D5.1: Specification of CASMACAT workbench

Daniel Ortiz, Vicent Alabau, Philipp Koehn and Ragnar Bonk

Distribution: Public

CASMACAT
Cognitive Analysis and Statistical Methods
for Advanced Computer Aided Translation

ICT Project 287576 Deliverable D5.1

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development.

Project ref no.	ICT-287576
Project acronym	CASMACAT
Project full title	Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation
Instrument	STREP
Thematic Priority	ICT-2011.4.2 Language Technologies
Start date / duration	01 November 2011 / 36 Months

Distribution	Public
Contractual date of delivery	January 31, 2012
Actual date of delivery	April 3, 2012
Date of last update	April 3, 2012
Deliverable number	D5.1
Deliverable title	Specification of CASMACAT workbench
Type	Report
Status & version	Draft
Number of pages	34
Contributing WP(s)	WP5
WP / Task responsible	UEDIN, CBS, UPV
Other contributors	
Internal reviewer	Philipp Koehn
Author(s)	Daniel Ortiz, Vicent Alabau, Philipp Koehn and Ragnar Bonk
EC project officer	Aleksandra Wesolowska
Keywords	

The partners in CASMACAT are:

University of Edinburgh (UEDIN)
Copenhagen Business School (CBS)
Universitat Politècnica de València (UPVLC)
Celer Soluciones (CS)

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

The CASMACAT Project Co-ordinator
Philipp Koehn, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
pkoehn@inf.ed.ac.uk
Phone +44 (131) 650-8287 - Fax +44 (131) 650-6626

Copies of reports and other material can also be accessed via the project's homepage: http://www.casmacat.eu/

Executive Summary

This document contains details about the design of the CASMACAT workbench. It outlines the major components, their interaction, and gives also implementation guidelines. The deliverable is a snapshot of the document at the beginning of the CASMACAT project, it will be refined throughout development and serves as technical documentation.

Contents

1 Introduction **5**

2 Use Cases **6**
2.1 "Authenticate" Use Case 6
2.2 "Upload Document" Use Case 6
2.3 "Edit Translation" Use Case 9
2.4 Modules Defined by the Use Cases 11

3 API Specification **11**
3.1 Authentication Module API 11
3.2 Document manager API 12
3.3 HTR module API 13
3.4 Translation module API 13
3.4.1 MT-related API 13
3.4.2 IMT-related API 14
3.4.3 Extension to Google translation API 16

4 System Architecture **16**
4.1 Plugin architecture 17
4.2 UPVLC special needs 17
4.3 CBS special needs 17
4.4 System Architecture 18

5 Data Formats And Implementation Details **19**
5.1 Implementation Languages 19
5.2 Database Schemas 19
5.2.1 Users 19
5.2.2 Documents 21
5.2.3 Assignments 21
5.2.4 Sentences 21
5.2.5 Translations 21
5.2.6 Machine Translation Systems 22
5.2.7 Translation Session 22
5.2.8 Session Setup 22
5.2.9 Key Events 22
5.2.10 UI Events 23
5.2.11 Eye Tracking Log 23

6 UI Specification **24**
6.1 Current Translation Environment 24
6.2 Proposal for CASMACAT Translation Environment 24
6.3 Additional Concerns 27
6.4 Concerns regarding eye tracking 28

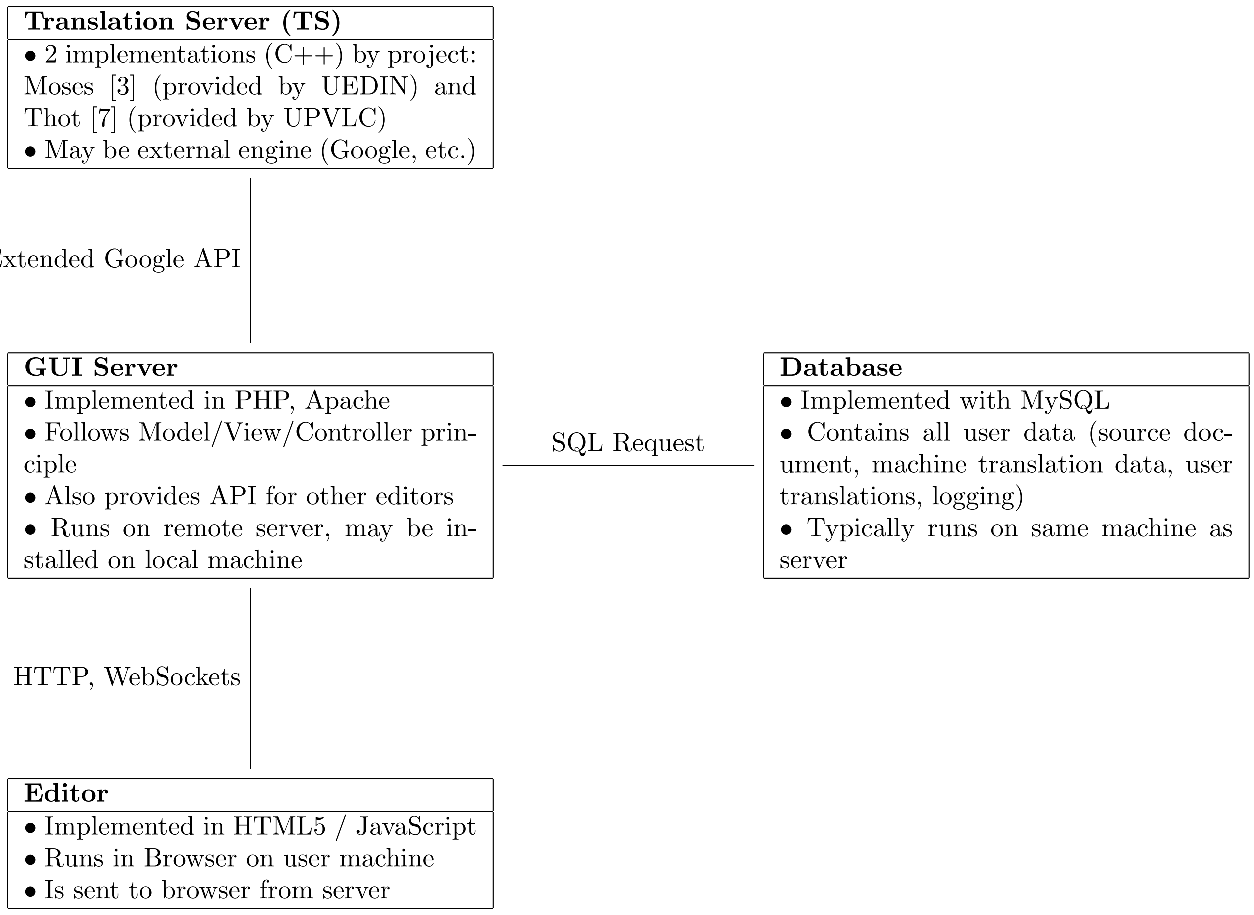
A Alternative System Architecture Proposals **30**
A.1 Architecture Alternative 1 30
A.2 Architecture Alternative 2 31

B Acronyms **34**

1 Introduction

The CASMACAT Workbench allows users to enter documents in a source language, and then receive assistance in translating them into a target language. The assistance is based on information from machine translation systems.

A simplified diagram of major components of the system is as follows:



Translation Server (TS)
• 2 implementations (C++) by project: Moses [3] (provided by UEDIN) and Thot [7] (provided by UPVLC)
• May be external engine (Google, etc.)

GUI Server
• Implemented in PHP, Apache
• Follows Model/View/Controller principle
• Also provides API for other editors
• Runs on remote server, may be installed on local machine

Database
• Implemented with MySQL
• Contains all user data (source document, machine translation data, user translations, logging)
• Typically runs on same machine as server

Editor
• Implemented in HTML5 / JavaScript
• Runs in Browser on user machine
• Is sent to browser from server

From a user perspective, the following installations will be the most common:

- **Basic:** The CASMACAT Workbench is hosted on a remote web site (`http://demo.casmacat.eu`). The user accesses this web site with her web browser. The CASMACAT Editor appears in the browser, and the user can use it to translate documents.
- **Advanced:** The user installs the CASMACAT GUI Server, a machine translation engine and the database on her own machine. The user can then access the Workbench with a web browser as in the above example.

Note that the modularity of the design allows for:

- use of any other machine translation engine, including other commercial ones
- use of any other editor, which supports some or all functionality of the CASMACAT Workbench

2 Use Cases

Before describing the API of the CASMACAT workbench, the basic functionality offered by the CASMACAT editor is defined in terms of a use case diagram. Figure 1 shows a use case diagram for the CASMACAT Editor.

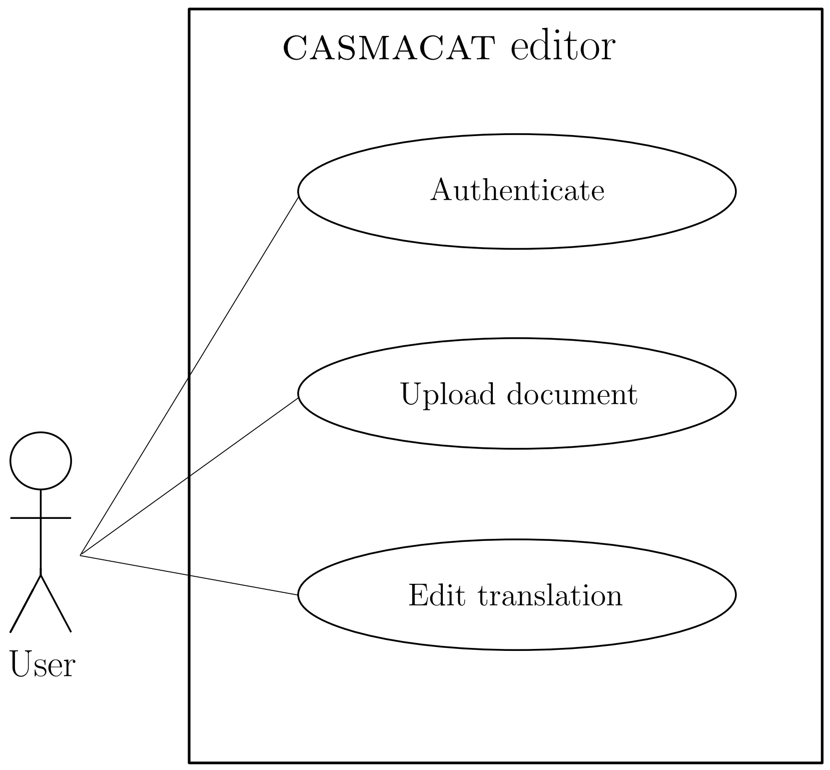


Figure 1: Use case diagram for the CASMACAT editor.

As can be seen in the diagram, the functionality that can be accessed from the CASMACAT editor is expressed by means of three use cases, namely, the "Authenticate", the "Upload Document" and the "Edit Translation" use cases. Below we show a more detailed description of each individual use case.

2.1 "Authenticate" Use Case

The "Authenticate" use case allows the user to identify herself to the CASMACAT workbench. Figure 2 shows an interaction diagram for the "Authenticate" use case. The CASMACAT editor communicates with the authentication module. The process does not differ from a typical authentication process. However, it should be noted that the authentication module returns a user identification (user_id) and a key to the CASMACAT editor. When the editor makes requests to other modules, such as the translation module that is depicted in the diagram, the request includes the user identification and the corresponding key, and the module checks the validity of the key by making a query to the workbench database. The user identification is important since there will be operations requested by the CASMACAT editor that depend on which user is executing them.

2.2 "Upload Document" Use Case

There exist two possible approaches for a translation process in CASMACAT . The first one is "document based" and the second one would be "on-the-fly".

