Computational Linguistics in the Translator's Workflow—Combining Authoring Tools and Translation Memory Systems

Christoph Rösener
Institute of Applied Information Sciences (IAI)
Martin-Luther-Straße 14
D-66111 Saarbrücken, Germany
chrisr@iai-sb.de

Abstract

In Technical Documentation, Authoring Tools are used to maintain a consistent text quality—especially with regard to the often followed translation of the original documents into several languages using a Translation Memory System. Hitherto these tools have often been used separately one after the other. Additionally Authoring tools often have no linguistic intelligence and thus the quality level of the automated checks is very poor. In this paper I will describe the integration of a linguistically intelligent Authoring Tool into a Translation Memory System, thereby combining linguistic intelligence with the advantages of both systems in a single environment. The system allows you not only the use of common authoring aids (spell, grammar and style checker) in source and target language—by using a single environment the terminology database of the Translation Memory System can be used by the authoring aid to control terminology both in the source and target document. Moreover, the linguistically intelligent Authoring Tool enables automatic extraction of term candidates from existing documents directly to the terminology database of the Translation Memory System.

1 Introduction

The benefit of Authoring Tools, especially in the area of Technical Documentation, is beyond debate (cf. Brockmann, 1997, Huijsen, 1998, Nyberg et al., 2003, Spyridakis et al., 1997). Combined with linguistic intelligence besides spell and grammar checking Authoring Tools are used to check terminology, style, and abbreviations in texts (cf. Bredenkamp et al., 2000, Carl et al., 2002b, Haller 2000, Reuther and Wigger, 2000). In most cases this is done in relation to special style guides and terminology, both given by the respective company (cf. O'Brien, 2003, Reuther, 2003, Shubert et al., 1995). Due to the fact that Authoring Aids are mostly used as a single application, style rules and terminology are kept and maintained in a special database together with the application. Moreover linguistically intelligent Authoring Aids also help the author to extract terminology candidates. Subsequently, where required, these candidates can be directly imported into the stored terminology. To enable this function it is also necessary to have the terminology database integrated in the application.

When translating technical documents it has for many years been common practice to use Translation Memory Systems to improve the consistency of translations. Translation Memory Systems store whole sentences or clauses (segments) and their translations in a multi-language translation memory. When the translator is translating a new document the segments are matched against those already present in the translation memory. This is done with the help of a fuzzy match algorithm, which calculates the degree of similarity between the current source segment and matching source segments from the translation memory. The degree of similarity is expressed as a percentage value (100% match means identical match). The matches are afterwards presented in descending order and the translator can paste them into the new translation.

Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics and Writing, pages 1–6,
Los Angeles, California, June 2010.

Besides the translation memory most of the Translation Memory Systems also have an integrated terminology database. On the basis of this database the system is able to suggest translations of single terms even when there is no match on sentence or clause level. The matching of terms is also done with fuzzy matching algorithms. Therefore related as well as identical terms are found in the database. With this function the Translation Memory System offers to some extent a feature similar to the Authoring Tools mentioned above. Yet the quality of the feature implemented in Translation Memory Systems is not the same due to the fact that in most cases this feature in Translation Memory Systems works without linguistic intelligence.

When using both tools, the Authoring Tool as well as the Translation Memory System as a single application, the main problem becomes immediately apparent: the double terminology database. Two terminology databases—Authoring Tool and Translation Memory System—, which have to be maintained in different applications, mean a lot of redundant work.

The other possibility—regular synchronizing of the databases—is very difficult, because it is not clear, which of the databases is the core database. Generally speaking, the author manually enters new terms in the context of translations into the terminology database of the Translation Memory System, whereas the results of automatic term extractions are stored in the terminology database of the Authoring Tool.

In the following I will present a system where the Authoring Tool is directly integrated in the Translation Memory workflow, thus allowing the handling of terminology in only one core database. It is the integration of CLAT (Controlled Language Authoring Tool) into the Across Translation Memory System—the crossAuthor Linguistic.

