

DELIVERABLE

Project Acronym: FLAVIUS

Grant Agreement number: ICT-PSP-250528

Project Title: Foreign Language Versions of Internet and User generated Sites

D4.3 Translation Memory able to collect data

Revision: 1.5

Authors:

Christophe Brun-Franc (Softissimo)

Constantin Walter (Across)

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Revision History

Revision	Date	Author	Organization	Description
0.1	February, 14th	Constantin Walter	ACROSS	Draft version
0.2	February, 21th	Christophe Brun-Franc	Softissimo	Revised version
1.0	March, 1st	Christophe Brun-Franc	Softissimo	Version sent to EC.
1.5	Sept, 15th	Constantin Walter	ACROSS	Revised version, included appendices

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Sommaire

1. Introduction	4
2. The Across Language Server	5
3. The Translation Memory crossTank	6
4. General API communication with Across Server	12
5. Modifications and Extensions in crossAPI	14
6. Implementation of FLAVIUS platform with Across Translation Memory	18
Appendix A (Advanced Code Samples).....	21
Appendix B (QA by Unit Tests).....	25

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

1. Introduction

FLAVIUS, Foreign LAnguage Versions of Internet and User-generated Sites, aims at bridging the language gap between content publishers and users by providing an online platform accessible to websites owners that will enable them to generate multilingual versions of their site, quickly, easily and efficiently in as many languages as they want.

FAVIUS is a European project under the Competitiveness and Innovation framework Programme (call identifier: CIP-ICT-PSP-2009-3)

Objective of the deliverable

The objective of this deliverable is to describe the translation memory used in FLAVIUS and its implementation

This deliverable contains an overview of the ACROSS translation memory and a description of the implementation of this TM in FLAVIUS.

This report is deliverable D4.3 Translation Memory able to collect data.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

2. The Across Language Server

The Across Language Server is an integrated solution for Corporate Translation Management (CTM) and provides the full set of features and components needed for translation support as such as well as for project and workflow management. As a central platform for corporate language resources and translation processes, it includes a translation memory – called crossTank – and a terminology system – called crossTerm.

The Across Language Server is a relational database-supported system, based on a client/server architecture. The use of a database enables central storage of all data, including the source and target texts and project-related and personal data of translators and customers. The database used by the Across Language Server is the Microsoft SQL Server. With the client/server architecture the clients/users working with Across have access to the same server/database and work with a common dataset.

Technically speaking the Across Language Server runs three different servers using each a separate database instance within the SQL server:

- the Across Server, responsible for the administration of system data, task management (translations, etc.) and workflows
- the crossTank Server, responsible for storing and providing translation memory entries
- the crossTerm Server, responsible for terminology administration

In scope of FLAVIUS the components of subject are the system administration component of the main Across Server and the crossTank Server as Translation Memory.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

3. The Translation Memory crossTank

The Across translation memory is called crossTank. The crossTank Server manages a database of legacy texts, subdivided into translation units i.e. pairs of source and target segments (see crossTank Manager).

The crossTank works like common translation memories do: if crossTank finds an identical or similar sentence in a new source text, it makes the translation available to the translator.

The crossTank Manager

The crossTank Manager serves the management and maintenance of translation units stored in the translation memory crossTank.

The crossTank Manger allows such operations as adding manually new translation units and searching, editing, deleting or merging existing translation units as well as the import, export and the maintenance of translation units.

Segmentation

The segmentation of the source and target texts is fundamental to the functioning of a translation memory system because it is this that allows the system to find a previously translated sentence or paragraph in the translation memory, to avoid having to translate it again.

In Across, there are two different segmentation modes and by this means also two different storing modes: segmentation (and storing) by sentences and segmentation (and storing) by paragraphs.

Segmentation by sentences is the default segmentation type enabled for all document formats in Across. This method first divides the source texts into paragraphs and subsequently into sentences with the help of predefined sentence rules.

In paragraph segmentation, the source text is simply divided up into paragraphs. (This can be useful, for example, to avoid sentences being inserted in the translation out of context, e.g. in the case of a pre-translation.)

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Attribution of translation units

Unlike file-based translation memories crossTank as a database-supported translation memory stores all translation units in one database. To enable the retrieval and use of the needed data, along with the language data, crossTank stores various system attributes. By default, crossTank uses system attributes for project, relation/customer, and subject. Moreover, additional user-defined system attributes can be created. Detailed filter settings enables the translator to narrow down his search and find the data he is looking for.

Additionally crossTank logs extensive information about the use of translation units stored in crossTank. For example, the history records how often each translation unit was used, i.e. how often it was inserted in a translation (e.g. during pre-translation or manually by a translator). Moreover, the history records when and by which translator a translation unit was last used.

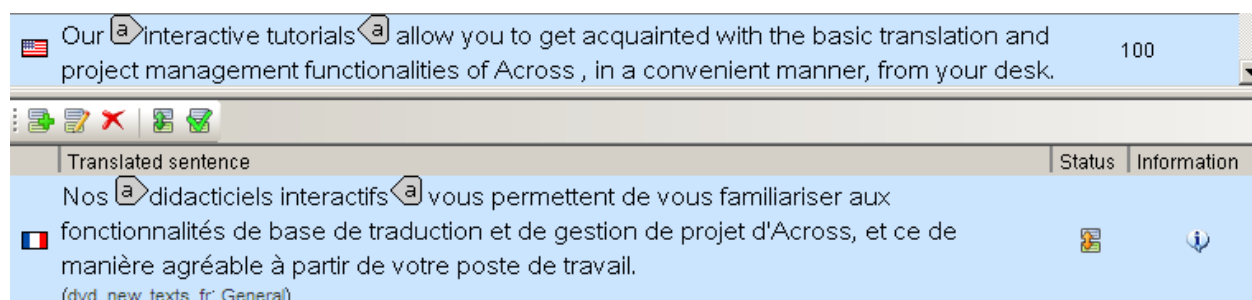
The status of translation units

In crossTank every translation unit has a status indicating the creation mode and reliability of the translation units. The following states are used:

- Newly inserted: The entry was added manually in crossTank (by inserting it manually in the crossTank Manager).
- Smartly inserted: The entry was inserted (semi-)automatically (without user interaction) to crossTank (e.g. by translating a segment in the translation editor crossDesk).
- Aligned: The entry was inserted within the scope of an alignment.
- Released: The entry was released by an authorized person.