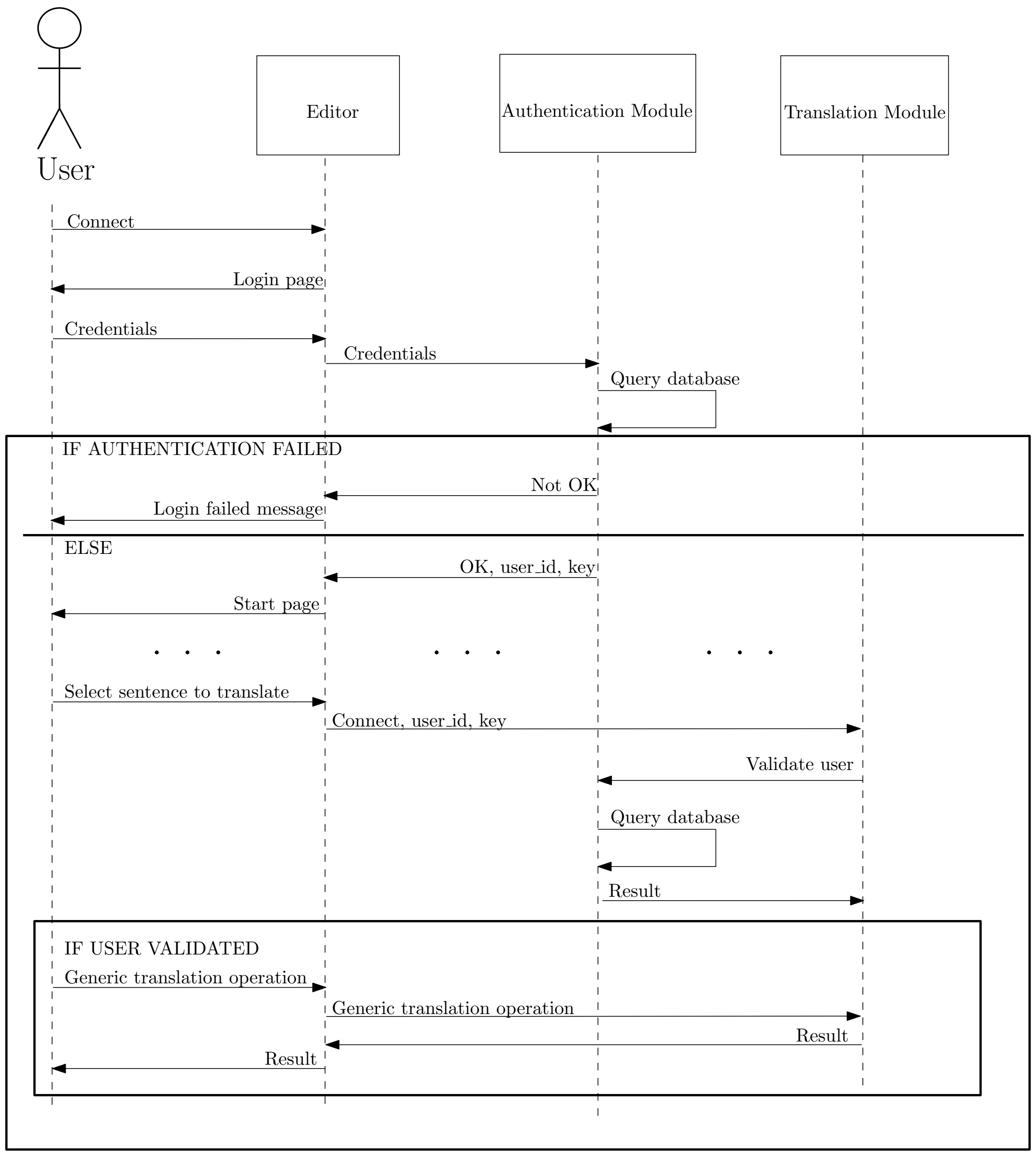


Figure 2: Interaction diagram for the “Authenticate” use case.

For the “document based” approach every source text that should be translated must be in form of a document. This could be a file but in any case it is considered to be a sequence of sentences/segments which has a name. This document is uploaded by a user or admin with the help of the document manager.

Figure 3 shows an interaction diagram for the “Upload Document” use case. After being authenticated, the user accesses the upload document page and chooses a document to be uploaded. The user browses the document to be uploaded and also set the machine translation system that will be used to carry out translations and other MT-related operations (the MT system is identified in the CASMACAT workbench by means of the “mt_sys_id” variable). The document is sent by the editor to the CASMACAT document manager. The document manager has to carry out different tasks for each newly uploaded document:

1. Preprocess the document (sentence splitting, text normalization).
2. Store the preprocessed document in the workbench database.

3. Generate translations for each sentence of the document (using the translation module).
4. Generate word-graphs for each sentence if it was requested (using the translation module).

It is also worthy of note that the “document based” approach described above enables configuration of particular experiments and is a good solution for tracking and logging.

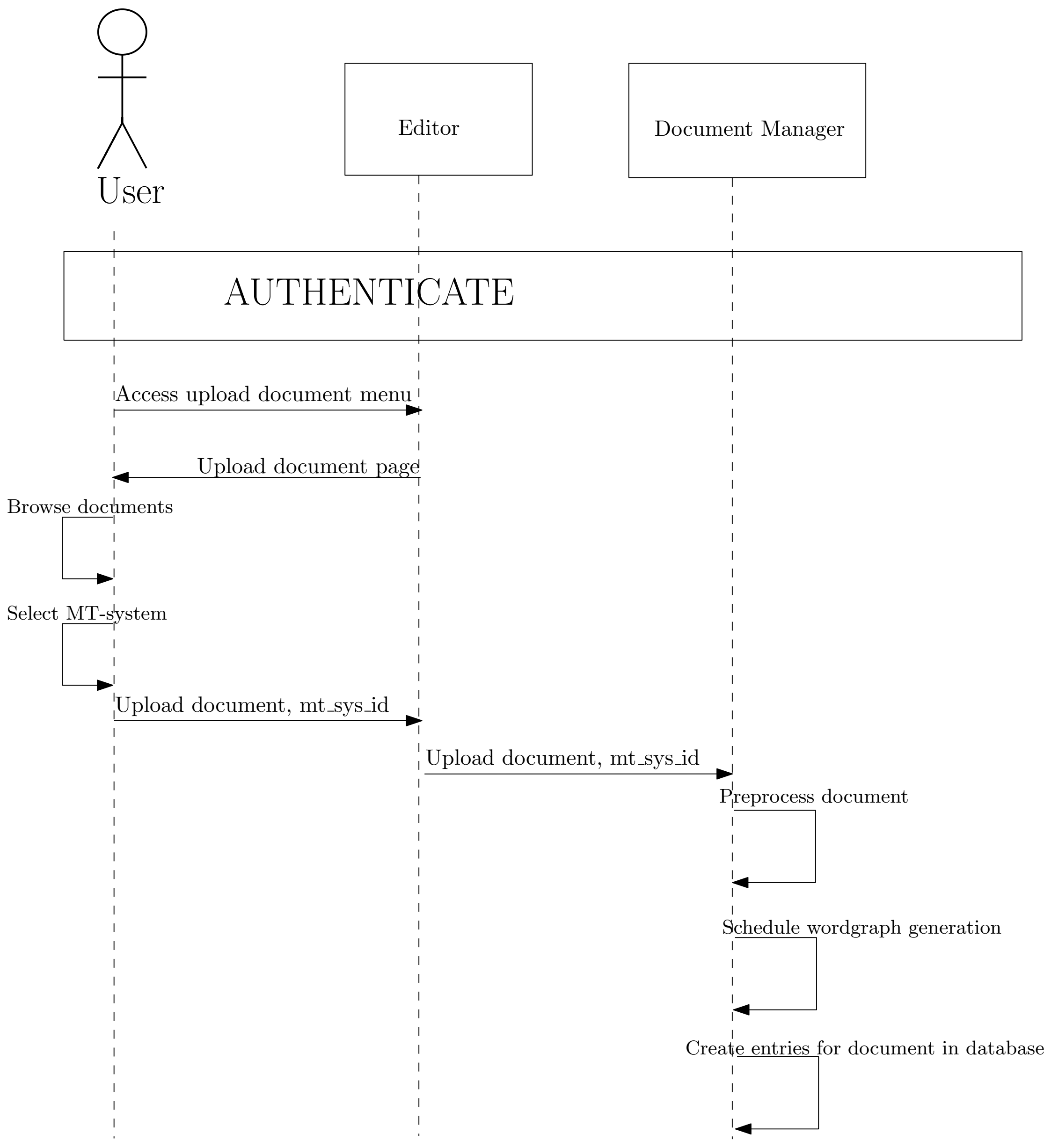

User
Editor
Document Manager
AUTHENTICATE
Access upload document menu
Upload document page
Browse documents
Select MT-system
Upload document, mt_sys_id
Upload document, mt_sys_id
Preprocess document
Schedule wordgraph generation
Create entries for document in database

Figure 3: Interaction diagram for the “Upload document” use case.

For the second approach, “on-the-fly”, only a part of the document or even only one segment or sentence should be translated. That is much more like what Google Translate is doing. Even though this approach seems not to be needed within CASMACAT . But if it is wanted at a later point of the project it can be easily done: A document can be automatically generated from the user’s input and then it can be processed just like described above.

Through the ’document based’ approach it is also always guaranteed that there is a translation present for a source sentence/segment.

2.3 "Edit Translation" Use Case

The "Edit Translation" use case allows the user to access the different translation functionalities that will be developed during the CASMACAT project.

After being authenticated, the users of the CASMACAT editor first select a document from a list of documents for translation. For this purpose, the CASMACAT editor communicates with the document manager. The document manager accesses the CASMACAT database to retrieve the requested document and returns the corresponding web page to the editor. Figure 4 shows a graphic representation of such process.

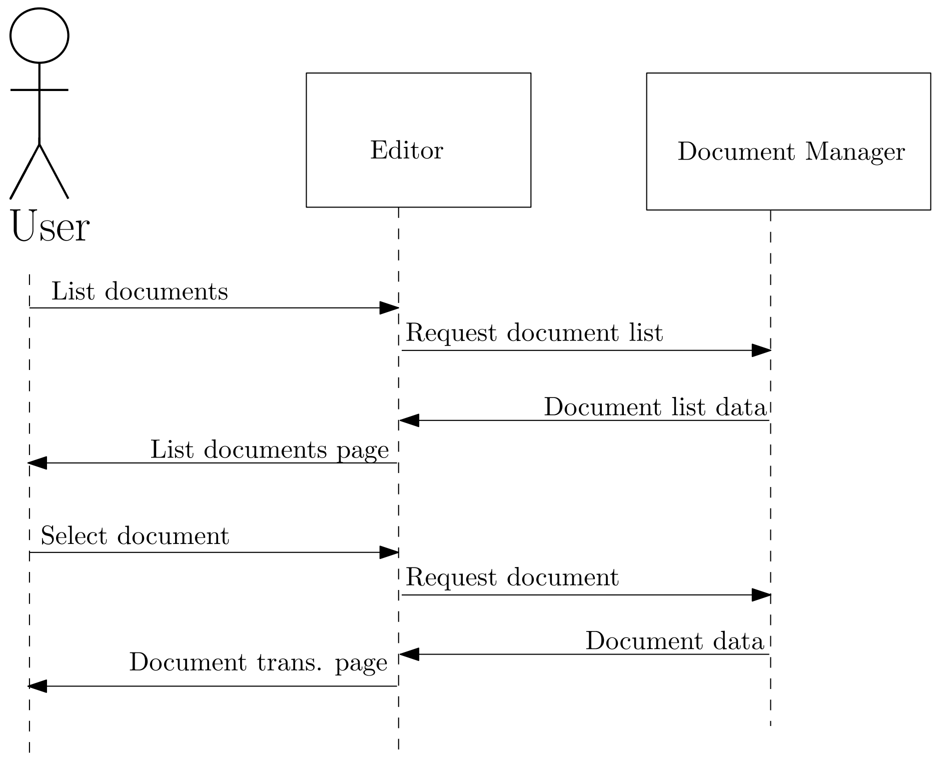


Figure 4: SELECT_DOCUMENT subroutine of the "Edit Translation" use case.

After selecting the document to be translated, the user selects an individual translation in which she is interested to work. The CASMACAT editor requests the translation to the document manager. After that, the editors asks the translation module to prepare itself to translate the current source sentence.

After the editor has retrieved the current translation of the sentence selected by the user, the user can correct it using one of the two operating modes that will be studied in the CASMACAT project, namely, post-edition and interactive machine translation (IMT).

If post-edition is enabled the machine translation is displayed in the edit textbox, and the user may change it in any way. When satisfied, the users presses a submit button (or enter) to send the translation to the server, to be stored in the database.