2 Description

For the understanding of the system and how the particular components work together it is necessary to begin with a description of the underlying modules. The system consists mainly of three modules: the CLAT/UMMT software package, the Across Language Server and the crossAuthor / crossAuthor Linguistic add-on.

2.1 CLAT/UMMT

CLAT/UMMT is a software package from the IAI (Institut der Gesellschaft zur Förderung der Angewandten Informationsforschung an der Universität des Saarlandes—Institute of the Society for the Promotion of Applied Information Sciences at Saarland University). CLAT is a tool designed to support technical authors in producing high-quality documentation (e.g., according to specific standards (cf. DIN ISO 12620, 1999, Herzog and Mühlbauer, 2007)). This is reached through linguistic correctness and compliance with company-specific requirements. CLAT offers

- spell and grammar checking to verify linguistic correctness
- style and terminology checking to verify compliance with company-specific writing guidelines

The CLAT spelling checker elicits incorrectly spelt or unknown words (e.g., *proplusion*). Besides that the CLAT spelling checker can also be used to check British versus American English (e.g., *organise* vs. *organize*). The CLAT grammar checker elicits grammatically incorrect sentences or parts of sentences (e.g., *He come*). In addition, typography errors that involve more than one word or character are also detected.

Stylistic weaknesses in terms of clarity, understandability, and stylistic appropriateness of sentences or parts of sentences are corrected by the CLAT style checker. Especially complexity issues (e.g., too many uses of *and* and *or*), ambiguity issues (e.g., indefinite or anaphoric expressions, such as *it, they, these,* those), as well as stylistic problems (e.g., contracted forms such as *they've*) are detected. Typically this is done on the basis of company-specific writing rules. Finally, the CLAT terminology checker elicits variants of preferred terms, as well as deprecated terms and admitted terms. When the terminology checker finds a deprecated term in the text, a message is displayed with the corresponding preferred term (e.g., *electric engine*—deprecated term, *electric motor*—preferred term) for correction.

In addition to the four standard checking functions CLAT also offers a function for eliciting term candidates. This function is not a checking function in the sense that it finds linguistic errors or weaknesses. Rather, it supports the terminology

workflow of a company. Nouns that have the properties typical of terms and have not been found during the check in the database, either as correctly used terms or variants or deprecated terms, are listed in a separate display window together with the context they occurred in. The author then has the possibility to decide whether any of the term candidates should be included in the terminology of the respective company.

CLAT checks documents with regard to the control functions mentioned above, reports every rule violation and gives technical authors the opportunity to revise their text and immediately re-check the corrections made. CLAT offers an additional function for working in editors that support tags (e.g., FrameMaker). This function, a context-sensitive search, enables the individual processing of individual tags. With the help of a DTD that must be created especially for CLAT, the CLAT server can process tags differently and ignore their contents entirely or only for individual style rules.

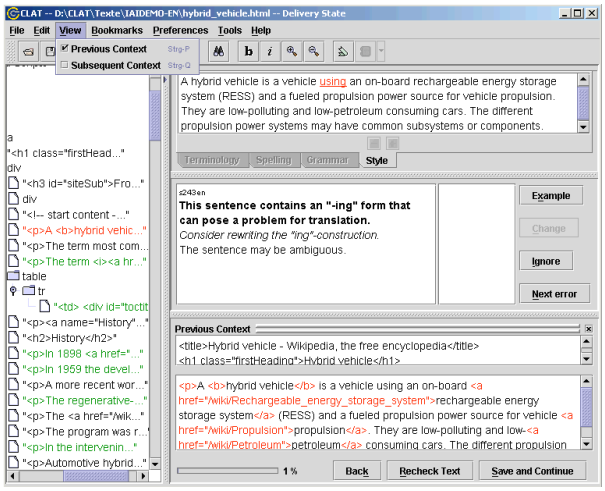

Figure 1. CLAT-Java Client (IAI, 2009a:10).