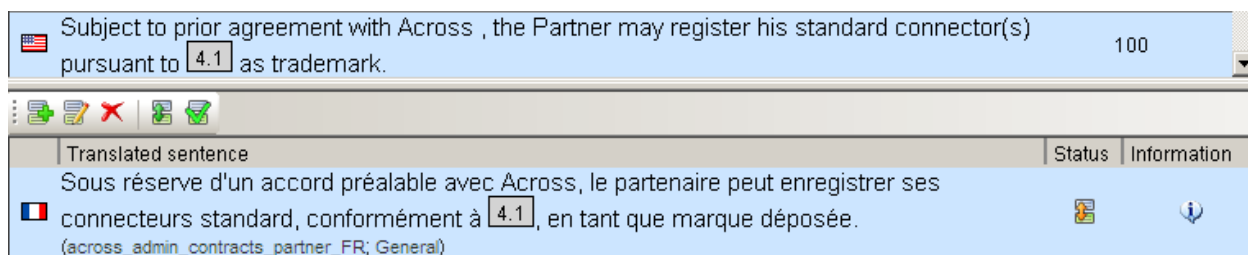
Storing of rich translation contents (tags etc.)

The crossTank optionally stores rich translation contents, i.e. tags, styles, and other characters (control characters, special characters, and white spaces).



The screenshot shows the crossTank interface. At the top, there is a header bar with a US flag icon, the text "Our interactive tutorials allow you to get acquainted with the basic translation and project management functionalities of Across , in a convenient manner, from your desk.", and a count of "100". Below this is a toolbar with icons for insert, delete, and other functions. The main area displays a translation unit with a "Translated sentence" column and "Status" and "Information" columns. The translated sentence is: "Nos didacticiels interactifs vous permettent de vous familiariser aux fonctionnalités de base de traduction et de gestion de projet d'Across, et ce de manière agréable à partir de votre poste de travail." The status column shows a small icon, and the information column shows a small icon. At the bottom of the unit, it says "(dvd_new_texts_fr; General)".

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services



This saving mode of crossTank is called Rich TM and is activated by default.

Auto-Adjustments

During pre-translation or when identifying and inserting translation units in the translation, Across tries to adjust the translation units with respect to tags and formatting.

If, for example, the only difference between a segment to be translated and a translation unit stored in crossTank is a different tag, the auto-adjustment function enables Across to identify the translation unit as a 100% match. Without the auto-adjustment, the crossTank entry would merely be a fuzzy match because of the deduction of a penalty.

Duplicates

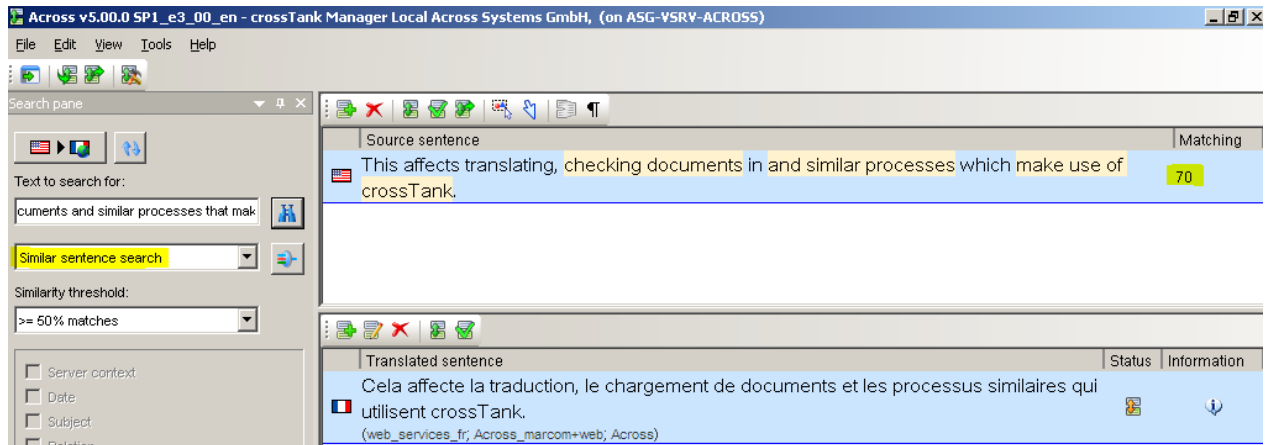
The multiple storage of translations can result in unwanted redundancies in crossTank, referred to as "duplicates". For example, this may happen when a sentence already translated and stored in crossTank is edited and stored again in crossTank. It is possible to prevent the storage of duplicates in crossTank, so that there will only be one translation for every source sentence.

However, multiple storage may be desirable, e.g. if there are two different translations for a sentence that are selected depending on the context. Therefore the translator can store deliberately a duplicate if needed.