If the user wants to interactively translate the sentence, then an iterative process is started in which the user partially validates the current translation and the server generates completions for such partial validations.

Figure 5 shows an interaction diagram for the "Edit Translation" use case.

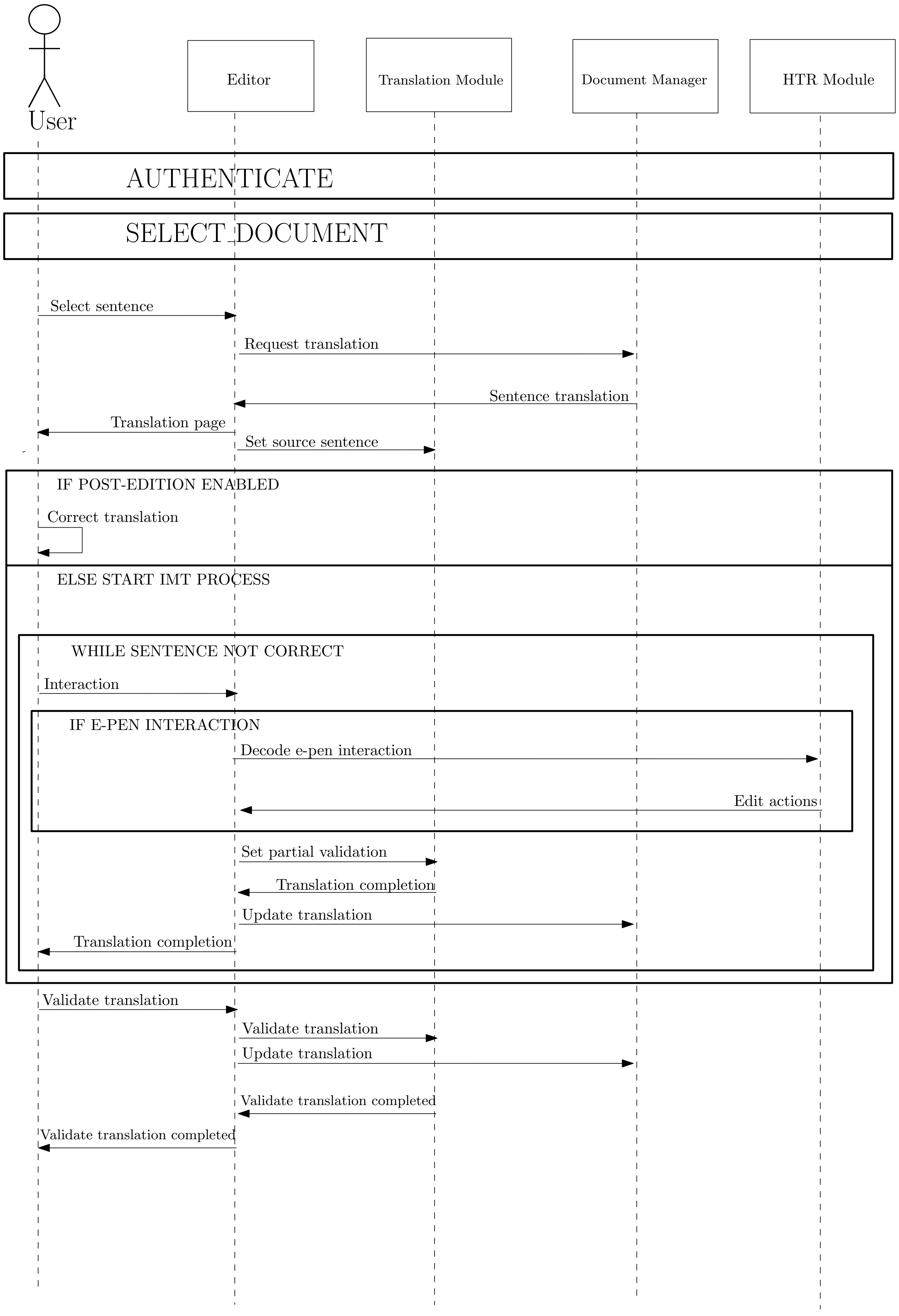


Figure 5: Interaction diagram for the “Edit Translation” use case.

2.4 Modules Defined by the Use Cases

As a by-product of the definition of the different use cases for CASMACAT workbench, a total of four different modules have been defined:

- **Authentication module**: manages user authentication, returns user identifications and keys associated to them to allow other modules to verify the identity of the users.
- **Document manager**: executes differents tasks such as document preprocessing, translation and word-graph generation, etc.
- **HTR module**: converts user e-pen interactions into edit actions. The edit actions determine a partial validation of the target translations that are completed by the translation module.
- **Translation module**: generates translations, word-graphs and completions for partially-validated target sentences.

3 API Specification

In this section, the basic details of the client-server API's are described. Since the CASMACAT editor communicates with different modules, in the following sections several different API's are described. The descriptions are mainly intended to describe the functions as such but not necessarily how the modules will be deployed in the final system architecture.

3.1 Authentication Module API

The authentication module manages user credentials returning a user identification and a related key which is used by other modules to validate the identity of the user.

- **authenticate**
  - **DESCRIPTION**: authenticates a user given her credentials. An error value is returned depending on whether the authentication was successful or not. If the authentication was successful, a user identification and a key associated to it are also returned.
  - **INPUT**:
    - **string login**
    - **string password**
  - **OUTPUT**:
    - **bool error_value**
    - **integer user_id**
    - **integer key**
- **validate_user**
  - **DESCRIPTION**: validates a user given her user_id and its associated key. An error value is returned depending on whether the validation was successful or not. After a module has validated a user, then its user id is kept during the rest of the session. This way, further calls to specific API functions will not require to pass the user id.
  - **INPUT**:
    - **integer user_id**
    - **integer key**
  - **OUTPUT**:
    - **bool error_value**

3.2 Document manager API

The document manager has the function to store and process the documents uploaded by the users, to generate translations for the sentences contained in it and to generate word-graphs if it was requested. This is a dedicated service since the involved tasks could be very sophisticated for different languages.

- **upload_document**
  - **DESCRIPTION**: executes different tasks to preprocess the documents uploaded by the users (text normalization, segment splitting,...), generates an initial translation for each segment. It also generates word-graphs if required (for this purpose, the value of the boolean variable "gen_wg" is inspected). Each document will have an associated machine translation system,"mt_sys_id", which is used to genererate translations and other MT-related operations.
  - **INPUT**:
    - **string document**
    - **integer mt_sys_id**
    - **bool gen_wg**
  - **OUTPUT**:
    - **bool error_value**
- **merge**
  - **DESCRIPTION**: merges the translated segments into a new translated document. For this purpose, "document_id" and "user_id" variables have to be supplied.
  - **INPUT**:
    - **integer document_id**
  - **OUTPUT**:
    - **string document**

Additionally, it is also important to define with more detail the profile of some functions used by the CASMACAT editor to request and update translations:

- **request_translation**
  - **DESCRIPTION**: this function accesses the database of translations governed by the document manager to retrieve the current partial translation of a given source sentence.
  - **INPUT**:
    - **integer doc_id**
    - **integer sentence_id**
  - **OUTPUT**:
    - **integer user_id** (identification of the user that made the last translation update)
    - **string[ ] target**
    - **bool[ ] validated_words**
- **update_translation**
  - **DESCRIPTION**: this function updates the current partial translation of a given source sentence in the database of translations governed by the GUI Server.

  - **INPUT**:
    - **integer doc_id**
    - **integer sentence_id**
    - **integer user_id**
    - **string[ ] target**
    - **bool[ ] validated_words**

3.3 HTR module API

This section describes the API of the HTR module that is used to manage interactions using the e-pen.

- **decode_epen_interaction**
  - **DESCRIPTION**: extracts information contained in an interaction using the e-pen. This information is transformed into edit actions to be applied by the CASMACAT Editor. As a result, a partial validation of the target translation is determined.
  - **INPUT**:
    - **string[ ] source**
    - **string[ ] target**
    - **bool[ ] validated_words**
    - **coordinates[ ] pen_strokes**
  - **OUTPUT**:
    - **edit_action action**
    - **string decoded_target_word**

3.4 Translation module API

The CASMACAT translation module provides a set of functions that allows the user to automatically or interactively translate sentences and to validate translations.

For each function, a brief description and the set of input and output parameters are shown. It is worthy of note that the source and target sentences are represented here as vectors of strings (**string**[ ]) instead of as strings. This representation is required to appropriately assign properties to the words that compose the sentences, such as confidence scores and word alignments, that are represented as vectors and matrices, respectively.

3.4.1 MT-related API

This section describes the MT system's API based on Moses.

- **translate**
  - **DESCRIPTION**: translates sentence specified as 'text'. If 'align' switch is on, phrase alignment is returned. If 'sg' is on, search graph is returned. If 'topt' is on, phrase options used are returned. If 'report-all-factors' is on, all factors are included in output. 'presence' means that the switch is on, if the category appears in the xml, value can be anything
  - **INPUT**:

    - **string text (required)**
    - **presence align (optional)**
    - **presence sg (optional)**
    - **presence topt (optional)**
    - **presence report-all-factors (optional)**
  - **OUTPUT**:
    - **string text**
    - **ALIGN align (if requested)**
    - **SG sg (if requested)**
    - **TOPT topt (if requested)**

- **update**
  - **DESCRIPTION**: updates a suffix array phrase table. If 'bounded' switch is on, seems to do nothing at the moment. If 'updateORLM' is on, a suffix array language model is also updated.
  - **INPUT**:
    - **string source (required)**
    - **string target (required)**
    - **string alignment (required)**
    - **presence bounded (optional)**
    - **presence updateORLM (optional)**

3.4.2 IMT-related API

This section describes the IMT related API. The IMT related API allows users to interactively translate sentences. The API includes functions to translate a source sentence, to partially validate translations and to validate a target sentence as an error-free translation of a given source sentence. This API is based on previous experience of UPVLC on IMT [8].

- **translate_source_sentence**
  - **DESCRIPTION**: this function translates a given source sentence.
  - **INPUT**:
    - **string[ ] source**
  - **OUTPUT**:
    - **string[ ] target**

- **set_partial_validation**
  - **DESCRIPTION**: this function is intended to be used in an IMT framework. Specifically, it allows the user to obtain a translation completion given a target translation that has been partially validated.
  - **INPUT**:
    - **string[ ] target**
    - **bool[ ] validated_words**
  - **OUTPUT**:
    - **string[ ] target**
    - **bool[ ] validated_words**

- **set_prefix**
  - **DESCRIPTION**: this function is a specialization of the set_partial_validation function. Specifically, it allows the user to obtain a translation completion given a target translation prefix, which is assumed to be validated, and the current suffix, which is not validated and can be discarded by the IMT system.
  - **INPUT**:
    - **string[ ] prefix**
    - **string[ ] suffix**
  - **OUTPUT**:
    - **string[ ] suffix**
- **get_alignments**
  - **DESCRIPTION**: this function computes an alignment matrix between the source and target sentence.
  - **INPUT**:
    - **string[ ] source**
    - **string[ ] target**
  - **OUTPUT**:
    - **bool[ ][ ] word_alignment**
- **get_translation_confidence**
  - **DESCRIPTION**: this function computes the confidence of the current translation given the source sentence.
  - **INPUT**:
    - **string[ ] source**
    - **string[ ] target**
    - **bool[ ] validated_words**
  - **OUTPUT**:
    - **float confidence_score**
- **get_word_confidences**
  - **DESCRIPTION**: this function computes the confidence of each target word given the source sentence.
  - **INPUT**:
    - **string[ ] source**
    - **string[ ] target**
    - **bool[ ] validated_words**
  - **OUTPUT**:
    - **float[ ] confidence_scores**
- **validate_translation**
  - **DESCRIPTION**: this function is used to validate a target sentence as an error-free translation of a given source sentence. The server can extend the parameters of its statistical translation models using this newly generated training pair.
  - **INPUT**:
    - **string[ ] source**
    - **string[ ] target**
    - **bool[ ] validated_words**

Note: **get_alignments**, **get_translation_confidence** and **get_word_confidences** are standalone functions since they represent optional features and should have a less frequent invocation rate than **set_partial_validation** and **set_prefix**. Besides, **get_translation_confidence** belongs to a separate task (3.1) whereas **get_word_confidences** belongs to Task 3.2. Thus, being separated functions will reduce their coupling and allow the developers to work independently from the other modules.