All CLAT checks are based on a linguistic analysis of the text document. The linguistic analysis consists of several steps that are described in the following:

- separating linguistic from non-linguistic data
- recognizing word boundaries
- analysing word forms: morphological analysis
- determining part of speech: grammatical analysis

The CLAT system consists of a CLAT server and CLAT Clients. CLAT Clients are user interfaces that are either stand-alone (Java CLAT Client) or they are CLAT-Ins that are plugged into an existing word processing program. The CLAT server handles the communication between the CLAT Clients and the Linguistic Engine. The Linguistic Engine is the core part of the CLAT system. It performs the linguistic analysis and the CLAT checks.

The main component of the Linguistic Engine is the program MPRO for the morphological and syntactic analysis of the given text. For further details on MPRO see Maas et al. (2009).

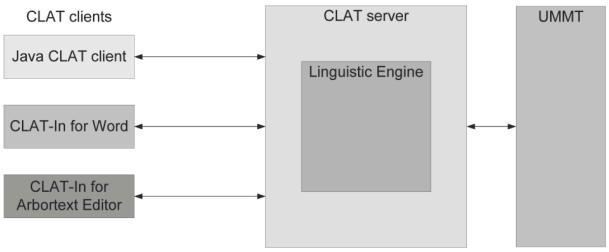

Figure 2. Architecture of the CLAT System (IAI, 2009b:4)

Due to the Linguistic Engine the quality level of the CLAT checks is very high, e.g. the morphological analysis enables CLAT to elicit morphological word form variants such as *electrical battery* as a variant of *electric battery*. Moreover due to the linguistic intelligence CLAT is able to detect even word order variants such as *source of propulsion power* as a variant of *propulsion power source*. These variants are found using complex methods of linguistic abstraction and do not need to be explicitly named in the terminological database (cf. Carl et al., 2002a, Hong et al., 2001, Thurmair, 2003).

Another system component of CLAT is UMMT (Utility for Mandate Management Tasks). It is the central configuration tool for the CLAT-Server. With UMMT, language resources used in CLAT are created, updated, and administered according to the requirements of the respective company.

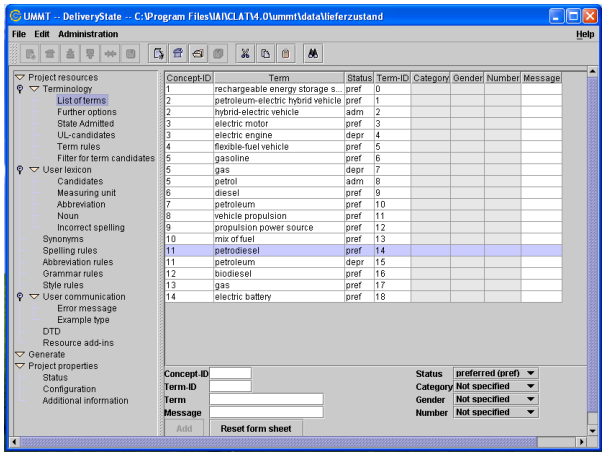

Figure 3. UMMT configuration tool for the CLAT-Server (IAI, 2010:19).

Some of the possible settings for CLAT projects in UMMT are:

- import/maintenance of terminology
- definition of stylistic and grammatical rules
- definition of special spellings and synonyms

All these settings are saved as a project and can be accessed by CLAT at run time. The central project and user management of UMMT allows creation and administration of CLAT projects as well as CLAT users or user groups. CLAT projects can be assigned to one or several CLAT users or user groups. For more detailed information about the CLAT/UMMT Software package see the IAI User Manuals (IAI, 2009a/b).

2.2 Across Language Server

The Across Language Server is the central software platform of the Across language system. The software includes a translation memory, a terminology system, and project management and translation workflow control tools. In this paper I will describe only a few components of the software—the translation memory, the terminology database and the user interface. For further information on the Across Language Server see the Across User Manuals (Across, 2009b/c).