Fuzzy-Search

The fuzzy search displays translation units that are similar to the current segment. The relative match rate is displayed for each suggested translation. The match rate is calculated by the similarity between the current segment and the translation unit in crossTank. Additionally penalties can be deducted due to differences e.g. in included tags. A minimum match rate for suggested translation to be shown can be defined in the crossTank settings. Filters can be set in order to display only translation units e.g. of a certain subject and/or of a certain translation unit status.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services



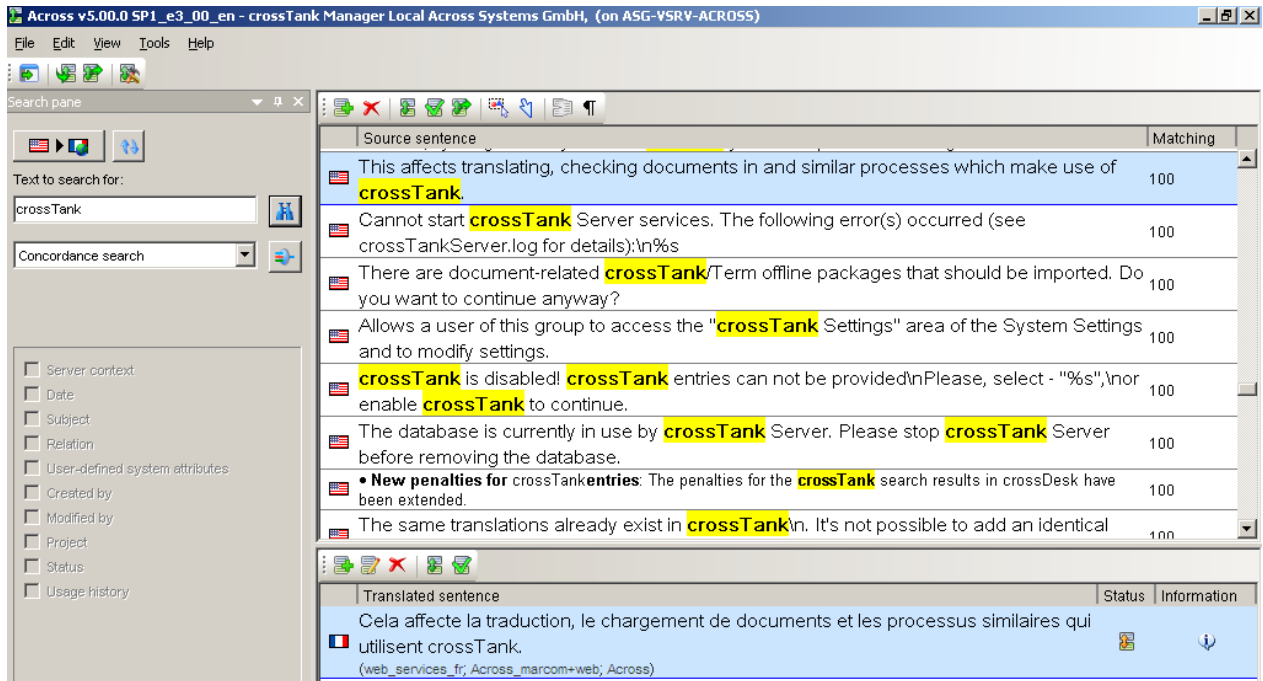
FLAVIUS platforms will only make use of the high-fuzzy-rate matches (100%). Otherwise translations will be applied using the machine translation engine by Language Weaver.

It may be subject for discussion if exclusively 100% matches are used or if a e.g. 98% match contains already better quality than the machine-generated translation.

Concordance Search

The concordance search allows to quickly look for a word or several words in crossTank. This allows to see at a glance the context a word has been used in the past and how others or the translator him- or herself has translated it previously. The concordance search is available in the crossTank Manager and in FLAVIUS context primarily used for administrative checks.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

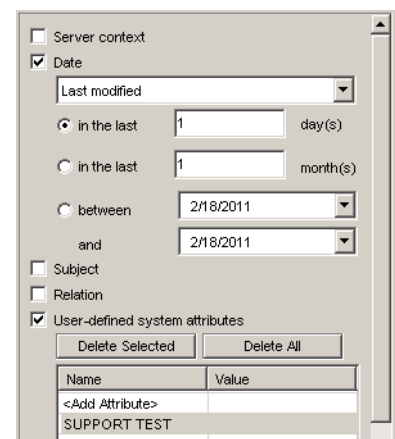


Concordance search options may be used during implementation to quick check population of crossTank. If it is tried to perform this check using fuzzy search whole search phrases are needed. Otherwise no result would be displayed because all potential return sets are below the minimum fuzzy threshold of 50%.

Search filters

Various search filters are available for refining searches in crossTank. The following filters can be used:

- Date
- Subject
- Relation/customer
- User-defined attributes
- Creator
- Modifier
- Project
- State
- Usage History (usage count, date of last usage, last user)



Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Penalties

To prevent unsuitable translation units, Across uses percentages (penalties) that are deducted from the match rate due to differences, e.g. in the formatting or system attributes (relation/customer, subject, etc.).

The following penalties can be used in Across:

- Different punctuation
- crossTank entries with the "Released" status.
- crossTank entries with the "Aligned" status.
- crossTank entries with the "Smartly inserted" status
- crossTank entries with the "Newly inserted" status
- Different other characters
- Different tags/inline objects
- Different formatting
- Wrong user-defined attribute
- Different relation
- Different project
- Different subject
- Switched words
- Wrong sublanguage

Penalties

Apply penalties for missing system attribute values

Penalties for (in %)

2	-	+	different sublanguage
10	-	+	switched words
10	-	+	different subject
0	-	+	different project
20	-	+	different relation
0	-	+	different user-defined attributes
1	-	+	different formatting
1	-	+	different inline elements
1	-	+	different special characters
0	-	+	state newly inserted
0	-	+	state smartly inserted
0	-	+	state aligned
0	-	+	state released
1	-	+	different punctuation