3.4.3 Extension to Google translation API

During discussion regarding the API specification, the project partners stressed the interest of extending the CASMACAT translation API to cover the basic functionality provided by the Google translation API.

Currently, the Google translation API provides three different functions:

- **translate**: translates source text from source language to target language.
- **languages**: list the source and target languages supported by the translate methods.
- **detect**: detect language of source text.

Among the above described functions, it is interesting to show the profile of the "translate" function:

- **translate**
  - **DESCRIPTION**: translates source text from source language to target language.
  - **INPUT**:
    - **key**
    - **string source**
  - **OUTPUT**:
    - **string target**

It should be noted that the "translate" function can be easily implemented in terms of the functions already defined in the CASMACAT translation API.

Interactive MT and enriched editing requires additional information from the machine translation system. This can be requested with additional parameters, such as:

- **sg** — request search graph
- **topt** — request translation options

4 System Architecture

An important thing at the beginning of the development of a new software is to agree on the system's architecture. The architecture influences wide parts of the software and is very hard to change afterwards. It also has an impact on API specification and is most likely more important than choosing a language for the implementation.

This section discusses the architecture designed for the CASMACAT workbench. Great efforts have been put into this part to find a common suitable solution. As there are very different needs, it was difficult to find a perfect solution. Even though there exists a version that is basically agreed on by the partners, there also exists alternatives that may become more reliable within the progress of the CASMACAT project. These alternatives are also included in this document for further reference if that was required (see appendix A).

Before presenting the CASMACAT system architecture, there are some basic clarifications of the plugin architecture and the special needs of the partners that are explained below.

4.1 Plugin architecture

A plugin architecture has been designed to facilitate the deployment of the different engines resulting from WP2, WP3 and WP4 research. The plugin architecture should use dynamic libraries to load the engines in run-time. The pimpl idiom [10] or the bridge pattern [9] should be used to provide binary compatibility [6].

This way, the engines and the server can be compiled separately only depending on a couple of header files (without linking dependencies). Furthermore, as these headers are expected to provide a minimalistic interface which should stabilise in early steps of development, binary compatibility with older engine versions will be probably valid for long periods of time. In addition to this, the pimpl idiom will not force the developers to change their data structures but just require them to write a handful of methods for data conversion. UPVLC has experience in a similar architecture involving several systems (thot, wordgraph server, moses) with satisfactory results [1].

4.2 UPVLC special needs

IMT systems impose some restrictions over the communication protocol, specially since requests must be served as real-time as possible and interactions are performed on character/keypress basis. For that matter, UPVLC proposed the rest of the partners to reconsider the use of xml-rpc requests in favour of json-rpc requests [2]. Although there is quite a bit of discussion in the web concerning which protocol is best (e.g. [5]), the consensus can be resumed in the following items:

- Pros of using json-rpc
  - Lightweight protocol, ie. higher chances to fit in the *maximum transmission unit* (MTU). Internet communication is the major source for lag. Hence, reducing the number of messages to pass is crucial.
  - Faster parsing.
  - Better integration with JavaScript and HTML5.
  - More human readable.
- Cons of using json-rpc
  - Less mature.
  - No type checking.

Since the client seems to be implemented in a web browser and the IMT system could benefit from extra speed, json-rpc seems to suit better the project's needs. Regarding the cons, although json-rpc is less mature than xml-rpc, it is already quite mature. A reasonable amount of open source libraries in the most popular programming languages can be found in the web.

4.3 CBS special needs

Logging the user actions (key strokes, e-pen usage and eye tracking data) and also the results of those actions must be correct and fast. For instance, also the predictions made by the IMT system must be stored because otherwise they will not to be replayed correctly as they can change over time. So it is a must that every action and its results are communicated to the module that is responsible for logging. And also the replay must work correctly and reproduce the session without errors. Depending on the translation session length and the amount of user actions this can be much data. How much that actually will be is currently difficult to know but it is approximately about 60MB for only the eye tracking data collected within 30 minutes. So it is intended to increase and we are not sure for now if the data should be communicated in real time, collected offline or if some buffering and then sending it in chunks is the best approach. But as at least key strokes and e-pen actions are commands that need to be sent to the appropriate module, it seems that the first approach is a good solution as everything will be sent twice otherwise.

4.4 System Architecture

After extensive discussion between the project partners, a specific system architecture was agreed. This system architecture tries to satisfy the set of special needs that were explained above.

Figure 6 shows a diagram of the architecture of the CASMACAT Workbench. According to the figure, there are five major components:

- **Editor**: is the interface between the user and the functionalities provided by the CASMACAT workbench. It communicates with two different components: the GUI server and the multimodal translation server.
- **GUI server**: serves web pages to the editor interface. Handles logging and replay information.
- **Translation server**: provides translation services including regular MT and IMT and the possibility to link an external translator server. In addition to this, it is also used for authentication and document management purposes.
- **HTR server**: handles user interactions by means of an e-pen.
- **Database**: two databases have been identified. the first one should be visible to the translation server and store user information, documents, partial or total translations of the documents. The second one should be visible to the GUI server and store replay information. The database does not store the the statistical parameters needed to implement the MT functionality.

In the proposed architecture, the GUI server, the translation server and the HTR server constitute separated entities. First, because the translation server should be decoupled from the GUI server to allow different translation clients, e.g. a batch command-line client for experimentation. Second, a physical separation will provide a more robust environment (the translation and the HTR servers are a complex piece of software) and will facilitate distributed computing. Furthermore, the HTR server has been separated from the Translation server since a client might not implement HTR features, e.g. a command-line client for experimentation purposes. Finally, certain functions included in the "Edit translation" use case, specifically the IMT and HTR functionality, impose certain performance constraints. For that reason, websockets are used. However, current websocket proxies for popular webservers are not quite mature yet. If a proper technology matures during the development of the project, then we will consider to put the translation server behind the web server.

Due to such time constraints, web sockets are used to directly communicate the graphical interface and the multimodal translation server.

The definition of the use cases shown in the previous section produced a total of four modules, namely, authentication, document manager, HTR and translation modules. According to Figure 6, the HTR module is deployed in the HTR server and the rest of the modules are deployed inside the translation server. The graphical interface accesses the functionality provided by the translation server by means of the APIs for the different modules described in section 3.

In previous versions of our system architecture proposal, the document manager and the authentication functionalities were managed by the GUI server. The design has been modified due to the following reasons:

- Document manager functionalities are language related. The document manager module may need to access the funcionality provided by the translation module to obtain translations for the source sentences or generate wordgraphs. Since the translation module is deployed in the multimodal translation server, the document manager fits more naturally inside the same server.
- Authentication has also strong connections with translation functionality, since the MT- and IMT-related functions require user information to work properly, especially for adapting models in online learning. In addition to this, the tasks performed by the document manager also require information about users. If the authentication module is deployed in the GUI server, the multimodal translation server cannot be used independently, reducing the modularity of the resulting components.

5 Data Formats And Implementation Details

In this section, the data formats and other implementation details of the CASMACAT Workbench are described.

5.1 Implementation Languages

The implementation languages have been defined at an early stage of the project. For most of the MT or IMT parts that will be C++ as those have already been implemented in this language. For the GUI Server this will be PHP running on Apache and HTML with JavaScript using the JQuery engine. The database will be a MySQL database for all components using a database. The browser extensions will also be implemented in C++.

5.2 Database Schemas

This section describes the basic database tables in which user data is stored. The elements marked with an asterisk are additions to the original specification.

5.2.1 Users

Field	Type	Description
id	int(11)	unique identifier
name	varchar(255)	textual descriptor
email	varchar(255)	verified email address
password*	varchar(255)	encrypted version of the password to test against
status	varchar(255)	'new', 'verified', 'blocked', 'admin'

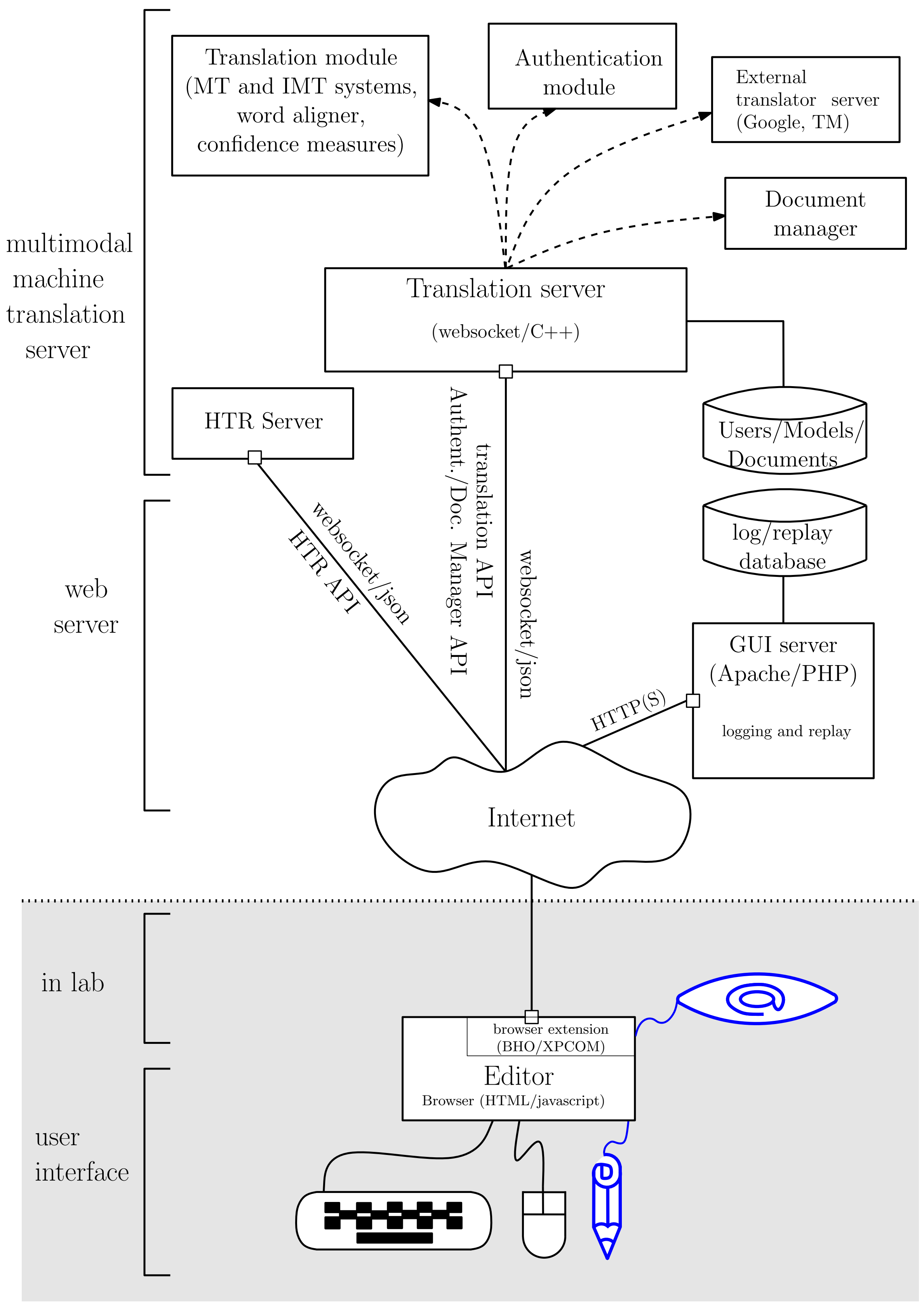


Figure 6: Proposal of system architecture.

5.2.2 Documents

Field	Type	Description
id	int(11)	unique identifier
name	varchar(255)	textual descriptor
status	varchar(255)	'new', 'translated', 'edited', 'reviewed'
text	text	raw content with formatting
user-id	int(11)	unique identifier of owner
system-id	int(11)	unique identifier of machine translation system used

5.2.3 Assignments

Documents are assigned to users to edit. The same document may be assigned to multiple users.