The translation memory within the Across language server—called crossTank—contains sentence pairs from earlier translations. If it finds an identical or similar sentence in a new source text, it offers the stored translation as the basis for an optional automatic pre-translation or as a suggestion, as soon as the translator has arrived at the relevant sentence of the source text in the editor. New translations can either be saved automatically in crossTank or the translator can also choose to save them manually.

The terminology database within the Across Language Server—called crossTerm—enables the translator to create and update multilingual sets of terminology, in particular company-specific terminology and glossaries of technical terms. crossTerm stores concepts and their verbal designations (e. g., translation, synonym, antonym, etc.) for all languages at a single level. It is possible to store many different types of additional information, as well as user-defined information.

Both modules—the translation memory and the database—are integrated in a central user interface—called crossDesk. It provides the translator with a text editor for the source and the target text as well as the functions mentioned above. Matches in crossTank and crossTerm are marked in the source text automatically and can be easily incorporated into the target text.

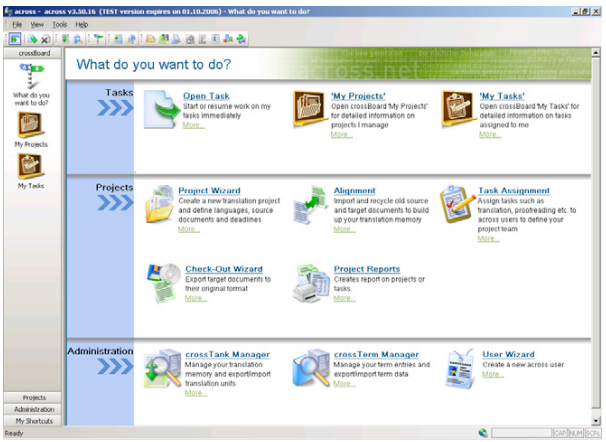

Figure 4. Central user interface crossDesk (Across, 2009d:24).

2.3 crossAuthor / crossAuthor Linguistic

crossAuthor and crossAuthor Linguistic are add-ons for separate source-text editors (e.g., MS Word, Adobe FrameMaker) and provide an interface to the Across Language Server. With crossAuthor, the sentence the user is currently working on in the source-text editor is sent to the Across Language Server. This sentence will then be searched for in crossTank. Relevant search hits are sent back to the user and are displayed in the corresponding crossTank window. At the same time

crossTerm is searched for any corresponding words in the current sentence. The relevant search hits are also transmitted to the crossAuthor add-on via the interface and displayed in the crossTerm window.

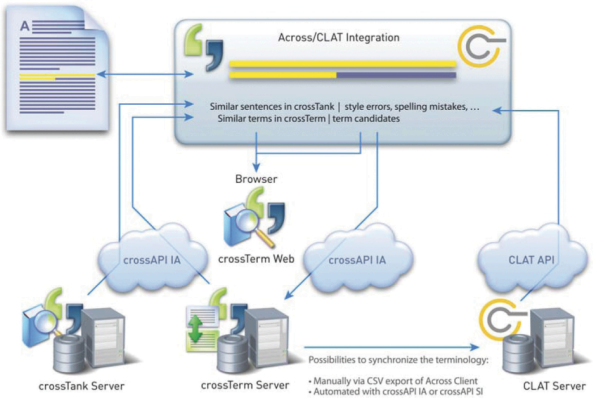

Figure 5. Across/CLAT Integration (Across, 2009a:40).

Finally, crossAuthor Linguistic is an expanded solution of crossAuthor featuring seamless integration of CLAT in Across. With crossAuthor Linguistic users have access to the crossTank and crossTerm matches as well as to the CLAT results.

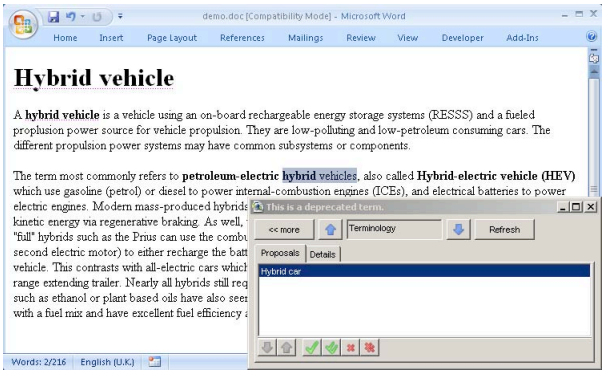

Figure 6. crossAuthor Linguistic correction window in MS Word 2003 (Across, 2009a:48).