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

4. General API communication with Across Server

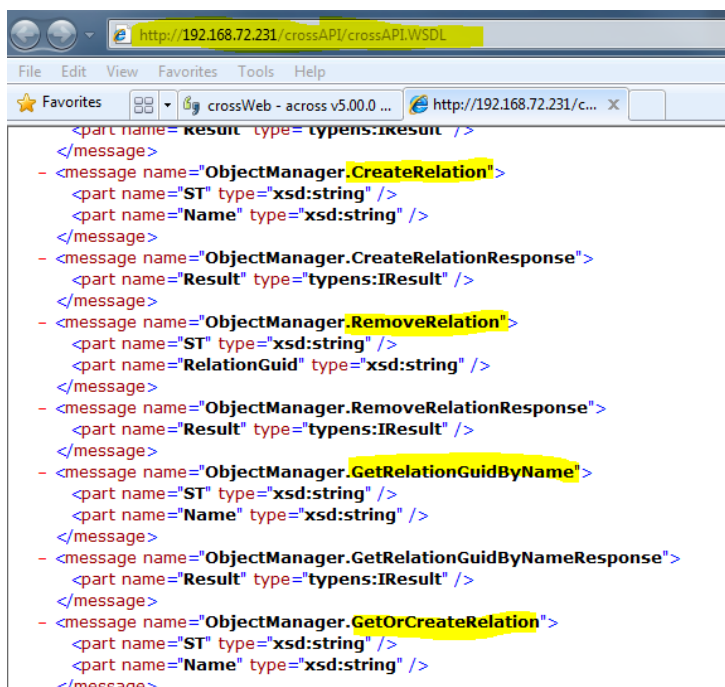
Across consists now of various programming interfaces for other applications (so called APIs).

crossAPI System Integration (SI)

This subset of the API is used for managing translation jobs or configurations inside Across.

crossAPI SI works more on the high level. Other systems may address for creating entire translation projects to be assigned to other Across systems.

For the FLAVIUS use case, crossAPI SI can be used to create new attribute sets like subjects or relation to separate the TM for content of different sites.



The crossAPI SI runs as a web service (SOAP) on the Across application server.

It will be addressed by the FLAVIUS platform via http.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

crossAPI Interactive (IA)

The crossAPI IA is designed for more granular functionality requests from the Across Server. I.e. other systems may ask for the best translations of a given text segment.

It is the major interface for integration of FLAVIUS platform with the Across Language Server environment.

The following chapter describes the necessary steps and changes in detail.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

5. Modifications and Extensions in crossAPI

Separation of API and crossTank core

Before setting up the Across Server environment the first time on FLAVIUS platform ACROSS had to work on the ability to separate API and application server.

ACROSS solved this by splitting up the program components. With this several tasks were related.

Communication issue

Instead of a direct communication between API and core application we had to establish communication of crossAPI Interactive component with crossTank core by TCP/IP.

- Extension of crossAPI IA with TCP connectivity
- Extension of crossTank core to allow direct TCP socket connections from API
- Introduction of dedicated logging
- Testing and QM

Deployment issue

The Across Language Server setup procedures are designed to deploy all components (except database server) on a single machine. The installation procedure had to be extended.

- Creation of separate MSIs
- Changing setup wizard to support multiple MSIs in this case
- Introducing support of configuration files during setup (using generic softkey files)
- Testing and QM

Implementation of additional methods

In detail several objects or methods needed enhancement.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Improving IcTaSearcher method set

For Flavius it is vital to be able to distinguish between different customer TMs and also to manage non released entries or test imports.

Responsible for this function set is the IcTaSearcher interface. It contains the structures for supporting this functionality.

Changes in this interface affect all functions contained. These are:

- AdjustTranslation
- SearchConcordance
- SearchConcordanceEx
- SearchConcordanceEx2
- SearchConcordanceRich
- SearchTranslations
- SearchTranslationsByID
- SearchTranslationsByIDRich
- SearchTranslationsEx
- SearchTranslationsEx2
- SearchTranslationsRich
- SearchTranslationsRichEx
- SearchTranslationsRichEx

In Flavius context only the SearchTranslation* functions are of interest, but it is not possible to limit the extension on them (and exclude the concordance search functions).

From the implementation point of view fuzzy search is an extension of concordance search. Concordance search provides a result set where the words from query string are present. Fuzzy search performs quite the same in it's first step. But furthermore it checks also for sequence of words and calculates the similarity value.

Internally a concordance search query makes use of the same SQL Stored Procedures as the Fuzzy Search. So the parameters for filtering are also available in concordance search. In fact it would have been additional effort to limit the filter functionality on Fuzzy Search.

But anyway - Concordance Search via crossAPI could be also useful in future if Flavius platform incorporates additional functionality for TM management.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Added function UpdateTranslations.editor

The interface IcTaEditor did not contain an option to update existing translations in a convenient way (of course it would be possible to delete the old one and add the updated translation new, but this is not convenient enough).

For this we introduced the function UpdateTranslations and UpdateTranslationsRich (for tagged content).

Improved interface IcTaFactory

Now IcTaFactory has function GetTranslationMemoryInfo. This functions returns an object that contains statistical information about population of the translation memory.

Flavius platform makes use of this by monitoring state and population of customer TMs.

Added Across.ISDD

This interface provides functions that are used for getting and applying sentence delimitation detection on whole phrases.

One major issue in querying translations from the translation memory is using the same segmentation rules in queries as the translation units are stored in TM. Across already offers segmentation rules for processing documents to be translated, but it is also necessary to provide this service to a service requester via crossAPI to have a consistent rule set for sentence detection inside Across TM and Flavius platform integration.

Prototype application for demonstration and testing

To demonstrate and assist the FLAVIUS platform developers in finding the right integration points with the Across Language Servers APIs ACROSS developed and provided a sample application how communication with ACROSS TM could happen.