Field	Type	Description
id	int(11)	unique identifier
document-id	int(11)	unique identifier of the document related to this assignment
user-id	int(11)	unique identifier of the user tasked with this this document
name	text	textual descriptor

5.2.4 Sentences

Each document consists of a number of sentences. Editing happens sentence by sentence.

Field	Type	Description
id	int(11)	unique identifier
document-id	int(11)	unique identifier of the document to which sentence belongs
input	text	raw input sentence
tokenized	text	tokenized version of input sentence
mt-output	text	raw machine translation output
markup	text	meta information about markup
start-state	int(11)	unique identifier of start state of search graph (in 'state' table)

Details about the meta information in markup still needs to be defined. It should contain information about inter-sentential (paragraph breaks), sentential (format), and intra-sentential (bold, underlined, italic words) properties.

5.2.5 Translations

When users edits a sentence, the result is stored in this table.

Field	Type	Description
id	int(11)	unique identifier
sentence-id	int(11)	unique identifier of the sentence of which this is a translation
assignment-id	int(11)	unique identifier of the assignment to which this translation belongs
translation	text	translation entered by the user
tokenized*	text	tokenized version of translation
validated-words*	set	array of bits indicating which words have already been validated by the user

5.2.6 Machine Translation Systems

Field	Type	Description
id	int(11)	unique identifier
url	varchar(255)	web address of server
preprocessor	varchar(255)	web address of preprocessor to be run (incl. tokenization)
postprocessor	varchar(255)	web address of postprocessor to be run

5.2.7 Translation Session

This table contains all information that is fix for one translation session.

Field	Type	Description
id	int(11)	unique identifier
user-id	int(11)	unique identifier of the user assigned with that session
setup-id	int(11)	unique identifier of the assigned session setup
start-time	datetime	start time of session
end-time	datetime	end time of session

5.2.8 Session Setup

The setup for sessions/experiments is stored here.

Field	Type	Description
id	int(11)	unique identifier
document-id	int(11)	unique identifier of the document that has been translated
description	text	description of the session/experiment
version	varchar(11)	version of Casmacat used to log this session
show-timer	boolean	should a timer be shown while the session runs/is replayed
ca-width	int(6)	width in pixel of the browsers client area
ca-height	int(6)	height in pixel of the browsers client area
css-file	varchar(255)	css file used

5.2.9 Key Events

Field	Type	Description
id	int(11)	unique identifier
translation-session-id	int(11)	unique identifier of the translation session in which this event occurred
translation-id	int(11)	unique identifier of translation in which this event occurred
type	varchar(255)	type of event (press, up, release)
time	int(11)	time of event (ms offset from session start)
ui-element-id	int(11)	ui element which generated the event
translation	text	text of translation at time of event
cursor-offset	int(11)	offset of the cursor in the target text field before the key event occurs
key-code	varchar(11)	JavaScript key code
char-code	varchar(11)	JavaScript character code
character	varchar(1)	UTF-8 character
type-of-operation	varchar(255)	type of event (i.e. normal typing, paste, delete, ...)
markedText	text	text that has been selected
pastedText	text	text that has been pasted
shift-key	tinyint	shift key pressed
ctrl-key	tinyint	ctrl key pressed
alt-key	tinyint	alt key pressed

5.2.10 UI Events

Field	Type	Description
id	int(11)	unique identifier
translation-session-id	int(11)	unique identifier of the translation session in which this event occurred
translation-id	int(11)	unique identifier of translation in which this event occurred
type	varchar(255)	type of event (press, up, click)
time	int(11)	time of event (ms offset from session start)
ui-element-id	int(11)	ui element which generated the event
cursor-offset	int(11)	offset of the cursor in the target text field before the ui event occurs
old-window-size	varchar(10)	window size before event
new-window-size	varchar(10)	window size after event
old-window-pos	varchar(10)	window position before event
new-window-pos	varchar(10)	window position after event
mouse-x	int(11)	mouse x position
mouse-y	int(11)	mouse y position
mouse-button	varchar(10)	mouse button
vertical-scroll-offset	int(11)	scroll offset of scrollbar

5.2.11 Eye Tracking Log

Field	Type	Description
id	int(11)	unique identifier
sys-time	datetime	time stamp of the system (browser)
tracker-time	datetime	time stamp of the eye tracker
xl	int(6)	x position of the left eye
yl	int(6)	y position of the left eye
xr	int(6)	x position of the right eye
yr	int(6)	y position of the right eye
pl	decimal(3,3)	pupil dilation of the left eye
pr	decimal(3,3)	pupil dilation of the right eye
fx	int(6)	x position (center) of fixation
fy	int(6)	y position (center) of fixation
gazed-comp	varchar(20)	ui component looked at (src/tgt text field and other components)
cursor-offset	int	offset of the cursor in the gazed component

6 UI Specification

This section discuses the specification of the user interface (UI) to be evaluated by CS in the WP6. Since the UI is to be evaluated by professional translators, it is of uttermost importance that the UI matches their expectations. To this respect, CS has provided a series of screenshots of a typical translation environment (TE) with Trados and a proposal for the CASMACAT TE. All of them will assume a 16:9 display, which is a *de facto* standard in CS.

6.1 Current Translation Environment

To begin with, a sketch of Trados TE (TE0) using SDL Trados Workbench 2007 + Microsoft Word or Tag Editor is shown in Figure 7. In the upper window, the Trados Workbench shows the information retrieved from the translation memory, along with terminology information, if present. The lower window it is shown a Microsoft Word editor, which has a component to communicate with the Trados Workbench. Both windows are separated applications and run in the front. In addition, back windows might include Internet browser, source document in original format, or reference documents among others. Finally, the Trados Tag editor replaces Microsoft Word as editor when source file format is a tagged file (with rich text content), as opposed to a plain text file.

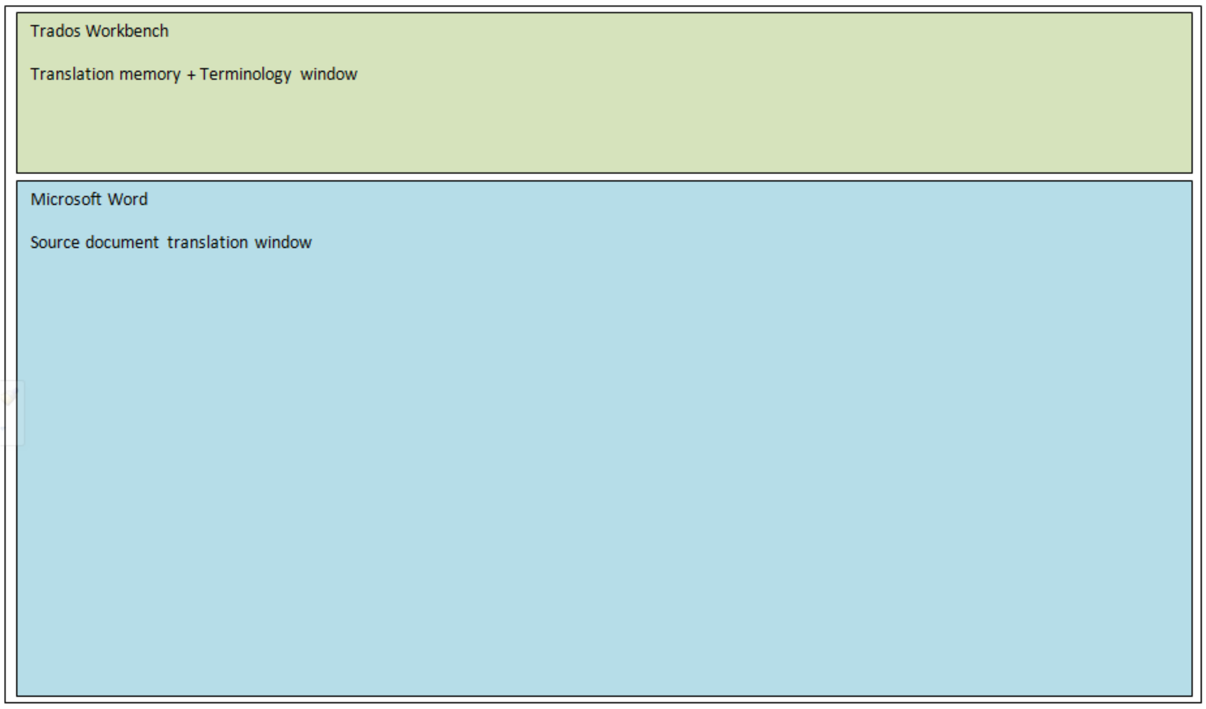
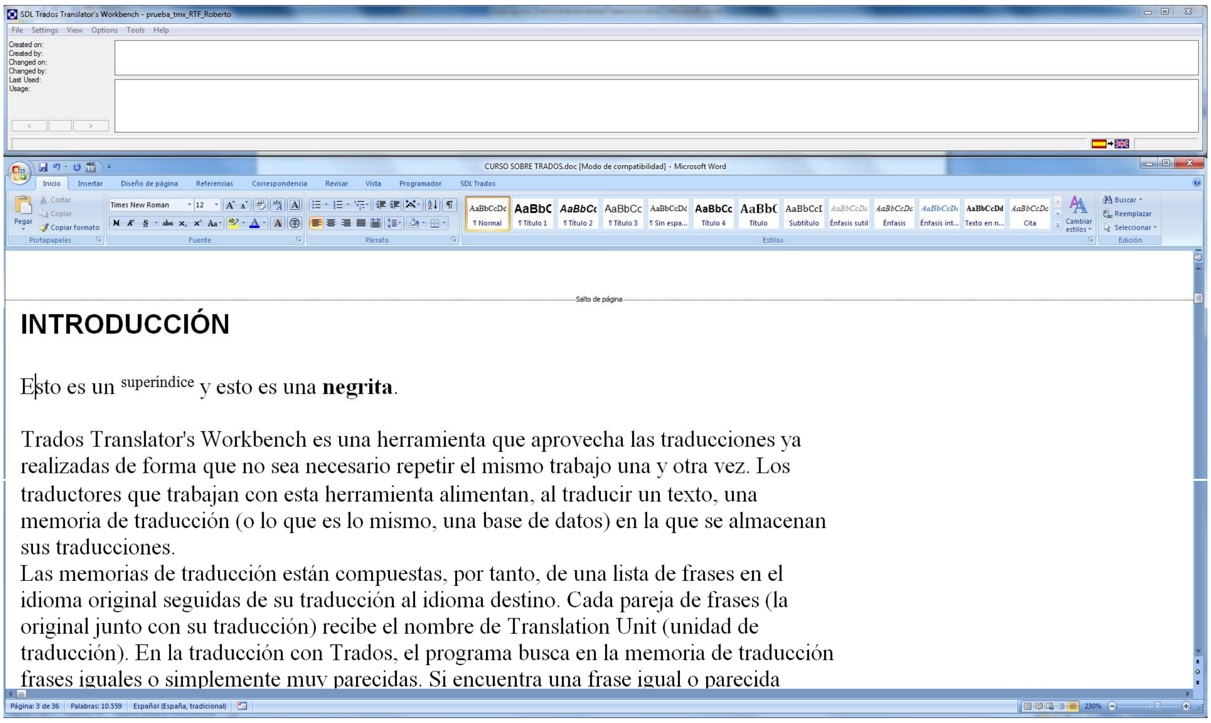

Trados Workbench
Translation memory + Terminology window
Microsoft Word
Source document translation window

Figure 7: Sketch of the current translation workbench in CS.

Figures 8, 9 and 10 show screenshots for different states of the current Trados Workbench.

6.2 Proposal for CASMACAT Translation Environment

CS has proposed two alternative TEs which must have the same functionality and features. Only the representation diverges. Both alternatives should be easily implementable in the web browser by means of style sheets (CSS) and probably minor structure modifications using the document object model (DOM) API.


