or via a websocket connection, which is preferred for speed and efficiency. Running the Client and the Server is straightforward, exemplified at the source repository.[1]

These three backend components are implemented as a package in Python and are integrated with the Click and Setuptools frameworks for designing the command line interfaces and convenient deployment. The Server is built in the Quart framework for asynchronous web applications, as it provides reliable concurrency, websockets, and server-side event streaming functionalities. We moved to Quart successfully after trying hard to achieve these functional goals with Flask combined with Gevent and other libraries. The Server is now deployed using Hypercorn and proxied via Nginx and Apache, as needed by the university infrastructure.

The Online Text Flow frontend component runs in the Browser. After simple authentication, the Server sends it a HTML5 document that gets constantly updated and re-rendered with the coming text flow events and the running JavaScript code. We use CSS3 to achieve the efficient layout, especially the flexible columns and their floating alignment. With further plans for the user interface layout and interactivity described in Section 5.1, we may consider using any appropriate JavaScript frameworks. However, currently, we do well without them and keep the code very lightweight in order to minimize any rendering delays.

5.3 Documentation

Here we very briefly document the components of Online Text Flow. The documentation is included in the application itself, so we can just reproduce these documentation messages.

5.3.1 Events.py

```
Usage: online-text-flow events [OPTIONS]

  Turn data from speech recognition into text for machine translation. The
  emitted events are classified sentences rather than text chunks evolving
```

[1] `https://github.com/ELITR/online-text-flow` for authorized users.

```
  in time and disturbing the flow. The complete text is emitted just once.

Options:
  -l, --line  Output the events as lines of artificial timestamps and text,
              where specific differences in timestamps group the events and
              classify the text as "complete", "expected", and "incoming".
              [default: --line]
  -j, --json  Output the events as JSON objects with detailed information
              about the data, the flow, the text, and other indicators.
  -t, --text  Output the resulting text split into classes by empty lines.
```

5.3.2 Client.py

```
Usage: online-text-flow client [OPTIONS] [KIND] [URL]

  Emit data from the standard input as the KIND of events to the URL/send
  websocket or the URL/post endpoint, depending on the scheme of the URL.
  KIND is empty and URL is http://127.0.0.1:5000 by default. Try ws://...!

  If an input line contains two integers as artificial timestamps and then
  some text, an event is being built from the consecutive lines while the
  timestamps increase. The specific difference of timestamps on one line
  classifies the text as "complete", "expected", "incoming", or ignored.

  If the data on a line is a JSON object, the event being built is posted,
  then the data object is decorated and posted as an event of its own.

  Lines that do not fit the logic are ignored. They do not emit the event in
  progress and are printed to the standard error. Use the --verbose option
  to observe the implementation details and the semantics of the events.

Options:
  -v, --verbose  Print the JSON event and the response code from the server.
                 [default: False]
```

5.3.3 Server.py

```
Usage: online-text-flow server [OPTIONS] [KIND]...

  Run the web app to merge, stream, and render online text flow events. Post
  events at /post. Send events thru a websocket at /send instead of posting
  separate requests. Listen to the event stream at /data. Browse at /.

  The KIND of events to browse by default is ['en', 'de', 'cs']. Change this
  on the command line for all browsers. Set the /menu endpoint for a custom
  menu in the browser, like /menu/en/de/cs, and empty to reset.

  http://github.com/ELITR/online-text-flow

Options:
  --host TEXT           [default: 127.0.0.1]
  --port INTEGER        [default: 5000]
  --user TEXT           [default: ********]
  --pass TEXT           [default: ********]
```

6 Conclusion

This deliverable described 3 different ways of presenting ASR and SLT output: the PerVoice Presentation Platform represents it as subtitles, alfaview presents it with longer context in a side pane similar to a chat and Online Text flow as paragraphs of growing text.

This diversity is intentional and allows us to experiment with and assess the usefulness of the different layouts in different application settings and preference of the event organizers.