Moreover, the CLAT connection enables the direct integration of the editor in the terminology creation process. With the CLAT term-candidate extraction, the editor can directly save auto-detected and extracted terminology as entries in the crossTerm database. As a result, crossTerm works as a core terminology database for both systems. To make the CLAT server work with the respective terminology it is only necessary to import the terminology into UMMT before starting CLAT. This can be done either manually via CSV or automatically by a special interface.

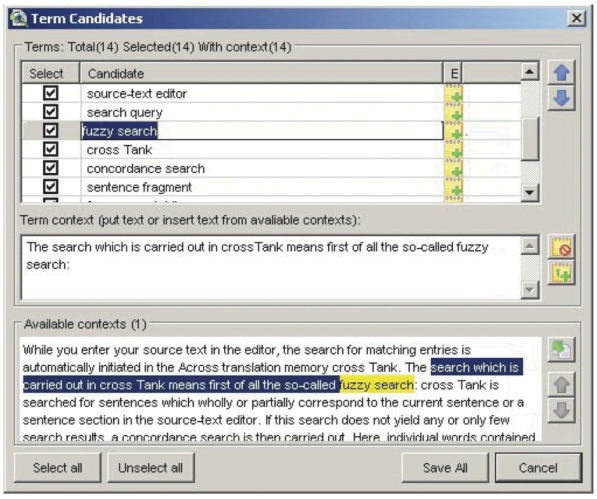

Figure 7. Term-candidate extraction in crossAuthor (Across, 2009a:50).

3 Conclusion

The integration of Authoring tools into Translation Memory Systems is an important step towards a single working environment for the translator's workflow. The system described enables the translator to maintain the terminology for both systems—the Authoring Tool as well as the Translation Memory System—in a single core database. Moreover, the integration of CLAT in Across allows the translator to save auto-detected and extracted term-candidates directly into this database.

Due to the linguistically intelligent analysis in CLAT it is necessary to have specific information about the terms (part of speech, gender, etc.). This information is automatically generated within the CLAT System. This is the reason why the described system actually still has two separate terminology databases. But this is not an important disadvantage of the overall system as long as one of the databases is the core database. The fact, that only one core database has to be maintained means a significant reduction of the workload of translators resp. terminologists.

The system presented is only a first step towards a fully integrated solution for the translators working environment. The focus for future research could be for example the development of an integrated linguistic intelligent Authoring Tool to proof whole translation memories.

References

Across Systems GmbH. 2009a. *User Manual. Translation-Oriented Authoring with crossAuthor / crossAuthor Linguistic v. 5.0.*

Across Systems GmbH. 2009b. *Across Step by Step. Unser Manual v. 5.0.*

Across Systems GmbH. 2009c. *Across at a glance. User Manual v. 5.0.*

Across Systems GmbH. 2009d. *Quickstart Across Language Server v. 5.0.*

Andrew Bredenkamp, Berthold Crysman, and Mirela Petrea. 2000. Building Multilingual Controlled Language Performance Checkers. In Adriaens et al. (Eds), *Proceedings of the Third International Workshop on Controlled Language Applications (CLAW 2000),* Seattle, Washington, pp. 83–89.

Daniel Brockmann. 1997. Controlled Language & Translation Memory Technology: a Perfect Match to Save Translation Cost. *TC Forum 4/97,* pp. 10–11.

Michael Carl, Johann Haller, Christoph Horschmann, and Axel Theofilidis. 2002a. A Hybrid Example-Based Approach for Detecting Terminological Variants in Documents and Lists of Terms. *6. Konferenz zur Verarbeitung natürlicher Sprache, KONVENS,* Saarbrücken. http://www.iai-sb.de/docs/konvens.pdf.