This additional application has several purposes:

- Emphasizing the functions needed for integration
- Supporting implementation by providing sample illustration
- Support in testing general connectivity
- Tool for validating results as collected by the FLAVIUS platform

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

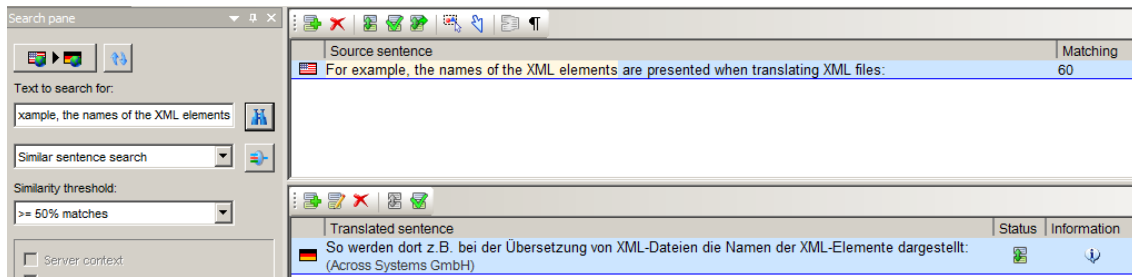
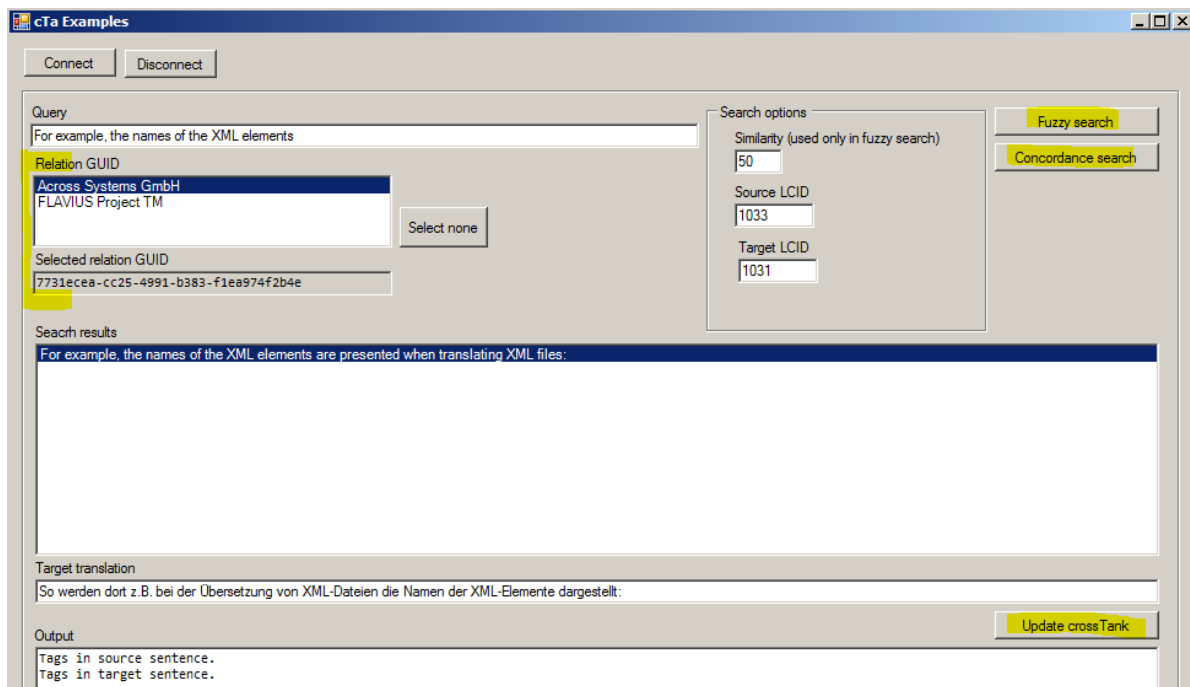


Illustration of sample sentence in crossTank

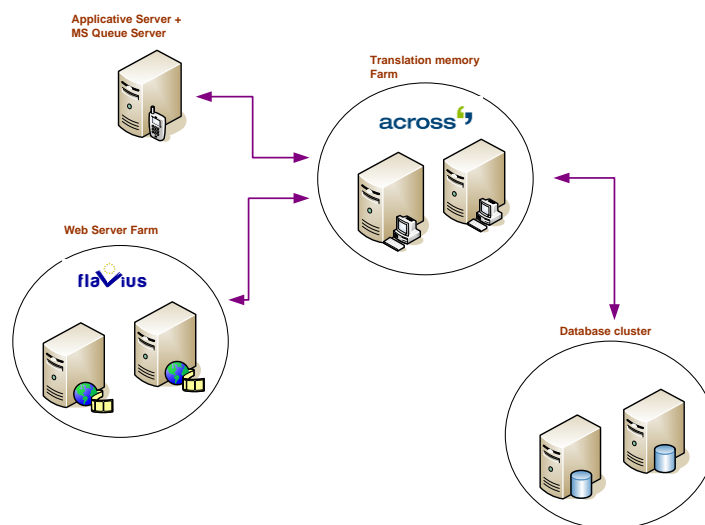


Same sentence in test application

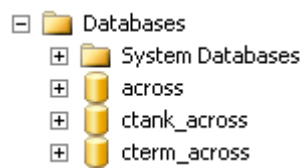
6. Implementation of FLAVIUS platform with Across Translation Memory

Technical architecture

Across Language Server with the Translation Memory module (crossTank) is installed on a physical server (distinct from the SQL Database Server).



The database server - SQL Server 2008 - is configured with the three database required for Across Language Server (Across, ctank_Across, cterm_Across).



The ctank_Across data is the repository for the translation memory.

A registered user FLAVIUS was created. This account will be the one used to access the Translation Memory through the CROSS API Interactive.

A relation attribute - called Owner - is used to link TM entries to a specific FLAVIUS web account.