SDL Trados Translator's Workbench - prueba_tmx_RTF_Roberto
File Settings View Options Tools Help
CURSO SOBRE TRADOS.doc [Modo de compatibilidad] - Microsoft Word
Inicio Insertar Diseño de página Referencias Correspondencia Revisar Vista Programador SDL Trados
Salto de página
Página: 3 de 36 Palabras: 10.559 Español (España, tradicional)

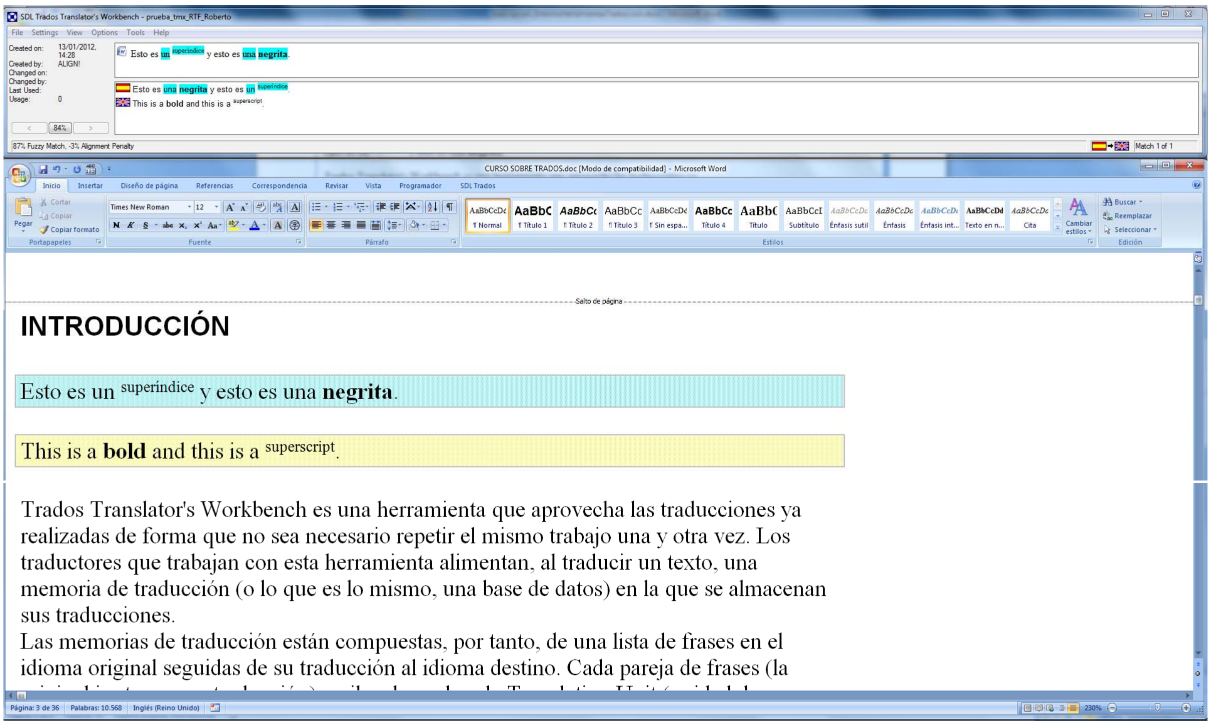
Created on:
Created by:
Changed on:
Changed by:
Last Used:
Usage:

INTRODUCCIÓN

Esto es un $^{\text{superíndice}}$ y esto es una **negrita**.

Trados Translator's Workbench es una herramienta que aprovecha las traducciones ya realizadas de forma que no sea necesario repetir el mismo trabajo una y otra vez. Los traductores que trabajan con esta herramienta alimentan, al traducir un texto, una memoria de traducción (o lo que es lo mismo, una base de datos) en la que se almacenan sus traducciones.
Las memorias de traducción están compuestas, por tanto, de una lista de frases en el idioma original seguidas de su traducción al idioma destino. Cada pareja de frases (la original junto con su traducción) recibe el nombre de Translation Unit (unidad de traducción). En la traducción con Trados, el programa busca en la memoria de traducción frases iguales o simplemente muy parecidas. Si encuentra una frase igual o parecida

Figure 8: Translation environment using SDL Trados Workbench 2007 + Microsoft Word (no segment open).


SDL Trados Translator's Workbench - prueba_tmx_RTF_Roberto
File Settings View Options Tools Help
CURSO SOBRE TRADOS.doc [Modo de compatibilidad] - Microsoft Word
Inicio Insertar Diseño de página Referencias Correspondencia Revisar Vista Programador SDL Trados
Salto de página
Página: 3 de 36 Palabras: 10.568 Inglés (Reino Unido)

Created on: 13/01/2012, 14:28
Created by: ALIGN!
Changed on:
Changed by:
Last Used:
Usage: 0

Esto es un $^{\text{superíndice}}$ y esto es una **negrita**.

Esto es una **negrita** y esto es un $^{\text{superíndice}}$.
This is a **bold** and this is a $^{\text{superscript}}$.

84%

87% Fuzzy Match, -3% Alignment Penalty

Match 1 of 1

INTRODUCCIÓN

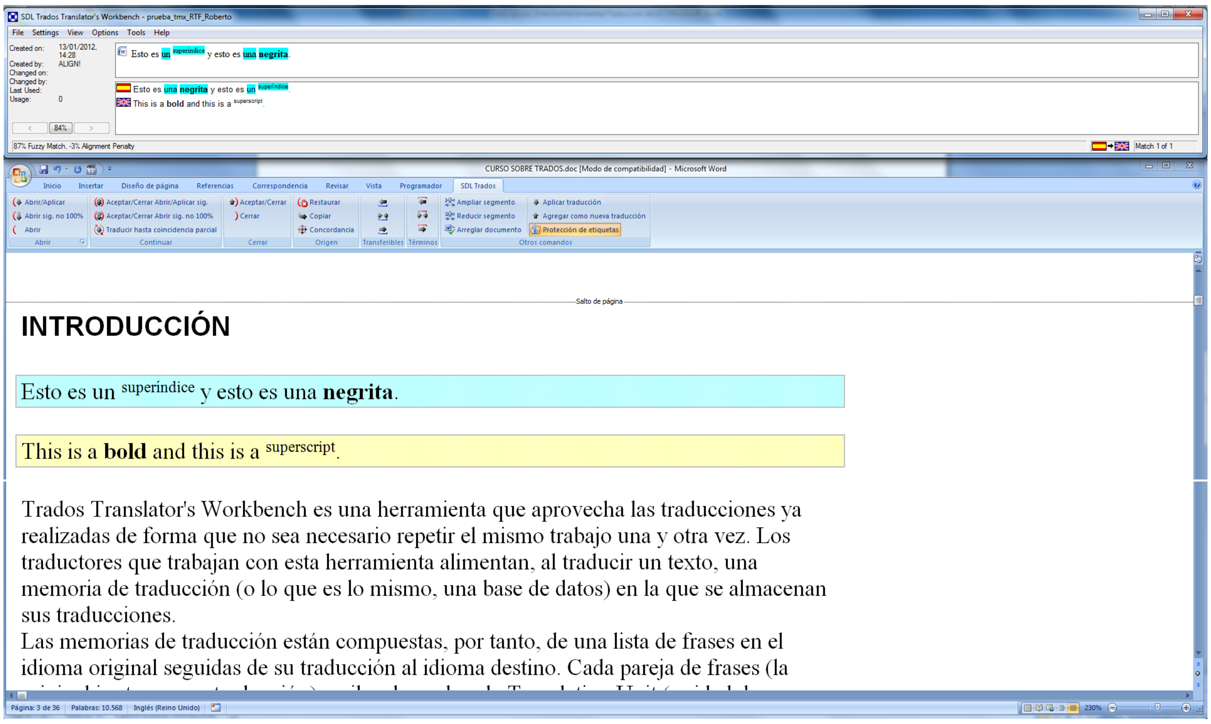
Esto es un $^{\text{superíndice}}$ y esto es una **negrita**.

This is a **bold** and this is a $^{\text{superscript}}$.

Trados Translator's Workbench es una herramienta que aprovecha las traducciones ya realizadas de forma que no sea necesario repetir el mismo trabajo una y otra vez. Los traductores que trabajan con esta herramienta alimentan, al traducir un texto, una memoria de traducción (o lo que es lo mismo, una base de datos) en la que se almacenan sus traducciones.
Las memorias de traducción están compuestas, por tanto, de una lista de frases en el idioma original seguidas de su traducción al idioma destino. Cada pareja de frases (la

Figure 9: Translation environment using SDL Trados Workbench 2007 + Microsoft Word (one segment is being translated; Word commands toolbar).

The first TE (TE1, see Figure 11) displays the text as running words, in a similar fashion to the Trados TE. On the upper side, the translated text is shown (if possible). On user configuration, the source text may be also displayed. On the bottom side, the remaining (source) text to be translated, as the document is usually translated sequentially. In the middle, it is


SDL Trados Translator's Workbench - prueba_tmx_RTF_Roberto

File Settings View Options Tools Help

Created on: 13/01/2012, 14:28
Created by: ALIGN!
Changed on:
Changed by:
Last Used:
Usage: 0

Esto es un ^{superíndice} y esto es una **negrita**.

Esto es una **negrita** y esto es un ^{superíndice}.
This is a **bold** and this is a ^{superscript}.

87% Fuzzy Match, -3% Alignment Penalty

CURSO SOBRE TRADOS.doc [Modo de compatibilidad] - Microsoft Word

INTRODUCCIÓN

Esto es un ^{superíndice} y esto es una **negrita**.

This is a **bold** and this is a ^{superscript}.

Trados Translator's Workbench es una herramienta que aprovecha las traducciones ya realizadas de forma que no sea necesario repetir el mismo trabajo una y otra vez. Los traductores que trabajan con esta herramienta alimentan, al traducir un texto, una memoria de traducción (o lo que es lo mismo, una base de datos) en la que se almacenan sus traducciones.
Las memorias de traducción están compuestas, por tanto, de una lista de frases en el idioma original seguidas de su traducción al idioma destino. Cada pareja de frases (la

Página: 3 de 36 Palabras: 10.568 Inglés (Reino Unido)

Figure 10: Translation environment using SDL Trados Workbench 2007 + Microsoft Word (one segment is being translated; Workbench commands toolbar).

shown the **active segment**. This part must be highlighted with bigger fonts and different background color to give a better focus. On the right side, a tabbed area is used to display translation alternatives, terminology and, probably, internet search. This tabs can gain focus according to user commands. Hotkeys are crucial for high-performance translators. Thus, they should be configurable, though they should default to the ones in Trados when possible. A list of hotkeys can be found in this list:

Move around the screen

Move around the segment

Focus on different areas

Copy source segment to target This is specially interesting for segments composed primarily by name entities, dates, etc.

Copy categories, tags and other entities from source to target Copy name entities, dates, glossary terms from source to target.

Other TBD

The position and visibility of windows should be configurable, e.g. to maximize the translation window. CS also proposes the possibility of adding comments to the translations, visible to other users, and a one click shortcut to preview the current target document. It also would be interesting to have an area for instant messaging.

The second translation environment (TE2, see Figure 12) differs from TE1 in the central part of the environment. In this case, instead of running text, the segments are shown in columns. The first column shows source segments whereas the second column shows target sentences if already edited by the translator. As in TE1, the active segment must be highlighted and present the same functionality, although in this case it is shown in two columns. Note that TE2 is suitable for high resolution wide screens.