Michael Carl, Johann Haller, Christoph Horschmann, Dieter Maas, and Jörg Schütz. 2002b. *The TETRIS Terminology Tool.* In *TAL*, Vol. 43:1. http://www.iai-sb.de/docs/ tal.pdf.

DIN ISO 12620. 1999. *Computer Applications in Terminology – Data Categories*.

Gottfried Herzog and Holger Mühlbauer. 2007. *Normen für Übersetzer und technische Autoren.* Beuth Verlag GmbH, Berlin.

Johann Haller. 2000. MULTIDOC – Authoring Aids for Multilingual Technical Documentation. *First Congress of Specialized Translation*, Barcelona, March 2000. http://www.iai-sb.de/docs/bcn.pdf.

Willem-Olaf Huijsen. 1998. Controlled Language – An Introduction. In Mitamura et al. (Eds), *Proceedings of the Second International Workshop on Controlled Language Applications – CLAW 98,* Language Technologies Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, pp. 1–15.

Munpyo Hong , Sisay Fissaha, and Johann Haller. 2001. Hybrid filtering for extraction of term candidates from German technical texts. *TIA-2001*, Nancy. http://www.iai-sb.de/docs/term_extract.pdf.

IAI (Institut der Gesellschaft zur Förderung der Angwandten Informationsforschung). 2009a. *CLAT-Client Manual 4.1*

IAI (Institut der Gesellschaft zur Förderung der Angwandten Informationsforschung). 2009b. *CLAT-Introduction Version 4.1*

IAI (Institut der Gesellschaft zur Förderung der Angwandten Informationsforschung). 2010. *UMMT Manual Version 4.1*

Heinz-Dieter Maas, Christoph Rösener, and Axel Theofilidis. 2009. Morphosyntactic and Semantic Analysis of Text: The MPRO Tagging Procedure. In: Cerstin Mahlow and Michael Piotrowski (Eds.): *State of the art in computational morphology. Workshop on systems and frameworks for computational morphology, SFCM 2009, Zurich, Switzerland, September 4, 2009*. Proceedings. New York: Springer (Communications in Computer and Information Science, 41), pp. 76–87.

Eric Nyberg, Teruko Mitamura, and Willem-Olaf Huijsen. 2003. Controlled Language for Authoring and Translation. In H. Somers. (ed), *Computers ad Translation: A Translator's Guide,* Amsterdam, John Benjamins, pp. 245–282.

Sharon O'Brien. 2003. Controlling Controlled English – An Analysis of Several Controlled Language Rule Sets. *Proceedings of the Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Applications Workshop (CLAW 2003)*, 15th–17th May, Dublin City University, Dublin, Ireland, pp. 105–114.

Ursula Reuther. 2003. Two in One – Can it Work? Readability and Translatability by means of Controlled Language. *Proceedings of the Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Applications Workshop (CLAW 2003)*, 15th–17th May, Dublin City University, Dublin, Ireland, pp. 124–132.

Ursula Reuther and Antje Schmidt-Wigger. 2000. Designing a Multi-Purpose CL Application. In Adriaens et al. (eds), *Proceedings of the Third International Workshop on Controlled Language Applications (CLAW 2000),* Seattle, Washington, pp. 72–82.

Serena Shubert, Jan Spyridakis, and Heather Holmback. 1995. The Comprehensibility of Simplified English in Procedures. *Journal of Technical Writing and Communication,* 25(4):347–369.

Jan Spyridakis, Serena Shubert, and Heather Holmback. 1997. Measuring the Translatability of Simplified English in Procedural Documents. *IEEE Transactions on Professional Communication.* 40(1):217–246.

Gregor Thurmair. 2003. Making Term Extraction Tools Usable. *Proceedings of the Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Applications Workshop (CLAW 2003)*, 15th–17th May, Dublin City University, Dublin, Ireland, pp. 170–179.