Use of CROSS API Interactive

FLAVIUS is communicating with Translation memory through the crossAPI Interactive (see chapter 4). This API is integrated in the FLAVIUS platform as an external reference (interface) and called through C#.

```
public crossClient(string installation_guid, string username, string
password)
{
    this.installation_guid = installation_guid;
    this.username = username;
    this.password = password;
}
```

This API allows not coping with the physical location of the FLAVIUS web application and Translation Memory. The remote communication is handled by the crossAPI interactive and is not handled by the FLAVIUS web application.

Thus, the TM part of the FLAVIUS web application is more focused on the functional aspects.

The crossAPI interactive is used in FLAVIUS for:

- Translating text using the TM
- Enriching the TM

Translating text using the Translation Memory

The translation process is performed in the FLAVIUS Translation module.

The module is called for each file with the list of segments extracted by the parser module.

The translation module calls the translation memory through the SearchTranslation function. For each segment, the module check if there is a match.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

According to the result:

- the translation is kept and no call will be done to the SMT engine
- the module calls the SMT engine to get an automatic translation

Once the entire file is translated, the module generates the new list of translated segments. A new version of the file will be generated from the list of translated segments and saved in the translated directory of the job.

Enriching the TM (through post-edition)

The Translation Memory is enriched with validated sentence through the post-edition interface.

During the post edition phase, the edited segments will be flagged as ready for an insertion in the translation memory.

At the end of the post edition process, the flagged segments will be extracted from the job data, and will be inserted in the translation memory through the UpdateTranslations function of the API.

The FLAVIUS platform adds several metadata to the segment unit - which is composed of the source text and the translation -. These metadata are:

- Owner
- Date of creation
- Author
- Status : Smartly inserted
- Project : the job name
- File : the file name
- Subject

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Appendix A (Advanced Code Samples)

Code samples for frequently used functions

Function: SearchTranslations

Carries out a similarity search in crossTank.

Syntax

```
TRANSLATION[] SearchTranslations(int SessionID, int Similarity, SENTENCEINFO Pattern, int TargetLcid, FILTER SearchFilter);
```

Parameters

SessionID: The ID of a crossTank session.

Similarity: The minimal similarity in percent.

Pattern: The search pattern.

TargetLcid: The LCID of the target language.

SearchFilter: The filter that should be used for the search.

Return value

An array of the translations found.

Example of usage:

```
public TRANSLATION[] FuzzySearch(string query, int similarity, string
relation, int source_lcid, int target_lcid)
{
    TRANSLATION[] translations = null;
    sessions = cross_tank.GetSessions();
    int session_id = sessions.CreateSession(this.instalation_guid);

    IcTaSearcher searcher = cross_tank.GetSearcher();
    SENTENCEINFO sentence_info = new SENTENCEINFO();
    sentence_info.Lcid = source_lcid;
    sentence_info.Sentence = query;

    FILTER filter = new FILTER();
    filter.ResultLimit = -1; //Get all results
    filter.TimeFilter = TimeFilterMode.TimeFilterUnknown;
    filter.ServerContextGuids = servers.ToArray();

    translations = (TRANSLATION[])searcher.SearchTranslations(session_id,
similarity, sentence_info, target_lcid, filter);
    sessions.CloseSession(session_id);
    return translations;
}
```

Function: UpdateTranslations

Edits translations in crossTank.

Syntax

```
void UpdateTranslations(int SessionID, ref TRANSLATION[] Results);
```

Parameters

SessionID: The ID of a crossTank session.

Results

Contains all information necessary to edit translations. After the update is completed it also contains the automatically generated information such as ID.

Example of usage:

```
public bool UpdateTranslation(TRANSLATION translation)
{
    try
    {
        int session_id = sessions.CreateSession(this.instalation_guid);
        IcTaEditor editor = cross_tank.GetEditor();

        System.Array translations = new TRANSLATION[] { translation };
        editor.UpdateTranslations(session_id, ref translations);
    }
    catch
    {
    }
    return true;
}
```

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Function: GetTranslationMemoryInfo

Returns statistical information about crossTank.

Syntax

TMINFO[] GetTranslationMemoryInfo(int SessionID, **FILTER** SearchFilter);

Parameters

SessionID: The ID of a crossTank session.

SearchFilter: A filter that specifies which information should be entered in the statistics.

Return value

An array containing the statistical information about crossTank.

Example of usage:

```
public int GetTranslationsCount()
{
    sessions = cross_tank.GetSessions();
    int session_id = sessions.CreateSession(this.instalation_guid);

    FILTER filter = new FILTER();

    ICtaTMInfo cTaTMInfo = cross_tank.GetTranslationMemoryInfo();
    var info = cTaTMInfo.GetTranslationMemoryInfo(session_id, filter);

    sessions.CloseSession(session_id);

    return ((TMINFO)inf.GetValue(0)).TranslationCount;
}
```

Interface: ISDD

This interface provides access to the SDD functionality.

Function: DetectAlphaWords

Used to retrieve the ranges of alphanumerical words in the supplied text.

Syntax

[ISddRangeList](#) DetectAlphaWords(string text);

Parameters

text: The text where the words should be detected.

Return value

A [ISddRangeList](#) object. Contains one range per detected alpha word.

Example of usage:

```
public int GetAlphaWordsCount(string text)
{
    client = new CrossClientClass();
    client.Open(this.instalation_guid, this.username, this.password);

    sdd = client.GetSDDInterface("");

    var rangeList = sdd.DetectAlphaWords(text);

    return rangeList.GetCount();
}
```

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Appendix B (QA by Unit Tests)

Establish connection test.