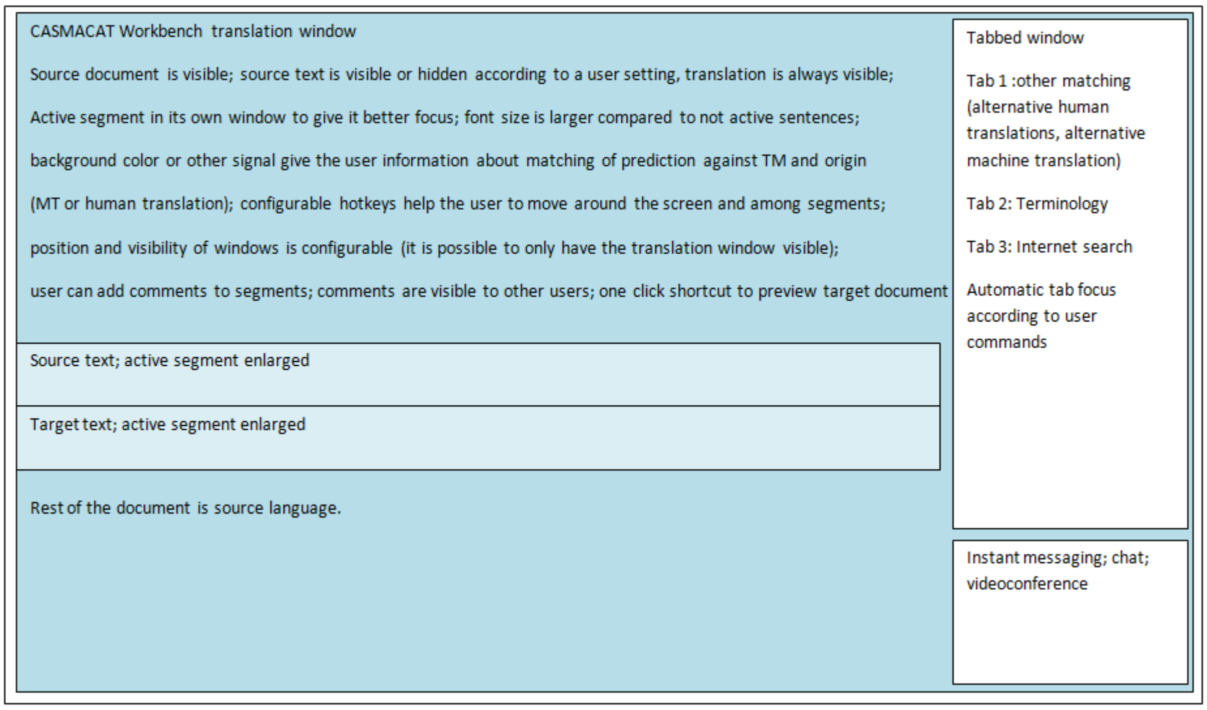

CASMACAT Workbench translation window
Source document is visible; source text is visible or hidden according to a user setting, translation is always visible;
Active segment in its own window to give it better focus; font size is larger compared to not active sentences;
background color or other signal give the user information about matching of prediction against TM and origin
(MT or human translation); configurable hotkeys help the user to move around the screen and among segments;
position and visibility of windows is configurable (it is possible to only have the translation window visible);
user can add comments to segments; comments are visible to other users; one click shortcut to preview target document
Source text; active segment enlarged
Target text; active segment enlarged
Rest of the document is source language.
Tabbed window
Tab 1 :other matching (alternative human translations, alternative machine translation)
Tab 2: Terminology
Tab 3: Internet search
Automatic tab focus according to user commands
Instant messaging; chat; videoconference

Figure 11: Translation environment 1 (TE1) with CASMACAT web translation.

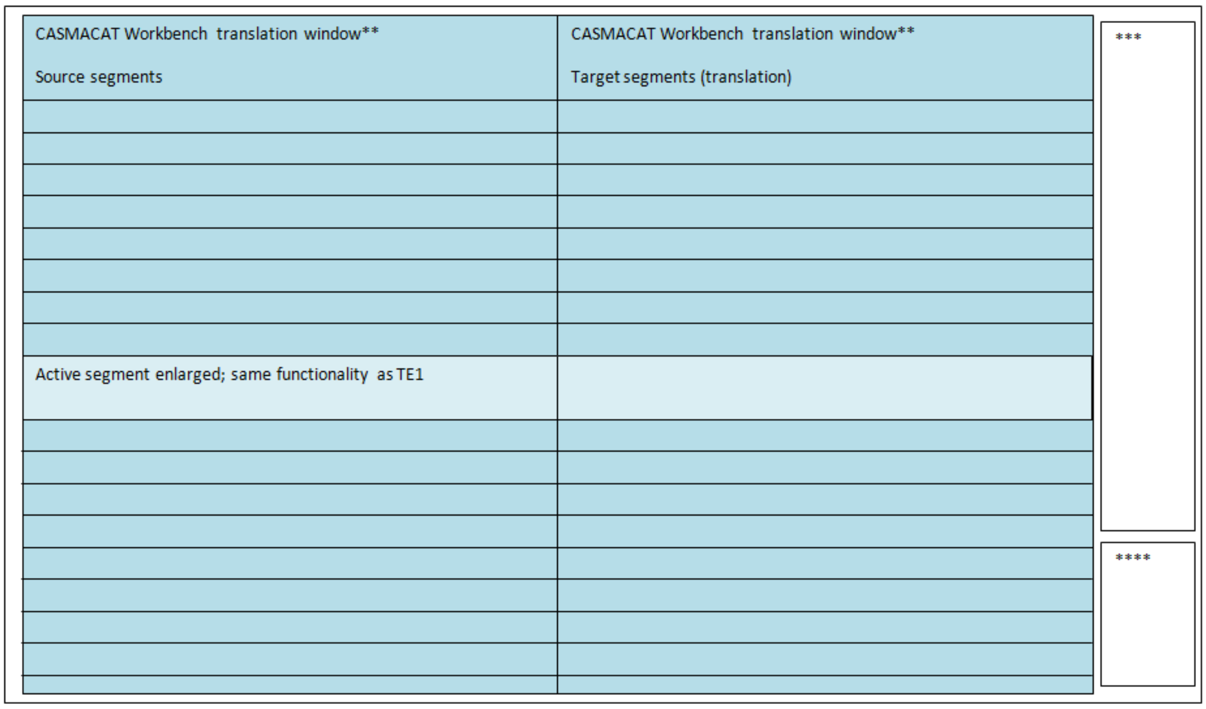

CASMACAT Workbench translation window**
Source segments
CASMACAT Workbench translation window**
Target segments (translation)
Active segment enlarged; same functionality as TE1
***
****

Figure 12: Translation environment 2 (TE2) with CASMACAT web translation.

6.3 Additional Concerns

It is unclear in the project documentation to what extent translation memories (TM) are being integrated in the UI (Task 3.3). TM are to be integrated in Moses as very large translation rules [4]. TM are of major importance when translating a document since they provide a reliable human translation. Thus, special considerations must be taken into account. First, it is important to keep information regarding the origin of the proposed translation (MT, TM, author, creation date, extra comments, etc). This is used by the translators to measure the reliability of the proposed translation. Second, splitting documents into segments is crucial for the success of the TM since, to find high matchings, the documents must be split in the same fashion. The general rule to apply is to split after end-of-sentence periods and new lines.

Other concerns are:

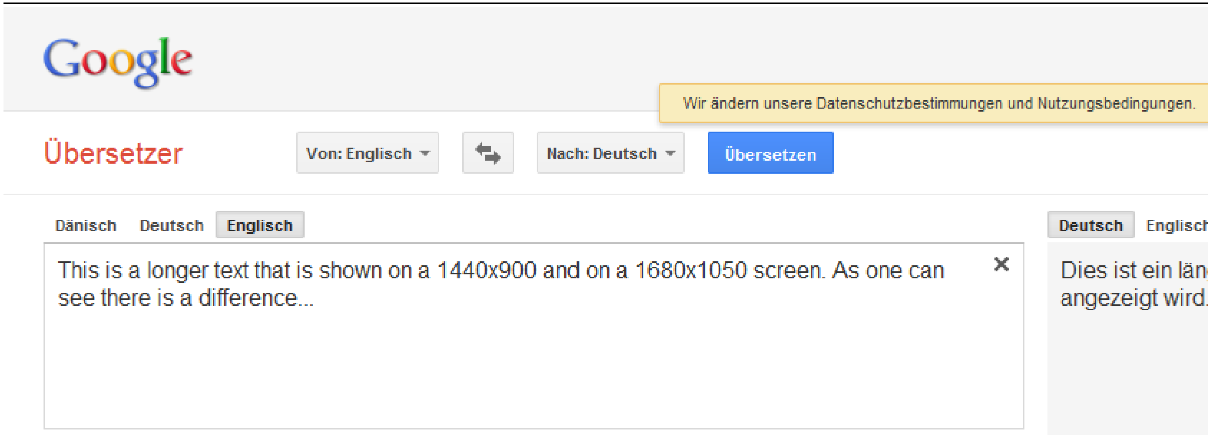
- how formatting tags, glossary terms and entities are to be handled (if they are to be handled at all).
- the UI must be as simple as possible to keep the focus on the real work.

6.4 Concerns regarding eye tracking

In the process of eye tracking the data provided by the eye tracker (gaze sample points) are used to identify ui componentes, words or letters the translator locked at. Those sample points are provided as x/y coordinates in pixels where the origin is the upper left corner of the screen. The coordinates are then adjusted so that their new origin is the one of the browser's client area. After that the coordinates are used to find the underlying component by a special function (inside the browser extension, see Figure 6) which searches for a matching component, letter, etc. at those coordinates in the browser window. All of this will be logged to be able to replay it later.

To have a proper replay now, one can imagine that the layout of the ui needs to be exactly the same as it was when the logging has been done.

If, for instance, a component used a relative size (like 100%) it will show up different (shrink or enlarge) when the browser's window size or screen resolution changes. Figure 13 shows an example with Google Translate where the screen resolution was set to 1440x900. When compared to Figure 14 (with a resolution of 1680x1050) one can easily see the problem: The text inside the source and target text areas is displayed differently. Both images have the same size but on the first one there are five words on the second line ('see there is a difference') where on the second one there is only one ('difference'). And even though that is not the aspect ratio mentioned above the problem gets visible and will also persist for an aspect ratio of 16:9.

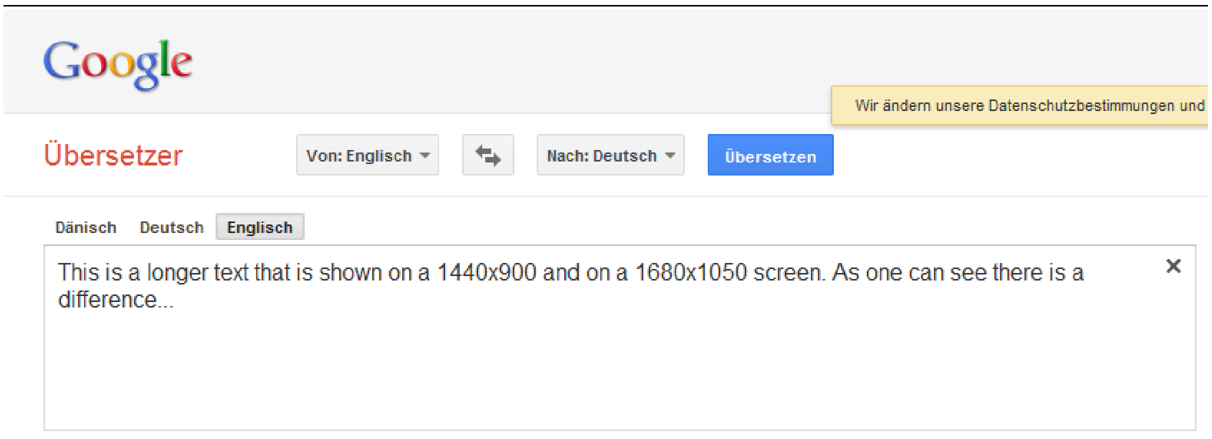


This is a longer text that is shown on a 1440x900 and on a 1680x1050 screen. As one can see there is a difference...

Dies ist ein län
angezeigt wird.

Figure 13: Example of a text area with a relative size at a resolution of 1440x900.

So the replay will not work correctly when the ui is rendered differently on log and on replay. The x/y coordinates get out of sync with the ui, in some cases they may still work but in many other they indicate a wrong word or component that has been looked at. That is a crucial problem and needs to be avoided.



This is a longer text that is shown on a 1440x900 and on a 1680x1050 screen. As one can see there is a difference...

Figure 14: Example of a text area with a relative size at a resolution of 1680x1050.

Bibliography

References

[1] V. Alabau, D. Ortiz-Martnez, V. Romero, and J. Ocampo. A multimodal predictive-interactive application for computer assisted transcription and translation. In *ICMI-MLMI '09: Proceedings of the 2009 international conference on Multimodal interfaces*, pages 227–228. ACM, 2009.

[2] R. Koebler. Rpc / json-rpc. `http://www.simple-is-better.org/rpc/`.

[3] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 177–180, Prague, Czech Republic, June 2007.

[4] P. Koehn and J. Senellart. Convergence of translation memory and statistical machine translation. In *AMTA Workshop on MT Research and the Translation Industry*, 2010.