Tests if the connection to the crossAPI IA could be successfully established and the session could be opened, using correct installation GUID, user name and password.

```
var connection = new Across.Client.Connection();
connection.Open(InstallationGuid);
Assert.IsTrue(connection.IsOpened);
var session = connection.CreateSession(Login, Password);
Assert.IsNotNull(session);
```

Result: Passed

Reject connection test

Tests if the connection function throws correct exception (Across.Client.AuthorizationException), if the invalid credentials are used for establishing connection to crossAPI IA.

```
[TestMethod]
[ExpectedException(typeof(AuthorizationException))]
public void InvalidCredentialsTestConnection()
{
    //Establishing connection with wrong credentials
}
```

Result: Passed (The generated exception is of the [ExpectedException] type)

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Add simple translation test

Tests if the simple text is added correctly to TM and can be then found with the 100% similarity search.

```
[TestMethod]
public void TestAddTranslations()
{
    //Insert some test translation in TM
    var resultTranslations = FindExactly(TestSourceText,
    SourceLanguageLCID, TargetLanguageLCID);

    // Check if the translation was added correctly
    Assert.IsNotNull(resultTranslations);
    Assert.IsTrue(resultTranslations.Count() == 1 );
    Assert.AreEqual(TestSourceText,
    resultTranslations.First().Source.Text);
    Assert.AreEqual(TestTargetText,
    resultTranslations.First().Target.Text);
}
```

Result: Passed. The translation that was found after insertion has the same source- and target-text.

Special characters insertion test

Tests if the sentences that contain special characters could be correctly inserted to the TM (could be then found with the same special characters).

```
[TestMethod]
public void TestAddTranslationWithSpecialCharacters()
{
    //Insert some text with special characters.

    var resultTranslations =
    FindExactly(TestTextWithSpecialCharacters,
    SourceLanguageLCID, TargetLanguageLCID);

    // Check if the translation was found correctly
    Assert.IsNotNull(resultTranslations);
```

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

```

Assert.IsTrue(resultTranslations.Count() == 1 );
Assert.AreEqual(TestTextWithSpecialCharacters,
resultTranslations.First().Source.Text);
Assert.AreEqual(TestTextWithSpecialCharacters,
resultTranslations.First().Target.Text);
}

```

Result: Passed. The test string "€£¥@™±≠≤÷×∞μαβπΩΣ" was successfully found after insertion

Number differences search test

Tests if the sentence could be correctly found if the search pattern has the difference only in some number.

```

[TestMethod]
public void TestSearchOnlyNumberDifferences()
{
    //Insert some text with number.

    //Search for a sentence that has only number-difference
    var resultTranslations = FindSimilar(TestNumber2Text,
SourceLanguageLCID, TargetLanguageLCID, 99);

    Assert.IsNotNull(resultTranslations);
    Assert.IsTrue(resultTranslations.Count() == 1 );
    Assert.AreEqual(TestNumber1Text,
resultTranslations.First().Source.Text);
    Assert.AreEqual(TestNumber1Text,
resultTranslations.First().Target.Text);
}

```

Result: Passed. The added sentence was "This is the number 1 example", the search pattern was "This is the number 2 example". The source sentence was found.

Remarks: Difference in numbers results in a penalty (1% penalty per one number). That is why user must use the search with similarity less than 100% and filter the result set for the 100% matches that were created by ACROSS auto adjustment of numbers.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Big test insertion test

Test if big text blocks can be successfully added to the TM. The translations that bigger than 255 characters, has a type Ntext in the database. The maximal size is 2.147.483.647 Byte.

```
[TestMethod]
public void TestBigTextInsertion()
{
    //Insert some big text.

    var resultTranslations = FindExactly(TestBigText,
    SourceLanguageLCID, TargetLanguageLCID);

    // Check if the translation was found correctly
    Assert.IsNotNull(resultTranslations);
    Assert.IsTrue(resultTranslations.Count() == 1 );
    Assert.AreEqual(TestBigText,
    resultTranslations.First().Source.Text);
    Assert.AreEqual(TestBigText,
    resultTranslations.First().Target.Text);
}
```

Result: Passed. The test text contains 20007 characters. The text was successfully added and then found. The 100% match search execution time on the test PC was 0.0041 milliseconds. Insertion operation time – 7.12 seconds.

Update simple translation test

Tests if the translation could be successfully updated.

```
[TestMethod]
public void TestUpdateTranslation()
{
    InsertExampleTranslation();

    var resultTranslations = FindExactly(TestSourceText,
    SourceLanguageLCID, TargetLanguageLCID);

    // Update found example translation(-s)
```

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

```

foreach (var resultTranslation in resultTranslations)
{
    resultTranslation.Target.Text = TestUpdatedText;
}
resultTranslations = FindExactly(TestSourceText,
SourceLanguageLCID, TargetLanguageLCID);

// Check if the translation was updated correctly
foreach (var resultTranslation in resultTranslations)
{
    resultTranslation.Target.Text = TestUpdatedText;
    Assert.AreEqual(resultTranslation.Target.Text,
    TestUpdatedText);
}
}

```

Result: Passed. The target text of the translation, that was found, was correctly updated.

Delete translation test

Tests if the deletion operation works correct.

```

[TestMethod]
public void TestDeleteTranslation()
{
    InsertExampleTranslation();

    var resultTranslations = FindExactly(TestSourceText,
SourceLanguageLCID, TargetLanguageLCID);

    // Check if the translation(-s) was added to the TM
    Assert.IsTrue(resultTranslations.Count() > 0);

    // Delete translations from the TM
    foreach (var resultTranslation in resultTranslations)
    {
        resultTranslation.Delete();
    }
}

```

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

```

// Search the deleted translations in the TM and check
that //they are not exist.
resultTranslations = FindExactly(TestSourceText,
SourceLanguageLCID, TargetLanguageLCID);
Assert.IsTrue(resultTranslations.Count() == 0);
}

```

Result: Passed. There was found no translation after delete operation.

ImportTMX test

Tests if a tmx file is successfully imported in the ACROSS TM.

```

[TestMethod]
public void TestImportTmx()
{
    // First delete this example if it exists in the TM
    DeleteExampleTranslation();

    ImportTMX(tmxFile);

    var resultTranslations = FindExactly(TestSourceText,
SourceLanguageLCID, TargetLanguageLCID);

    // Check if the translations was added correctly in TM
    Assert.IsNotNull(resultTranslations);
    Assert.IsTrue(resultTranslations.Count() == 1);
    Assert.AreEqual(TestSourceText,
foundedTranslations.First().Source.Text);
    Assert.AreEqual(TestTargetText,
resultTranslations.First().Target.Text);
}

```

Result: Passed. The test translation was successfully found after importing test tmx file.

Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Import corrupted .tmx file test

Tests if the API aborts correctly if the TMX to be imported is corrupted.

```
[TestMethod]
public void TestImportCorruptedTmx()
{
    //Import TMX file, that has corrupted structure
    ImportTMX(corruptedFile);

    // Check if an error has occurs during importing
    corrupted TMX // file
    Assert.IsTrue(crossApiProxy.TankManager.GetJobStatus(result) == 2);
}
```

Result: Passed. This test waited for a error code (2) from the API. The code = 2 was returned.

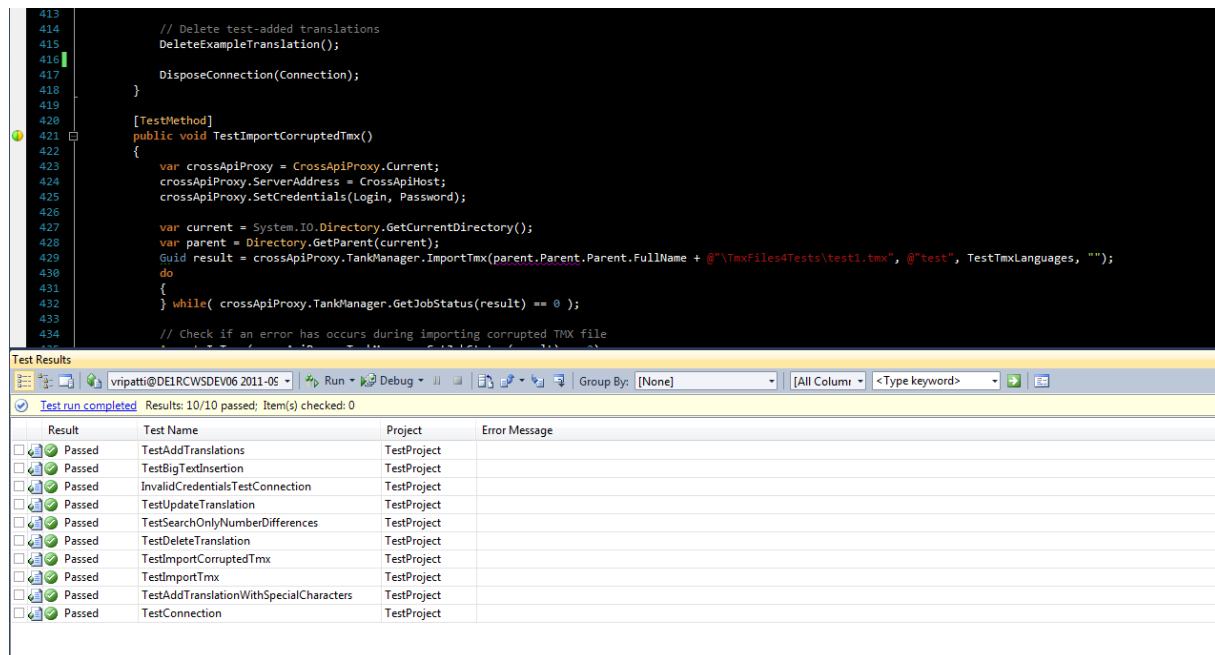
Project co-funded by the European Commission within the ICT Policy Support Programme	
Dissemination Level	
C	Confidential, only for members of the consortium and the Commission Services

Final Remarks

All these tests are written using Visual Studio S2010 Unit Testing framework, so the developer should not install some additional software.

The described implementation of unit tests runs inside the ACROSS development system when a new built is created. These tests are also implemented inside FLAVIUS platform to verify the appropriate functionality there.

Passing the unit test results inside Visual Studio:



```

413
414 // Delete test-added translations
415 DeleteExampleTranslation();
416
417 DisposeConnection(Connection);
418 }
419
420 [TestMethod]
421 public void TestImportCorruptedTmx()
422 {
423     var crossApiProxy = CrossApiProxy.Current;
424     crossApiProxy.ServerAddress = CrossApiHost;
425     crossApiProxy.SetCredentials(Login, Password);
426
427     var current = System.IO.Directory.GetCurrentDirectory();
428     var parent = Directory.GetParent(current);
429     Guid result = crossApiProxy.TankManager.ImportTmx(parent.Parent.Parent.FullName + @"\TmxFiles4Tests\test1.tmx", @"test", TestTmxLanguages, "");
430     do
431     {
432         while( crossApiProxy.TankManager.GetJobStatus(result) == 0 );
433     }
434     // Check if an error has occurs during importing corrupted TMX file

```

Result	Test Name	Project	Error Message
Passed	TestAddTranslations	TestProject	
Passed	TestBigTextInsertion	TestProject	
Passed	InvalidCredentialsTestConnection	TestProject	
Passed	TestUpdateTranslation	TestProject	
Passed	TestSearchOnlyNumberDifferences	TestProject	
Passed	TestDeleteTranslation	TestProject	
Passed	TestImportCorruptedTmx	TestProject	
Passed	TestImportTmx	TestProject	
Passed	TestAddTranslationWithSpecialCharacters	TestProject	
Passed	TestConnection	TestProject	

Unit Testing result