[5] F. Marinescu and S. Tilkov. Json vs xml debate. `http://www.infoq.com/news/2006/12/json-vs-xml-debate`.

[6] Murray. Abi stability of c++ libraries. `http://www.murrayc.com/blog/permalink/2007/03/12/abi-stability-of-c-libraries/`.

[7] D. Ortiz, I. Garca-Varea, and F. Casacuberta. Thot: a toolkit to train phrase-based statistical translation models. In *Proceedings of the MT-Summit*, pages 141–148. Asia-Pacific Association for Machine Translation, Phuket, Thailand, September 2005.

[8] D. Ortiz-Martnez, L. A. Leiva, V. Alabau, and F. Casacuberta. Interactive machine translation using a web-based architecture. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 423–425, 2010.

[9] Wikipedia. Bridge pattern. `http://en.wikipedia.org/wiki/Bridge_pattern`.

[10] Wikipedia. Opaque pointer. `http://en.wikipedia.org/wiki/Opaque_pointer`.

A Alternative System Architecture Proposals

During the elaboration of this deliverable, different system architecture proposals were evaluated. Such architecture proposals tried to satisfy the special needs manifested by the different project partners (see section 4). In the following sections we give the details of each architecture alternative, including a brief textual description and its corresponding architecture diagram.

A.1 Architecture Alternative 1

In Figure 15 the first proposal is shown. It seems to be useful to have the architecture divided into two main parts. Part 1 contains all that is needed for translation purposes only while part 2 is considered for UI, user input, logging and replay. Between those two parts there will be a generalized translation API as an extension to the Google Translation API as the interface.

This architecture focuses on those main aspects:

- Provide a generalized translation API as an extension to the Google Translate API.
- Decouple translation related aspects from UI related ones. This means to separate the system logically into two main parts:
  - A translation part
  - A user interface part
- Make translation part usable standalone. No need to have unwanted/unused UI functions.
- Have the UI free from any linguistic knowledge.
- Provide a single point of access for others to use for their translation UI (The Translation-Server in Figure 15). That is on the one hand to have a clear interface and on the other hand it is useful, because only one system needs to be exposed within a government/company (explicit configuration in firewall/DMZ).
- Have the logging and replay functionality separated from the translation part. That is, for instance, that the load for logging and replay will not affect the translation part.
- To have a correct logging all calls that need to be logged are done through the GUI-Server
- Push as much logic as possible to the server side. Client side (browser) programming (JavaScript) is most likely more difficult and error prone than server side implementation.
- Design the architecture as a distributed system. That makes it easier to extend and/or replace functionality and also improves scalability.

The functionality of most components of this architecture is already known: MT, IMT, TM, HTR, GUI and Browser. But this architecture introduces two components that need some more explanation: Translation-Server and Document-Management-Server.

Translation-Server This component couples all translation related parts through a single API. It merged and standardizes the different APIs into one single interface and generalizes data types. That means that it acts like a bridge between the different translation engines providing. It also offers possibility to add additional engines as needed by keeping the API unchanged to the outside world. The outside world can thus easily access the translation functionality through a clear interface.

Document-Management-Server The Document-Management-Server has mainly the function to split up documents into segments and to merge those segments back into one single document when all segments have been translated. This is a dedicated server as the segmentation could by where sophisticated for different languages.

Authentication and authorisation in this scenario need some discussion but could be done like suggested be UPVLC (see Google API).

A.2 Architecture Alternative 2

Figure 16 shows a second system architecture proposal. Such proposal still wants to supply an API for the outside world. But in this approach the translation API and UI API are mixed. Also there is no Translation-Server. That means that GUI- and Translation-Server have been merged into one single instance. Those modifications are made mainly to simply the system, especially authentication and authorisation and also eliminate the fact, that the Translation-Server will only be a router in some cases.

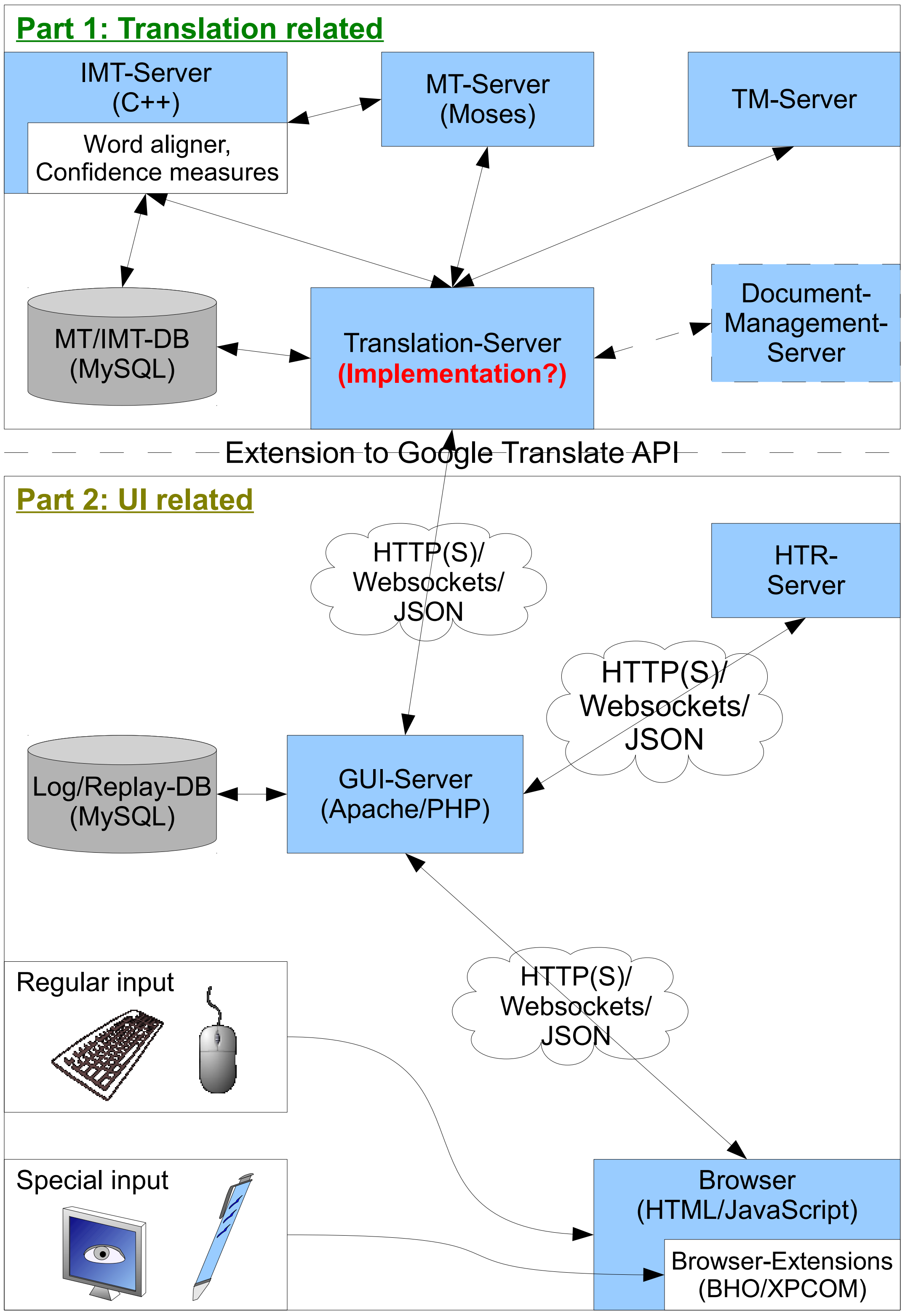


Figure 15: Proposal 1 of a system architecture.

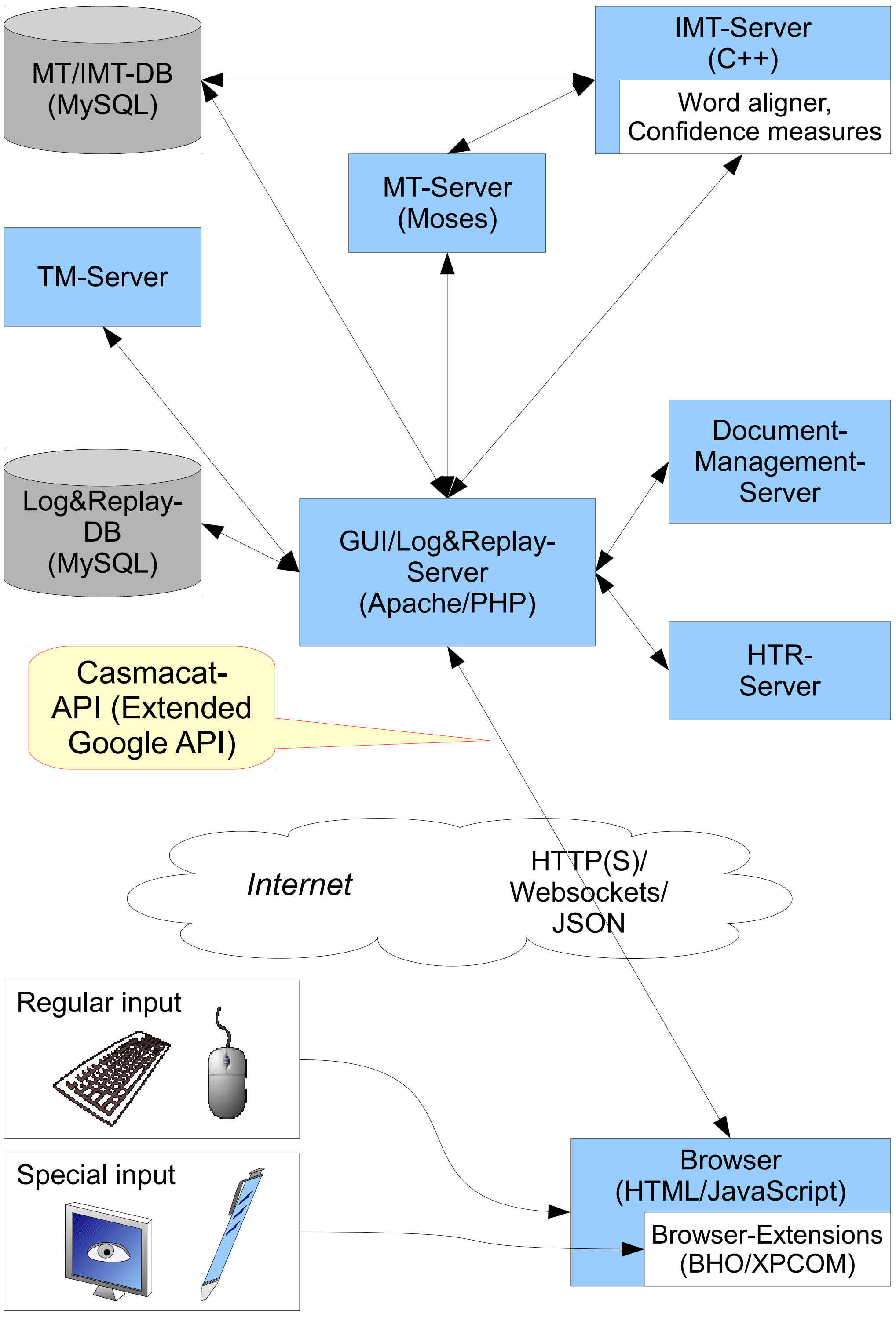

MT/IMT-DB (MySQL)
IMT-Server (C++)
Word aligner, Confidence measures
MT-Server (Moses)
TM-Server
Document-Management-Server
Log&Replay-DB (MySQL)
GUI/Log&Replay-Server (Apache/PHP)
HTR-Server
Casmacat-API (Extended Google API)
Internet
HTTP(S)/ Websockets/ JSON
Regular input
Special input
Browser (HTML/JavaScript)
Browser-Extensions (BHO/XPCOM)

Figure 16: Proposal 2 of a system architecture.

B Acronyms

API	application programming interface
DOM	document object model
GUI	graphical user interface
HTR	handwritten text recognition
IMT	interactive machine translation
MT	machine translation
MTU	maximum transmission unit
PHP	hypertext pre-processor
PIMPL	pointer to implementation
TE	translation environment
TS	translation server
UI	user interface